

DynamicVGGT: Learning Dynamic Point Maps for 4D Scene Reconstruction in Autonomous Driving

Supplementary Material

The following appendices provide additional technical details and experimental results that support the main findings of this work. Section 1 provides a Scene Flow estimation on Waymo[3] dataset. Section 2 provides a detailed description of our complete model architecture. Section 3 outlines the training pipeline and implementation details. Section 4 offers additional qualitative results and visualizations. Section 5 introduces a detailed of Dynamic 3D Gaussian Splatting Pipeline. Finally, Section 6 discusses the limitations of our approach.

1. Scene Flow estimation on Waymo

A key capability of DynamicVGGT is scene motion estimation, which we evaluate on the Waymo Open Dataset. Following [2], we report standard metrics: 3D End-Point Error (EPE3D), Acc5, Acc10, angular error (θ err). We compare against NSFP[2] and STORM[5] in Table 1.

Table 1. Comparison of flow estimation on Waymo(val)

Methods	EPE3D(m) ↓	Acc ₅ ↑	Acc ₁₀ ↑	θ (rad) ↓
NSFP[2]	0.698	42.17	54.26	0.919
STORM[5]	0.276	81.12	85.61	0.658
DynamicVGGT	0.434	67.71	79.90	0.706

2. More Architecture Details

The encoder and alternating attention modules are the same as those in VGGT [4], we use 12 layers for the motion-aware temporal attention modules(MTA) in our main experiment. And we define a set of learnable motion tokens. Specifically, we initialize $N = 16$ learnable motion tokens as a model parameter $M \in \mathbb{R}^{1 \times 2 \times N \times C}$, where C is the embedding dimension. The second dimension of size 2 corresponds to the two consecutive frames involved in the temporal attention computation. These tokens are initialized using a normal distribution with a small standard deviation of $\sigma = 10^{-6}$ to promote stable model convergence at the start of training. The prediction heads for generating future point maps and their associated confidence scores are built upon a DPT head, which we have significantly modified into a specialized multi-view temporal architecture. Instead of using layers from the Alternating Attention (AA) module, our head takes intermediate features from the 2nd, 5th, 8th, and 11th layers of the MTA module. To explicitly model the temporal dependencies crucial for forecasting,

the head incorporates a Multi-Head Attention block that operates along the time dimension, functionally analogous to the core mechanism in the MTA. For Dynamic3DGS head, we use a lightweight convolutional block with a 7×7 kernel, stride 1, and padding 3, followed by a ReLU activation, to extract RGB features from the images. This is crucial because the frozen VGGT backbone is designed for geometric modeling and cannot reconstruct realistic images on its own. The extracted RGB features are then fused with temporal features from the MTA module. After fusion, deep convolutional and nonlinear transformations extract rich 3D spatial and temporal information.

3. Training Details

Our DynamicVGGT model defaults to using 12 layers of Motion-aware Temporal Attention (MTA), resulting in approximately 1.4 billion parameters. The model is initialized from pre-trained VGGT weights and fine-tuned in two stages, optimizing about 800 million parameters in total. For differentiable voxelization, we set the voxel size to 0.002. In Stage 1, we train for 10 epochs using the AdamW optimizer with a hybrid learning rate schedule—linear warm-up for the first 0.5 epoch followed by cosine decay—peaking at a learning rate of 1×10^{-6} . In Stage 2, we further fine-tune the model for 50 epochs with the Gaussian head enabled, using the same schedule but a higher peak learning rate of 5×10^{-5} . All input frames, depth maps, and point maps are resized such that the longer image side does not exceed 518 pixels. The temporal step Δt is randomly sampled from 1 to 3. We adopt a dynamic batch sizing strategy similar to VGGT, processing 12 images per batch. For the training objective, we set $\lambda_1 = 0.01$, $\lambda_2 = 0.1$, and $\lambda_3 = 0.01$. We apply a cosine learning rate scheduler with 8K warm-up iterations and a peak learning rate of 2×10^{-4} . The image aspect ratio is randomly jittered between 0.33 and 1.0 for data augmentation. For training stability, gradient norm clipping with a threshold of 1.0 is used, and gradient checkpointing is enabled to reduce memory consumption.

4. More Visualizations Results

In this section, we provide extended qualitative results, covering point map reconstruction and 3DGS outputs.

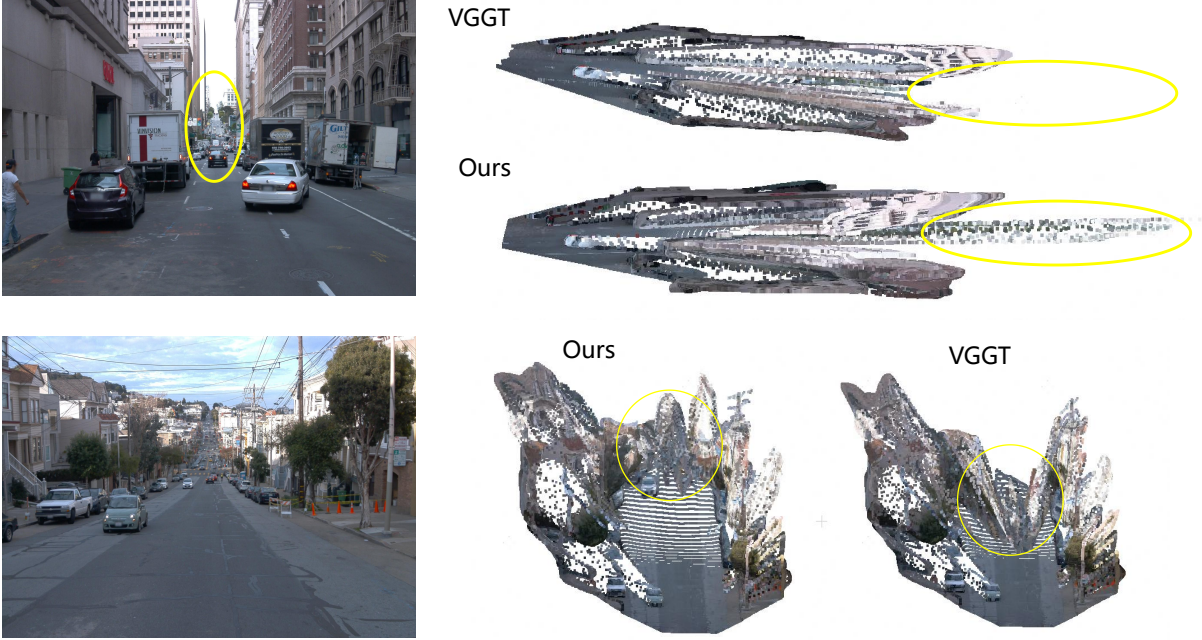


Figure 1. **More compare visualization results of Point Reconstruction.** In outdoor driving scenes, our method achieves noticeably better global geometric reasoning than VGGT.

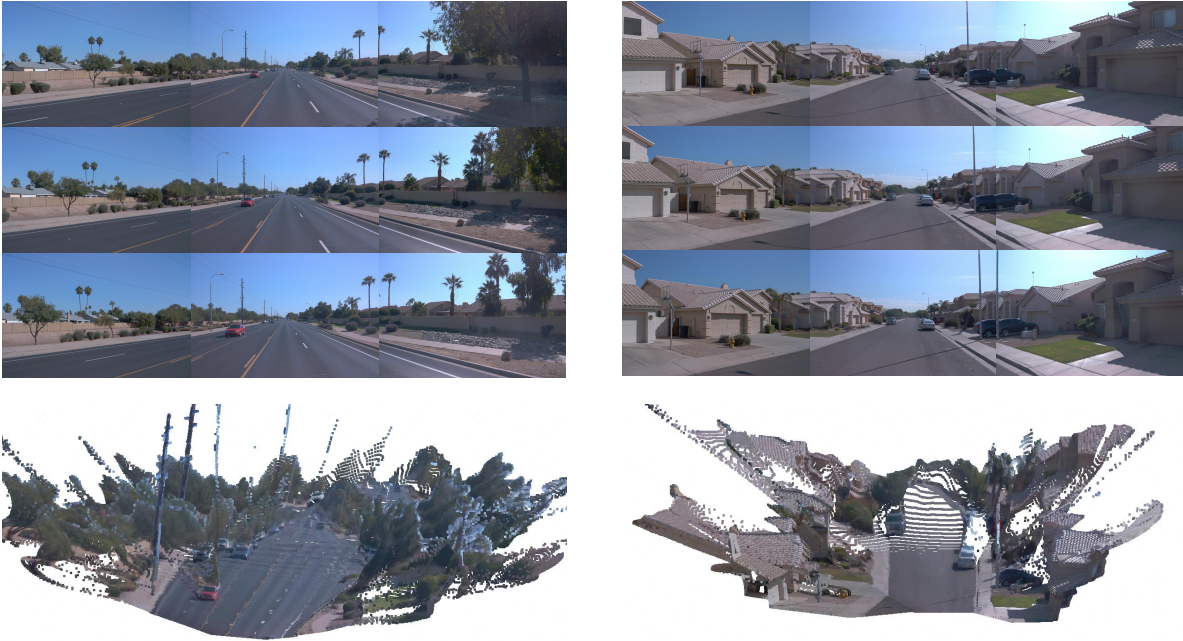


Figure 2. **More visualization results of Point Reconstruction.** We further visualize additional scenes using the same input setting of three consecutive frames sampled at a 0.4 s interval across three camera views. As shown on the right-hand scene, the geometric pose estimated from the left-view camera is less accurate due to the limited overlap between the left and front camera views. In contrast, the other camera views exhibit generally good reconstruction quality with well-aligned point estimates.

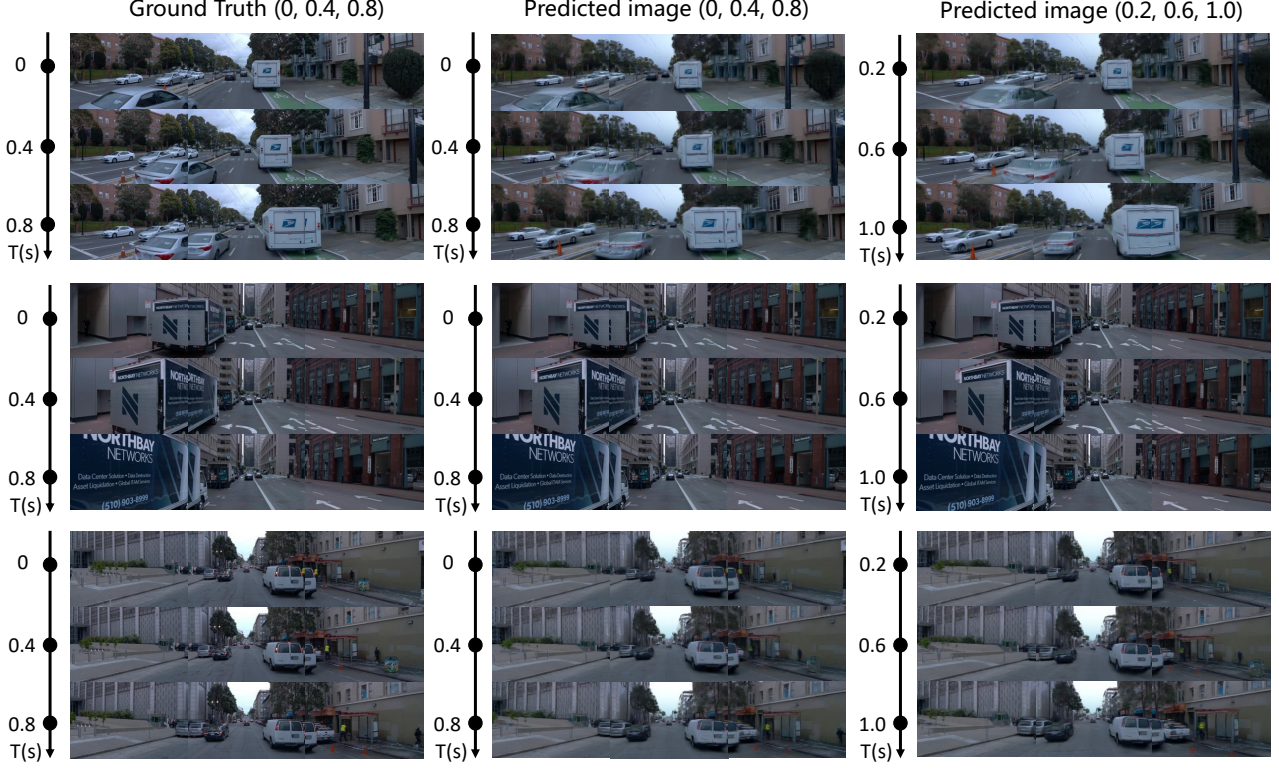


Figure 3. More visualization results of 3DGS image reconstruction and Novel view synthesis.

5. Dynamic 3D Gaussian Splatting Pipeline

Our dynamic 3D Gaussian splatting pipeline transforms 2D image features into structured 3D representations and efficiently renders novel views through differentiable rendering. The pipeline consists of four key components: Gaussian parameter prediction, motion-aware temporal modeling, differentiable rendering with dual passes, and optimization strategies.

5.1. Gaussian Parameter Prediction

Given 2D feature maps $\mathbf{F} \in \mathbb{R}^{B \times T \times V \times C \times H \times W}$ extracted from the encoder, where B denotes batch size, T temporal frames, V camera views, and (H, W) spatial dimensions, we predict Gaussian parameters through a lightweight convolutional head:

$$\mathbf{G}_{\text{raw}} = \text{ConvHead}(\mathbf{F}) \in \mathbb{R}^{B \times S \times H \times W \times D}, \quad (1)$$

where $S = T \times V$ represents the total number of source views, and D is the dimensionality of raw Gaussian parameters. The raw predictions are then split and activated to obtain valid Gaussian parameters:

- **Rotation:** Represented as normalized quaternions for sta-

ble optimization:

$$\mathbf{q} = \frac{\mathbf{q}_{\text{raw}}}{\|\mathbf{q}_{\text{raw}}\|_2 + \epsilon} \in \mathbb{R}^{B \times N \times 4} \quad (2)$$

- **Scale:** Exponential activation ensures positive definiteness of covariance matrices:

$$\mathbf{s} = \exp(\mathbf{s}_{\text{raw}}) \in \mathbb{R}^{B \times N \times 3} \quad (3)$$

- **Opacity:** Sigmoid activation constrains values to valid transparency range:

$$\alpha = \sigma(\alpha_{\text{raw}}) \in \mathbb{R}^{B \times N} \quad (4)$$

- **Appearance:** Spherical harmonics coefficients with residual learning from RGB inputs:

$$\mathbf{sh} = \text{RGB2SH}(\mathbf{I}) + \Delta\mathbf{sh} \in \mathbb{R}^{B \times N \times K \times 3} \quad (5)$$

where $K = (\text{degree} + 1)^2$ is the number of SH coefficients.

- **Weights:** Importance weights for subsequent merging operations:

$$\mathbf{w} = \sigma(\mathbf{w}_{\text{raw}}) \in \mathbb{R}^{B \times N} \quad (6)$$

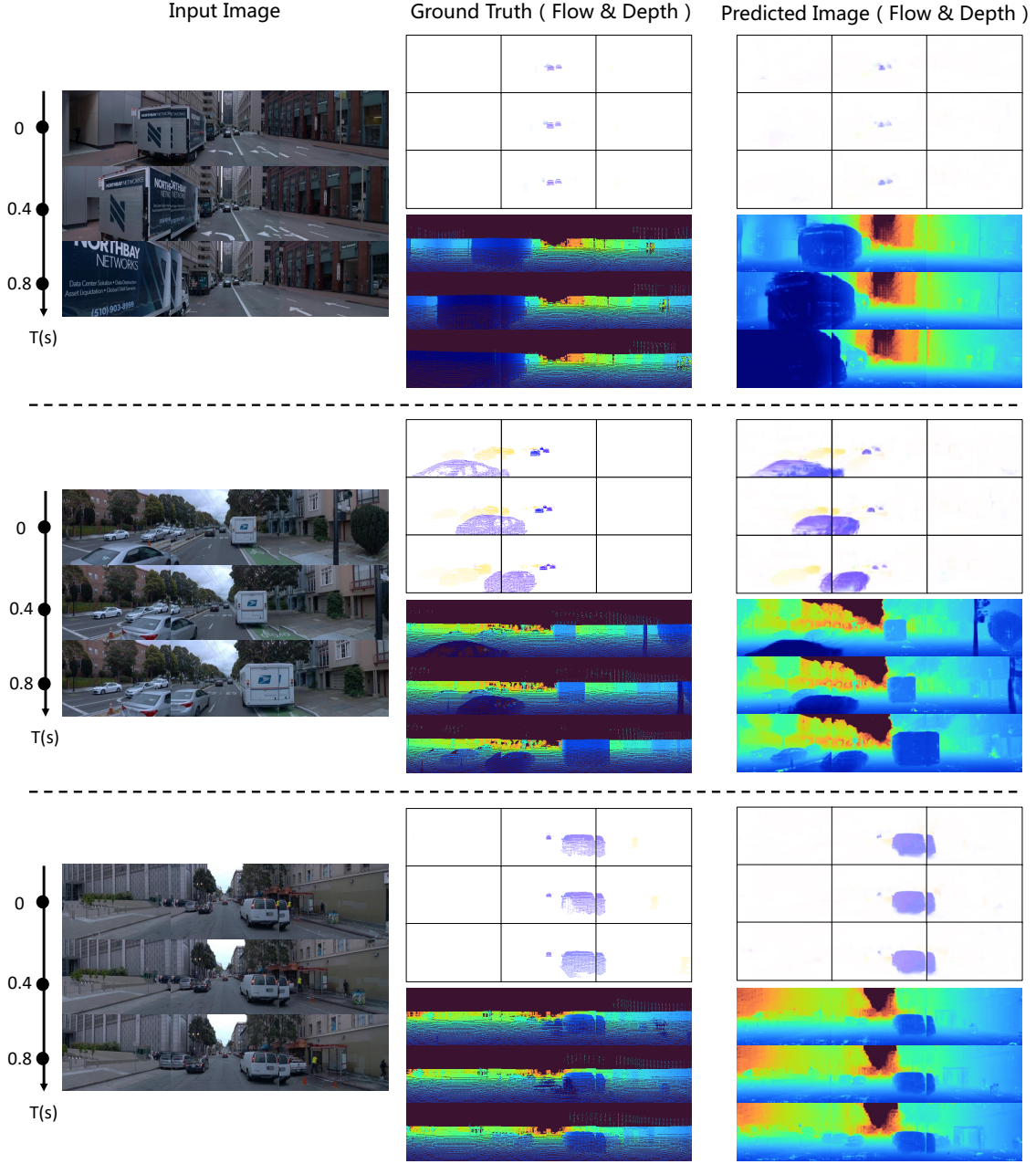


Figure 4. More visualization results of 3DGS depth and flow estimation.

5.2. Motion-Aware Temporal Modeling

For dynamic scene modeling, we introduce a hierarchical motion token mechanism that captures temporal evolution through reusable motion primitives. The motion tokens $\mathbf{M} \in \mathbb{R}^{B \times T \times V \times M \times C}$ from the transformer backbone first undergo token flattening and projection to obtain a fixed-size representation $\mathbf{M}_{\text{proj}} \in \mathbb{R}^{B \times K \times C}$. Each projected to-

ken then generates a specialized query vector through dedicated MLP heads, while the motion basis decoder produces reusable 3D motion primitives $\mathbf{B} \in \mathbb{R}^{B \times K \times 3}$.

The image features \mathbf{F}_{img} are processed to obtain keys $\mathbf{K}_{\text{img}} = \text{MLP}_{\text{key}}(\mathbf{F}_{\text{img}})$. The motion weights are computed via cross-attention between the token queries and image

keys:

$$\mathbf{W}_{\text{motion}} = \text{Softmax} \left(\frac{\mathbf{Q} \cdot \mathbf{K}_{\text{img}}^T}{\tau} \right), \quad (7)$$

where \mathbf{Q} represents the query tokens, \mathbf{K}_{img} is the key from the image features, τ is a temperature parameter used for scaling, which controls the sharpness of the softmax output. The resulting tensor $\mathbf{W}_{\text{motion}}$ contains motion weights for each query-token pair. The velocity field is obtained as a weighted combination of motion bases:

$$\mathbf{v} = \sum_{k=1}^K \mathbf{W}_{\text{motion}}^{(k)} \cdot \mathbf{B}^{(k)}. \quad (8)$$

This represents the motion velocity for each voxel in the scene, where K is the number of motion bases. Temporal propagation of Gaussian positions follows the physical motion model:

$$\mathbf{p}(t) = \mathbf{p}(t_0) + \mathbf{v} \cdot (t - t_0). \quad (9)$$

This design enables efficient motion modeling by decomposing complex scene motion into combinations of reusable motion primitives, providing both computational efficiency and strong generalization across different temporal sequences.

5.3. Differentiable Rendering with Dual Passes

We employ a dual-pass rendering strategy that shares the same Gaussian geometric properties but uses different appearance features. The rendering equation accumulates contributions from all Gaussians for pixel \mathbf{x} :

$$C(\mathbf{x}) = \sum_{i=1}^N c_i \cdot \alpha_i \cdot \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (10)$$

where the blending weight α_i is computed based on the Gaussian's projected covariance and viewing direction. For appearance rendering, we use spherical harmonics coefficients sh_i for photorealistic color reconstruction, outputting RGB images that capture view-dependent lighting effects. For motion rendering, we leverage the same rendering pipeline but use 3D velocity vectors \mathbf{v}_i as appearance attributes:

$$\mathbf{V}(\mathbf{x}) = \sum_{i=1}^N \mathbf{v}_i \cdot \alpha_i \cdot \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (11)$$

This approach is mathematically consistent since the rendering equation's linear accumulation property allows any per-Gaussian 3D attribute to be projected and aggregated using the same blending weights. The motion field $\mathbf{V}(\mathbf{x}) \in \mathbb{R}^3$ represents the aggregated 2D motion vectors at each pixel, maintaining full differentiability for end-to-end motion learning while ensuring geometric consistency between the appearance and motion representations.

5.4. Optimization Strategies

Voxel Pruning Follow Anysplat[1], neighboring Gaussians within spatial voxels are merged through weighted averaging:

$$\mathbf{p}_{\text{merged}} = \frac{\sum_{i \in \mathcal{V}} w_i \cdot \mathbf{p}_i}{\sum_{i \in \mathcal{V}} w_i}, \quad (12)$$

where \mathcal{V} represents Gaussians within the same voxel, and w_i are their importance weights.

6. Limitations

Our model demonstrates robust performance, but it also has several limitations. First, loss constraint design for 3DGS remains relatively simplistic, which may limit optimal performance in image reconstruction. Second, the point cloud generation relies on a multi-layer perceptron with pixel shuffling for interpolation and upsampling. This approach tends to generate floating points, which may lead to estimation errors.

References

- [1] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *arXiv preprint arXiv:2505.23716*, 2025. 5
- [2] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021. 1
- [3] 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. 1
- [4] 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
- [5] Jiawei Yang, Jiahui Huang, Yuxiao Chen, Yan Wang, Boyi Li, Yurong You, Apoorva Sharma, Maximilian Igl, Peter Karkus, Danfei Xu, et al. Storm: Spatio-temporal reconstruction model for large-scale outdoor scenes. *arXiv preprint arXiv:2501.00602*, 2024. 1