STORM: SPATIO-TEMPORAL RECONSTRUCTION MODEL FOR LARGE-SCALE OUTDOOR SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

We present STORM, a spatio-temporal reconstruction model designed to reconstruct in-the-wild dynamic outdoor scenes from sparse observations. Existing dynamic reconstruction methods rely heavily on dense observations across space and time and strong motion supervision, therefore suffering from lengthy optimization time, limited generalizability to novel views or scenes, and degenerated quality caused by noisy pseudo-labels. To bridge the gap, STORM introduces a data-driven Transformer architecture that jointly infers 3D scenes and their dynamics in a single forward pass. A key design of our scene representation is to aggregate 3D Gaussians and their motion predicted from all frames, which are later transformed to the target timestep for a more complete (*i.e.* "amodal") reconstruction at any given time from any viewpoint. As an emergent property, STORM can automatically capture dynamic instances and their high-quality masks using just the reconstruction loss. Extensive experiments show that STORM accurately reconstructs dynamic scenes and outperforms other per-scene optimization (+3.7 PSNR) or feed-forward approaches (+1.5 PSNR); it can reconstruct large-scale outdoor scenes within just 200ms and render in real-time. Beyond reconstruction, we qualitatively demonstrate four additional applications of our model, illustrating the potential of self-supervised learning for advancing dynamic scene understanding. Our code and model will be released.

1 INTRODUCTION

Understanding and reconstructing dynamic 3D scenes from visual data is a fundamental challenge in computer vision, with significant applications in autonomous driving, robotics, and mixed reality, among many others. While static scene reconstruction methods have evolved from per-scene optimization (Mildenhall et al., 2021; Kerbl et al., 2023) to more data-driven approaches that leverage generalizable priors for improved data efficiency (Zhang et al., 2024; Tang et al., 2024; Xu et al., 2024; Gao et al., 2024b; Wu et al., 2024b), most dynamic scene reconstruction methods still rely heavily on per-scene optimization, dense spatiotemporal observations (Park et al., 2021; Yang et al., 2024b), and strong motion supervision, such as dynamic objects' masks (Li et al., 2021b; Wang et al., 2024b), optical flow (Li et al., 2021b), or point trajectories (Wang et al., 2024b). Consequently, these models suffer from noise in the above pseudo-labels, require lengthy training times that range from hours to days, and cannot benefit from the data-driven advancements (*e.g.* scaling laws (Zhai et al., 2022)) that are nowadays leveraged by generalizable static reconstruction methods.

Our goal is to develop a scalable and data-driven solution for dynamic scene reconstruction that overcomes the current limitations. To this end, we present STORM, a self-supervised approach for reconstructing dynamic 3D scene representations and scene motions directly from sparse, multi-timestep, posed camera images. STORM leverages a Transformer model (Vaswani et al., 2017; Dosovitskiy, 2020) to reconstruct dynamic scenes in a *single* feed-forward pass, reducing reconstruction time from hours to seconds while leveraging data priors learned from large-scale datasets. More importantly, unlike existing methods that require pseudo-labels, our approach relies solely on a self-supervised reconstruction loss, allowing for a significantly more cost-efficient data acquisition.

STORM is enabled by our proposed bottom-up amodal aggregation and transformation design. Specifically, for each frame, we predict pixel-aligned or patch-aligned 3D Gaussian Splats (3DGS) (Kerbl et al., 2023) and their dynamics, capturing the instantaneous state of the scene at

each timestep. As such image-aligned 3DGS can only represent the observed region from the context frames, we transform the Gaussians predicted from all the input frames to the target timestep, aggregating to an "amodal" version of the dynamic scene (Huang et al., 2022). By minimizing the reconstruction loss defined over this aggregated representation, our approach achieves accurate, self-supervised estimation of the scene's temporal changes, as inaccuracies in dynamics would lead to poor aggregation and transformation results, ultimately causing large reconstruction errors.

Building upon this foundation, we introduce motion tokens—a set of learnable tokens prepended to the Transformer's input sequence. The motion tokens interact with image tokens through self-attention operations and are decoded as motion bases after the Transformer's forward pass. They are designed to capture common motion primitives over time while also regularizing the degrees of freedom in predicted motions, motivated by the fact that the scene elements often move all together as groups (Wang et al., 2024b; Lei et al., 2024; Luiten et al., 2023). Concretely, we represent the motion of each 3D Gaussian using 3D velocity vectors, with the final motion computed as a weighted combination of shared velocity bases, determined by the similarity between motion tokens and image tokens. As a result, the motion tokens not only capture the low-dimensional structure of scene dynamics but also enable unsupervised dynamic instance or motion group segmentation.

Lastly, we introduce a few practical techniques to enable `STORM` to operate effectively on in-the-wild captures. We address challenges such as sky modeling and camera exposure mismatch using auxiliary sky and affine tokens, and improve large novel view extrapolation and fine-grained human motions, such as leg and arm movements, using latent Gaussians and a latent decoder.

We conduct extensive experiments on the Waymo Open Dataset (Sun et al., 2020) to evaluate the performance of `STORM`. Our results demonstrate that `STORM` accurately reconstructs dynamic scenes in real-time (0.2s for a 2-second clip), significantly surpassing per-scene optimization methods and other generalizable feed-forward models in photorealisim, geometry and motion estimation quality. These findings highlight the potential of self-supervised learning for advancing dynamic scene reconstruction and understanding. Our contributions are summarized as follows:

- We propose `STORM`, a feed-forward and self-supervised method that reconstructs dynamic 3D scenes from sparse, multi-timestep, posed camera images fast and accurately.
- We propose a bottom-up design that aggregates and transforms per-frame 3D Gaussian Splats into a cohesive scene representation, which enables self-supervised motion estimation. Additionally, we introduce motion tokens that capture common motion primitives and regularize motion prediction, facilitating dynamic instance and motion group segmentation without requiring explicit motion or correspondence supervision.
- We present several enhancements for handling in-the-wild scenarios, including sky modeling, camera exposure inconsistency handling, large novel view extrapolation, and fine-grained human motions reconstruction, making `STORM` suitable for real-world applications.

2 Related Work

Dynamic scene reconstruction. Derived from neural radiance fields (NeRFs) (Mildenhall et al., 2021), previous NeRF-based approaches model scene dynamics either by applying deformations to a canonical volume (Pumarola et al., 2021; Tretschk et al., 2021; Cao & Johnson, 2023; Fang et al., 2022; Park et al., 2021; Wu et al., 2022; Fridovich-Keil et al., 2023), or by chaining point-level scene flow motions (Xian et al., 2021; Gao et al., 2021; Li et al., 2021b; 2023; Liu et al., 2023). These per-scene optimization methods typically require dense observations across time and viewpoints or explicit motion supervision, such as optical flow (Li et al., 2021b; Wang et al., 2023; Li et al., 2023; Yang et al., 2023b; Gao et al., 2024a; Karaev et al., 2023; Fischer et al., 2024a) or dynamic masks (Liu et al., 2023; Li et al., 2023), to overcome the ill-posed nature of reconstructing dynamic scenes from sparse views. Building upon this foundation, recent 3D Gaussian Splatting (3DGS)-based methods (Kerbl et al., 2023) similarly apply deformation to the canonical space (Wu et al., 2024a; Yang et al., 2024b) or rigid transformations of particles (Luiten et al., 2023; Wang et al., 2024b), but they still rely on dense views or motion supervision. Importantly, these approaches are limited to per-scene optimization that lacks data priors, except for, *e.g.* Ren et al. (2024) that only focuses on object-scale reconstructions. We instead propose a feed-forward model trained on large-scale data, enabling outdoor dynamic scene reconstruction without per-scene optimization or explicit motion supervision.

Feed-forward reconstruction. Feed-forward approaches for 3D reconstruction and rendering aim to generalize across scenes by learning from large datasets. Early works on generalizable NeRFs focus on object-level (Chibane et al., 2021; Johari et al., 2022; Reizenstein et al., 2021; Yu et al., 2021) and scene-level reconstruction (Suhail et al., 2022; Wang et al., 2021; Du et al., 2023; Wang et al., 2024a). These methods typically rely on epipolar sampling or cost volumes to fuse multi-view features, requiring extensive point sampling for rendering, which results in slow speed and often unsatisfactory details. More recently, feed-forward models based on 3DGS have been proposed (Szymanowicz et al., 2024b; Charatan et al., 2024; Wewer et al., 2024; Zhang et al., 2024; Szymanowicz et al., 2024a; Tang et al., 2024; Xu et al., 2024). These models leverage large-scale object-centric synthetic datasets (Chang et al., 2015; Deitke et al., 2023; 2024) or indoor datasets (Zhou et al., 2018) for improved speed, view synthesis and generalization. However, these methods are primarily designed for static scenes and struggle with dynamics. Unlike them, our approach is a 3DGS-based feed-forward model operated on large-scale outdoor *dynamic* scenes; more importantly, it also recovers scene motions without explicit motion supervision.

Reconstruction for outdoor urban scenes. Building photorealistic reconstructions of dynamic urban scenes from on-car logs is crucial for autonomous driving, as it enables closed-loop training and testing. The focus has shifted from reconstructing static scenes (Guo et al., 2023) to dynamic ones. Most existing methods for dynamic urban scene reconstruction rely on box annotations to enable controllability, but these require expensive ground truth labels (Wu et al., 2023; Chen et al., 2024; Yang et al., 2023a; Tonderski et al., 2024; Fischer et al., 2024b; Zhou et al., 2024a; Ost et al., 2021; Williams et al., 2024; Fischer et al., 2024a) and degenerate with noisy pseudo-labels (Yan et al., 2024; Zhou et al., 2024b). Methods that do not require box annotations typically lack controllability over individual objects (Yang et al., 2024a; Chen et al., 2023; Huang et al., 2024). Furthermore, these approaches are per-scene based, do not leverage data priors, and require lengthy training times ranging from hours (Yan et al., 2024; Yang et al., 2024a) to days (Xie et al., 2023). In contrast, our method is a fast, scalable feed-forward model that reconstructs dynamic urban scenes purely through *self-supervision* in *seconds*. By differentiating between different instance groups *emerged* from our motion tokens, our approach enables better decomposition and controllability. Lastly, optimizing memory and efficiency for very large-scale outdoor scene reconstruction (Lin et al., 2024) complements our work and could further improve our method, which we leave to future exploration.

3 Self-supervised Spatial-Temporal Reconstruction Model

Problem formulation. Our goal is to recover spatiotemporal scene representations from a set of posed images. Specifically, given a set of images $\mathbf{I}_t^v \in \mathbb{R}^{H\times W\times 3}$ with height H and width W captured at multiple timesteps t and optionally multiple viewpoints v, along with their corresponding camera intrinsic and extrinsic parameters, we aim to reconstruct the underlying appearance, geometry, and dynamics of the scene over the observed duration. The core challenge arises from the transient and incomplete nature of the data: each point in the 4D space-time volume is typically observed only *once*, making it difficult to infer a comprehensive spatiotemporal representation.

3.1 STORM

To address the above-mentioned challenges, we propose `STORM`, illustrated in Fig. 1. We adopt a Lagrangian representation by modeling scene elements as a set of 3D Gaussians (3DGS) (Kerbl et al., 2023) that translate over time. Specifically, we begin by predicting 3DGS for each frame, which captures the instantaneous state of the scene at each timestep. To model dynamics, we task the model with predicting the motion of each Gaussian, parameterized by velocities. With these velocities, we could transform the 3DGS from their observed context timesteps into any target timestep. This process aggregates the per-frame predictions into a cohesive *amodal* representation that is consistent over time. Notably, our method trains solely with the reconstruction loss, without relying on any external motion supervision such as optical flow, scene flow, dynamic masks, or point trajectories, which significantly lift the data requirements. Below, we introduce our method in detail.

Network and Input. `STORM` builds upon a standard full-attention Transformer model (Vaswani et al., 2017; Dosovitskiy, 2020), similar to Zhang et al. (2024). Following standard Vision Transformers (Dosovitskiy, 2020), we divide images into 2D non-overlapping patches, which are then embedded through a linear patch embedding layer to obtain image tokens. To incorporate 3D spatial information, we generate ray tokens by passing the *patchified* Plücker ray map (Plucker, 1865), that encodes the ray origin and direction at each pixel, through a linear embedding layer. These ray ori-

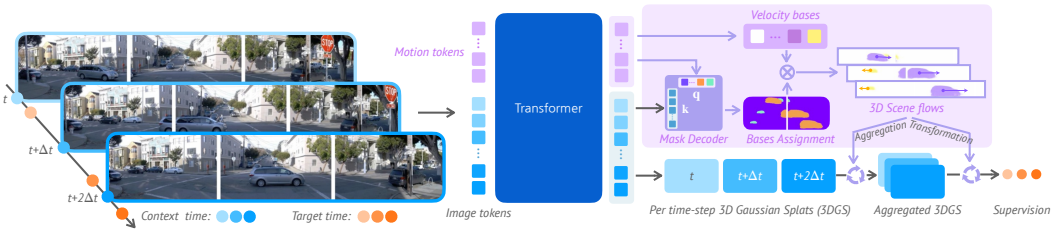


Figure 1: **STORM Overview.** From sparsely observed context frames, STORM reconstructs per-frame 3D Gaussian splats (3DGS) and predicts their scene flows using prepended learnable motion tokens and a dynamic mask decoder. The mask decoder computes weights for combining motion bases, derived from the motion tokens, to obtain scene flows. These predicted scene flows enable the aggregation and transformation of 3DGS over time, and the predicted weights support unsupervised motion group segmentation. The learning process is guided purely by reconstruction losses.

gins and directions are computed based on the camera's intrinsic and extrinsic parameters. Lastly, temporal information is infused via a time embedding layer (Peebles & Xie, 2023), which maps a frequency-encoded time vector into a time token. The final input to our Transformer model is a 1D sequence formed by combining image tokens, ray tokens, and time tokens.

Output. The main output of our model is a set of pixel-aligned 3D Gaussians (Kerbl et al., 2023), each defined as $\mathbf{g} \equiv (\boldsymbol{\mu}, \mathbf{R}, \mathbf{s}, o, \mathbf{c})$, where $\boldsymbol{\mu} \in \mathbb{R}^3$ and $\mathbf{R} \in \mathbb{SO}(3)$ represent the center and orientation, $\mathbf{s} \in \mathbb{R}^3$ indicates the scale, $o \in \mathbb{R}^+$ denotes the opacity, and $\mathbf{c} \in \mathbb{R}^3$ corresponds to the color. The orientation is parameterized with a quaternion following Kerbl et al. (2023) using a normalized 4-dimensional vector. The centers of 3D Gaussians are recovered from ray origins and directions that are pre-computed from camera parameters and the ray distance by $\boldsymbol{\mu} = \text{ray}_o + d \cdot \text{ray}_{dir}$, where d is the 1-channel ray-distance predicted by our model. Together, by default, the output from our model is $\{\mathbf{G}_t^v \in \mathbb{R}^{H \times W \times 12}\}$ before activation and normalization. These parameters are predicted from the ViT feature map $\{\mathbf{F}_t^v \in \mathbb{R}^{H//p \times W//p \times e}\}$ using a linear layer, where p denotes the patch size of the Transformer, and e represents the channel dimension. These 3D Gaussians live in 3D space independently in each timestep. Next, we describe how we obtain their dynamics and aggregate them to form the final amodal, synchronized scene representations. We omit the view index v unless necessary.

Scene dynamics. To capture the dynamics of a 3D scene, we model the motion of each 3D Gaussian using two velocity vectors $\mathbf{v} \equiv [\mathbf{v}_t^-, \mathbf{v}_t^+] \in \mathbb{R}^6$ (we will detail how to compute them later), which denotes the backward/forward speed of a Gaussian at timestep t. Empirically, we find that assuming the Gaussians to translate with a constant speed within the given clip (usually just 2 seconds) can reach a good balance between model complexity and representation power. We hence specify the translation of the Gaussian at time t' using:

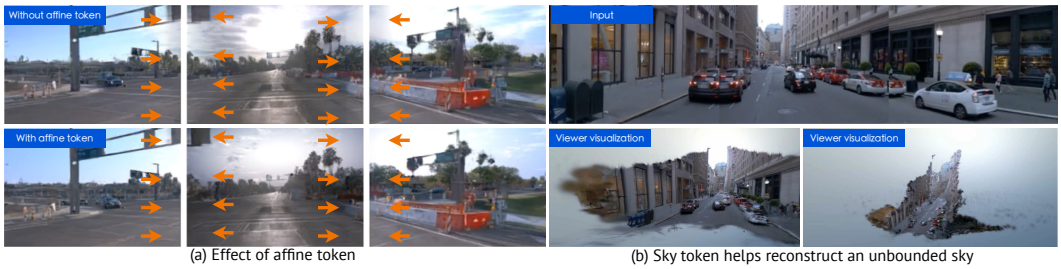
$$\boldsymbol{\mu}_{t \to t'} = \begin{cases} \boldsymbol{\mu}_t - (t' - t)\mathbf{v}_t^- & t' < t \\ \boldsymbol{\mu}_t + (t' - t)\mathbf{v}_t^+ & t' > t \end{cases}. \tag{1}$$

Amodal aggregation. To create a unified representation of the scene that is consistent over time, we aggregate the per-frame 3D Gaussians into an *amodal*, synchronized representation. Specifically, the Gaussians $\mathcal{G}_{t'}$ at an arbitrary render target timestep t' is the following union:

$$\mathcal{G}_{t'} = \bigcup_t \mathbf{G}_{t \to t'}, \tag{2}$$

where $\mathbf{G}_{t \to t'}$ contains translated Gaussians with centers $\boldsymbol{\mu}_{t \to t'}$ from the prediction $\mathbf{G}_t$. This amodal representation combines observations from various moments, capturing the complete geometry and appearance of the scene, as well as the underlying dynamics. It allows us to reason about the scene holistically and facilitates tasks such as rendering the scene from novel viewpoints and times.

Motion tokens and mask decoder. With the common wisdom that scene dynamics often exhibit low-dimensional structures composed of shared motion patterns (Wang et al., 2024b; Kratimenos et al., 2023; Lei et al., 2024), we introduce M learnable *motion tokens* (indexed by m), where $M \ll N$ and N is the number of image tokens. These motion tokens are prepended to the input sequence



(a) Effect of affine token (b) Sky token helps reconstruct an unbounded sky

Figure 2: **Effect of affine token and sky token.** (a) The affine token handles exposure mismatches between cameras, eliminating artifacts like the black foggy floaters caused by exposure differences (orange arrows). (b) The sky token enables our method to predict sky colors for every pixel during rendering, even when they are not observed in any context frames.

of the Transformer and interact with all other input tokens via self-attention. STORM leverages these tokens to capture common motion primitives present in the scene over time. At the end of the Transformer, the motion tokens are decoded into *velocity bases* $\mathbf{vb} \equiv (\mathbf{vb}^-, \mathbf{vb}^+) \in \mathbb{R}^{M\times 6}$ through a linear layer and *motion queries* $\mathbf{q} \in \mathbb{R}^{M\times e'}$ via a set of Multi-Layer Perceptrons (MLPs)[1], where e' is the dimension of the motion embedding space. The image embeddings $\mathbf{F} \in \mathbb{R}^{(H//p\times W//p)\times e}$ are also mapped to this space to produce *pixel-aligned motion keys* $\mathbf{k} \in \mathbb{R}^{(H\times W)\times e'}$ using several deconvolution layers. Here each key vector $\mathbf{k}_{i,j} \in \mathbb{R}^{e'}$ corresponds to a spatial location (i, j).

Inspired by SAM (Kirillov et al., 2023), we leverage the similarity between the motion queries $\mathbf{q}$ and the motion keys $\mathbf{k}$ to compute the weights $\mathbf{w} \in \mathbb{R}_+^{(H\times W)\times M}$ for combining the velocity bases. Specifically, the weights $\mathbf{w}^{(i,j)} \in \mathbb{R}_+^M$ at each spatial location (i, j) are computed by:

$$w_m^{(i,j)} = \frac{\exp\left(\frac{\mathbf{q}_m \cdot \mathbf{k}_{i,j}}{\tau}\right)}{\sum_{m'=1}^{M} \exp\left(\frac{\mathbf{q}_{m'} \cdot \mathbf{k}_{i,j}}{\tau}\right)}, \tag{3}$$

where τ is a temperature hyperparameter controlling the sharpness of the distribution (we set $\tau = 0.5$ in all experiments). The weights $\mathbf{w}$ are then used to combine the velocity bases for each Gaussian associated with each pixel at location (i, j), reaching the final velocity vectors $\mathbf{v}$ used in Eq. (1):

$$\mathbf{v}^{(i,j)} = \sum_{m=1}^{M} w_m^{(i,j)} \mathbf{vb}_m, \quad \text{where } \sum_{m=1}^{M} w_m^{(i,j)} = 1 \text{ and } 0 \le w_m^{(i,j)} \le 1. \tag{4}$$

This design aims to capture the low-dimensional structure of scene dynamics and to regularize the motion prediction problem by reducing its degrees of freedom.

3.2 STORM IN THE WILD

Modeling unbounded scenes from multi-view videos captured in the wild introduces additional challenges, including representing sky, managing lighting variations over time, handling exposure differences between cameras, and accurately modeling humans.

Auxiliary tokens for sky and exposure mismatch. In in-the-wild video collections, especially those from autonomous vehicles, sky modeling and exposure mismatches are common issues. Specifically, the sky often lacks well-defined depth, and a same 3D point may appear with varying colors in different images due to exposure differences across cameras. To address these, we introduce two types of learnable auxiliary tokens into the input sequence: the *sky token* and the *affine token*, designed similarly to the motion tokens.

A single sky token is used to capture sky information. At the output of the Transformer, this sky token conditions a modulated MLP that takes the ray direction $\mathbf{d}$ as input and outputs the sky color:

$$\mathbf{c}_{\text{sky}} = \text{MLP}_{\text{sky}}\left(\gamma\left(\mathbf{d}\right); \text{sky\_token}\right), \tag{5}$$

[1]Following the mask decoder design of SAM (Kirillov et al., 2023), we use different MLP weights for different motion tokens, as we find this results in a cleaner motion mask.

Figure 3: **Latent-STORM Examples.** Using latent Gaussians and a decoder, Latent-STORM can photorealistically reconstruct human leg movements. Note how the leg angles change over time in Latent-STORM, while they remain static in regular STORM (blue arrows). Please refer to "Human Modeling with Latent-STORM" section in our anonymous page for video comparisons.

where $\gamma(\cdot)$ is a frequency-based positional embedding function as in Mildenhall et al. (2021). This setup allows us to query sky colors for every pixel we wish to render. Given a rendered image $\mathbf{I}_{\text{GS}}$ before sky composition and a rendered opacity map $\hat{\mathbf{O}}$, the final image combining the sky is:

$$\mathbf{I}' = \mathbf{I}_{\text{GS}} + (1 - \hat{\mathbf{O}}) \cdot \mathbf{c}_{\text{sky}}. \tag{6}$$

To handle exposure mismatches across different cameras, we introduce an affine token, repeating it v times corresponding to the number of cameras. These tokens aim to capture exposure variations between cameras. At the Transformer's output, each affine token is mapped to a scaling matrix $\mathbf{S} \in \mathbb{R}^{3\times3}$ and a bias vector $\mathbf{b} \in \mathbb{R}^3$ via a linear layer. The final rendered image is obtained by applying the affine transformation to every pixel: $\hat{\mathbf{I}} = \mathbf{S}\mathbf{I}' + \mathbf{b}$. The affine transformation is similarly explored in previous work (Rematas et al., 2022) but in a per-scene optimization setting. We provide examples in Fig. 2 to illustrate the roles of these two token types.

Latent-STORM. As an *optional* enhancement, we introduce the use of latent Gaussians coupled with a latent decoder to improve STORM's performance on large novel view extrapolation and human body modeling. Instead of predicting pixel-aligned Gaussians with a 3-channel color vector, we predict *patch-aligned* Gaussians with a c-channel latent vector. This approach reduces the number of Gaussians while increasing the modeling capacity of each Gaussian. As a result, the output from our model changes from $\mathbf{G}_t^v \in \mathbb{R}^{H\times W\times 12}$ to $\mathbf{G}_t^v \in \mathbb{R}^{H/p\times W/p\times(9+c)}$, where p is the patch size of the Transformer. After rasterization, we obtain a $p\times$ downsampled latent feature map $\mathbf{F}$, which is then composited with a learnable *inpainting token* using the opacity map: $\hat{\mathbf{F}} = \mathbf{F} + (1 - \hat{\mathbf{O}}) \cdot \text{inpainting\_token}$. This composited feature map is upsampled to the original resolution using a convolutional decoder (Rombach et al., 2022) to produce color and depth outputs.

This design addresses the limitations of color-based Gaussians in handling occluded regions that are not visible in any of the context views, since the decoder can infer and reconstruct these unseen areas from the inpatinting tokens within a reasonable extrapolation range.[2] Additionally, modeling appearance changes over time due to lighting effects—such as dimmed surfaces or shadow variations—is facilitated by the decoder's ability to handle complex visual effects.

Lastly, capturing fine-grained human motions, such as leg and arm movements, is challenging with sparse observations. We find that the decoder can photorealistically recover these subtle motions, albeit with a slight compromise in real-time performance due to the additional decoding process. Fig. 3 compares this property. We name this new variant of our method as Latent-STORM.

3.3 Implementation

Model architecture. By default, we use a 12-layer Vision Transformer (ViT-B) (Dosovitskiy, 2020) with full attention and a patch size of 8, along with $M = 16$ motion tokens. For the mask decoder and MLP components, we adopt the implementation from SAM (Kirillov et al., 2023), and set the final projected motion queries and keys to be 32 dimensions. For the modulated MLP for sky modeling, we follow DiT (Peebles & Xie, 2023) to use adaptive layer norm for modulation. Our Gaussian Splatting backend is based on `gsplat` (Ye et al., 2024). Further details are provided in Appendix.

[2]Significant hallucination requires strong generative capabilities, which we leave for future work.

Supervision and loss functions. After aggregating the amodal scene representation $\mathbf{G}$ from all observed timesteps, we transform it into the target timesteps we wish to render. During training, we randomly select a start timestep t and sample 4 target timesteps within the range $[t, t + 2\text{s}]$ for supervision. Using the predicted, aggregated, and transformed amodal Gaussians, we minimize reconstruction loss, sky loss, and velocity regularization loss:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{sky}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}, \tag{7}$$

where reconstruction loss $\mathcal{L}_{\text{recon}}$ includes RGB loss, perceptual loss, and depth loss. The sky loss $\mathcal{L}_{\text{sky}}$ encourages zero opacity for Gaussians on the sky-region. The velocity regularization loss is implemented as $\mathcal{L}_{\text{reg}} = \|\mathbf{v}\|_2$ where we encourage the velocity vectors predicted from all Gaussians from all timesteps to be small. We postpone the details of training, implementation and loss functions to Appendix A.

4 Experiments

Dataset. We primarily conduct experiments on the Waymo Open Dataset (Sun et al., 2020), which contains 1,000 sequences of driving logs collected from autonomous vehicles: 798 sequences for training and 202 for validation. Each sequence consists of a 20-second video recorded at 10FPS. For training and testing, we use the frontal three cameras at an $8\times$ downsampled resolution (160×240). The input to our model consists of 4 context timesteps, evenly spaced at $t + 0$s, $t + 0.5$s, $t + 1.0$s, and $t + 1.5$s, where t is a randomly chosen start timestep.

4.1 Rendering

Setup. We assess novel view synthesis from sparse view reconstructions using the validation set of the Waymo Open Dataset (Sun et al., 2020). Each video sequence is segmented into 10 non-overlapping clips, each 2.0 seconds long and consisting of 20 frames (3 camera views per frame). For reconstruction, we provide the 1st, 5th, 10th, and 15th frames as context frames, and evaluate on the remaining frames. This setup enables evaluation of both interpolation (0s to 1.5s) and extrapolation (1.5s to 2.0s), resulting in 2,019 video clips, or 96,912 total images. We report standard metrics: PSNR, SSIM, and Depth RMSE. Additionally, we analyze performance on both full images and dynamic regions for a more comprehensive evaluation. Lastly, we report the inference time. For per-scene optimization methods, this refers to the test-time fitting time, while for generalizable methods, it refers to the time required for the model to feedforward and output 3D Gaussians before rendering.

Baselines. We compare our method against two categories of approaches: per-scene optimization methods and feed-forward models. For per-scene optimization, we evaluate against a neural field-based approach, EmerNeRF (Yang et al., 2024a), and 3DGS-based approaches, including 3DGS (Kerbl et al., 2023), PVG (Chen et al., 2023), and DeformableGS (Yang et al., 2024b). Since LiDAR data is not provided at *test* time in our setup, we train these baselines without LiDAR supervision to ensure a fair comparison. In the second category, we compare against recent large reconstruction models, including LGM (Tang et al., 2024) and GS-LRM (Zhang et al., 2024). We notice that the default LGM, which predicts raw 3DGS coordinates, performs poorly. Instead, we modify it to predict depth and recover positions similar to our approach, and denote it as LGM*.

Results. We present the quantitative results in Table 1. Compared to per-scene optimization methods, `STORM` achieves significantly better performance in dynamic regions and full images in terms of photorealism, geometry, and inference speed. Specifically, in dynamic regions, `STORM` outperforms the best per-scene method by a substantial 5dB in PSNR and 0.346 SSIM. For full images, `STORM` attains around 0.5 to 1dB PSNR gain. Notably, `STORM` achieves these improvements while reducing inference time from tens of minutes to just 0.18 second, making it suitable for real-time applications. This confirms our motivation of building data-driven models that excels with data priors, addressing the limitations of per-scene optimization methods. Compared to other generalizable feed-forward models, `STORM` demonstrates a robust ability to model scene dynamics and process multi-timestep, multi-view images holistically. This enables us to model dynamic scenes better. To our knowledge, `STORM` represents the first feed-forward reconstruction method for dynamic scenes, and we hope that our high-level ideas and approach will benefit future research.

Results on additional datasets. We evaluate the applicability of `STORM` on the NuScenes (Caesar et al., 2020) and Argoverse2 (Wilson et al.) datasets, comparing against other generalizable methods in Table 2. We provide detailed setups in Appendix B.1. Our method achieves the best performance

Table 1: **Comparison to state-of-the-art methods on the Waymo Open Dataset.** We compare photorealism, geometry and speed metrics against both per-scene optimization methods and generalizable feed-forward methods. PSNR, SSIM, and Depth RMSE (D-RMSE) are reported. Speed metrics are estimated on a single A100 GPU. *: reproduced by us.

Methods	Dynamic-only			Full image			Inference speed	Real-time rendering
	PSNR↑	SSIM↑	D-RMSE↓	PSNR↑	SSIM↑	D-RMSE↓	Time↓	(>200FPS)
Per-Scene Optimization methods								
EmerNeRF (Yang et al., 2024a)	17.79	0.255	40.88	24.51	0.738	33.99	14min	×
3DGS (Kerbl et al., 2023)	17.13	0.267	13.88	25.13	0.741	19.68	23min	✓
PVG (Chen et al., 2023)	15.51	0.128	15.91	22.38	0.661	13.01	27min	✓
DeformableGS (Yang et al., 2024b)	17.10	0.266	12.14	25.29	0.761	14.79	29min	✓
Generalizable feed-forward methods								
LGM (Tang et al., 2024)	17.36	0.216	11.09	18.53	0.447	9.07	0.06s	✓
LGM* (Tang et al., 2024)	19.58	0.443	9.43	23.59	0.691	8.02	0.06s	✓
GS-LRM* (Zhang et al., 2024)	20.02	0.520	9.95	25.18	0.753	7.94	**0.02s**	✓
Ours								
Latent-STORM	21.26	0.535	9.42	25.03	0.750	8.57	0.18s	✓
STORM	**22.10**	**0.624**	**7.50**	**26.38**	**0.794**	**5.48**	0.18s	✓

Table 2: **Comparison to state-of-the-art methods on additional datasets.** We report full-image PSNR and Depth RMSE metrics on the NuScenes (Caesar et al., 2020) and Argoverse2 (Wilson et al.) datasets.

Method	NuScenes		Argoverse2	
	PSNR↑	D-RMSE↓	PSNR↑	D-RMSE↓
LGM	23.21	7.34	22.93	14.20
GS-LRM	24.53	7.71	24.49	14.70
Ours	**24.90**	**5.43**	**24.80**	**13.51**

Table 3: **Comparison on scene flow estimation on the Waymo Open Dataset.** All methods require LiDAR input at test time, whereas our method relies solely on camera images, making the comparisons highly conservative. Zoom-in required.

Methods	EPE3D $(m)\downarrow$	$\text{Acc}_5(\%)\uparrow$	$\text{Acc}_{10}(\%)\uparrow$	θ (rad) $\downarrow$	Inference Time $\downarrow$
NSFP (Li et al., 2021a)	0.698	42.17	54.26	0.919	∼27s/frame
NSFP++ (Najibi et al., 2022)	0.711	53.10	63.02	0.989	∼167s/frame
Ours	**0.276**	**81.12**	**85.61**	**0.658**	∼0.025s/frame

in both *full*-image PSNR and Depth RMSE metrics. Measuring performance on dynamic regions is expected to have more gains. These results validate the generalizability of STORM across datasets.

4.2 Flow Estimation

Setup and baselines. A unique capability of STORM is scene motion estimation, which we demonstrate using the Waymo Open Dataset (Sun et al., 2020). This dataset provides ground truth 3D scene flows, which we *do not* use for supervision. We measure 3D scene flow estimation accuracy using standard metrics following Li et al. (2021a): End-Point Error in 3D (EPE3D), Acc_5, Acc_{10}, angular error θ_{err}, and inference time with definitions provided in Appendix. For baselines, we compare STORM against NSFP (Li et al., 2021a), and NSFP++ (Najibi et al., 2022). Since existing methods cannot directly synthesize scene flows at novel views, we evaluate the scene flows estimated on the context frames. Specifically, we provide the 1st, 5th, 10th, and 15th sensor observations as input and evaluate on these frames rather than on the remaining ones. Notably, all these state-of-the-art methods require LiDAR input at *test time*, whereas STORM relies *solely* on camera images, making our comparisons highly conservative.

Results. As presented in Table 3, STORM consistently outperforms all methods across all metrics, achieving marked improvements in EPE3D, Acc_5 and Acc_{10} despite using only camera images as input. While NSFP (Li et al., 2021a) and NSFP++ (Najibi et al., 2022) excel at scene flow estimation with dense space-time observations (10Hz), they struggle with sparse observations (2Hz). In contrast, STORM demonstrates robust performance even with sparse data. To the best of our knowledge, STORM is the first scene flow estimation method that does not require depth signals at test time. These results demonstrate the effectiveness of our approach in explicit motion understanding and its potential for scene flow estimation without reliance on additional sensors at test time.

4.3 Ablation Study, Qualitative Results and Application

Ablation study. We analyze the impact of the velocity regularization coefficient λ_{reg}, number of motion tokens M, and input timesteps during training and testing in Appendix B.2. In short, without velocity regularization, training collapses because of gradient explosion, and an optimal λ_{reg} ($5e$-3) yields the best performance (which we use in all our tables). STORM is robust to the choice of M

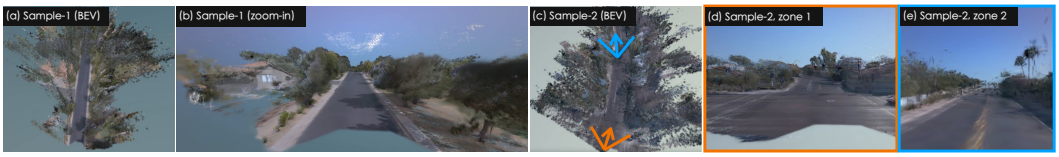


Figure 4: **Iterative reconstruction of static scenes.** STORM reconstructs 20-second-long videos within 1 second in an iterative manner, which can serve as initialization for per-scene optimization methods for further refinement.

Figure 5: **Iterative reconstruction of dynamic scenes. Top**: STORM reconstructs 20-second-long videos within 1 second in an iterative manner. **Bottom**: Furthermore, by chaining scene flows, we obtain point trajectories for dynamic Gaussians.

and performs best with $M = 16$ for both rendering and flow estimation tasks. Furthermore, when trained on a fixed number of timesteps (*e.g.*, 4), STORM demonstrates zero-shot generalization to varying input timesteps (*e.g.*, 1, 2, 6, 10), though re-training for specific configurations achieves optimal results.

Larger scene reconstruction. Figs. 4 and 5 show the results of applying STORM iteratively to 20-second posed videos, each includes 600 images captured across 3 cameras over 200 timesteps. We process videos clip by clip, completing the inference for an entire video within **1 second** on a single A100 GPU. By merging the Gaussians predicted from each clip, we achieve a comprehensive 4D scene reconstruction in a feedforward manner. These results highlight STORM's potential for holistic dynamic scene reconstruction, even in long, complex sequences. Additionally, the predicted 3D Gaussians can serve as initialization for per-scene optimization methods for further refinement, which we leave for future work.

Point tracking. While per-point trajectory estimation is not the primary focus of our work, STORM models the motion of Gaussians over time, enabling us to derive point trajectories by chaining scene flows. In the bottom row of Fig. 5, we present examples of point tracking, demonstrating STORM's potential for applications such as motion analysis and object tracking. We believe this approach could inspire further exploration in related tasks, such as 2D pixel or 3D point tracking.

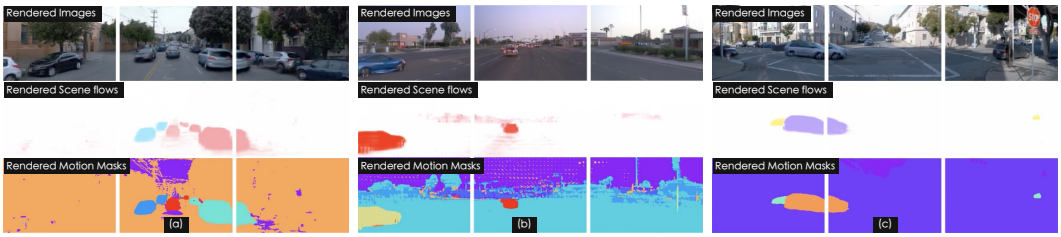


Figure 6: **Self-supervised scene flow estimation and motion segmentation.** For each sample, we show the rendered camera images (top), scene flows (middle), and motion assignments (bottom).

Scene flow estimation and motion segmentation. We visualize the predicted 3D velocities and motion token assignments, *i.e.*, the motion segmentation mask decoded by the mask decoder, in Fig. 6.

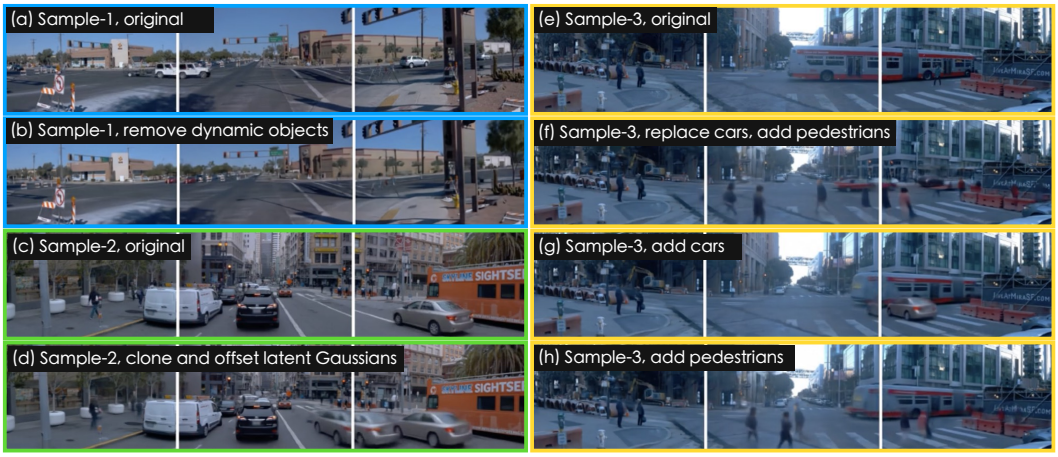


Figure 7: **STORM editing examples.** We present examples of removing or cloning vehicles (a-d), as well as adding or replacing pedestrians and vehicles (e-h). Notice how STORM harmonizes the edited images due to the use of the decoder. More examples can be found at this anonymous page.

The motion segmentation mask is derived by applying an argmax operation on per-pixel assignment weights $\mathbf{w} \in \mathbb{R}_{+}^{H \times W \times M}$ along the motion token dimension M (see Eq. (4)). These visualizations demonstrate that STORM captures scene dynamics and groups 3D Gaussians that correspond to the same moving patterns, resulting in instance-level or motion-level segmentations. These unsupervised motion assignment masks enable us to select Gaussians based on their assignments for editing, without using ground truth 3D bounding boxes.

Editing, control, and inpainting. In Fig. 7, we demonstrate how Latent-STORM[3] enables edibility. Since each latent Gaussian *still* represents a *physical particle in space* with associated features, we can edit the scene by adding, removing, or modifying these Gaussians *before feature map rasterization and decoding*. For instance, to remove an object, we simply exclude its corresponding latent Gaussians from rendering. Using a lightweight decoder, Latent-STORM can reconstruct human movements (f, h), hallucinate occluded regions, and provide other advantages, such as image harmonization (f, g, h). For instance, Latent-STORM can recover leg movements and hand gestures. Furthermore, one common challenge in driving scene reconstruction is the difficulty of inpainting occluded regions or removing static objects without leaving black holes. Latent-STORM addresses this by synthesizing these areas, albeit with slight blurriness. Please refer to the anonymous page for more visualizations if interested. We believe incorporating diffusion models into this process could further improve generation quality, which we leave for future work. Overall, these results highlight the flexibility of STORM as a powerful tool for scene editing and animation, including the ability to hallucinate regions unseen in context frames.

5 Discussion and Conclusion

Conclusion. In this work, we have introduced STORM, a scalable spatio-temporal model designed to reconstruct dynamic scenes from sparse observations without requiring explicit motion supervision. Through extensive experiments, we have demonstrated STORM's ability to reconstruct dynamic outdoor scenes and estimate scene dynamics. Our method significantly surpasses existing per-scene optimization and feed-forward approaches, showcasing its versatility for a wide range of applications, including view synthesis, scene editing and point tracking. Looking ahead, we hope STORM will become a foundational model for various tasks across multiple domains, enabling more efficient and flexible approaches to 4D scene reconstruction, motion estimation, and beyond. As research in spatio-temporal modeling progresses, we believe STORM has the potential to unlock new possibilities for real-time dynamic scene analysis, interactive applications, and further advancements in self-supervised learning. We additionally discuss the limitations of our method in Appendix C.

[3]Our default STORM works well for vehicle editing, with slightly reduced performance for human modeling.

Ethics. STORM uses well-established public datasets Sun et al. (2020) that follow strict ethical guidelines in their data collection process. Should STORM be used to in-house collected data, we commit to blur and protect sensitive information. We will only train the model on data that is well calibrated for safety and privacy purpose. We strive to develop a safe and ethical spatial-temporal reconstruction model.

Reproducibility. At the core of STORM is a Vision-Transformer (Dosovitskiy, 2020)-based neural network with carefully designed output heads and supervisions. The Transformer backbone that we use is standard and can be found in many existing libraries such as `huggingface/timm`. For the remaining parts we provide all the implementation details including the network architectures as well as the hyperparameters in Section 4 and Appendix A for ease of re-implementation. We will release our full model and code, including STORM and Latent-STORM upon paper acceptance.

REFERENCES

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 130–141, 2023.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 2024.

Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint arXiv:2311.18561*, 2023.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, et al. Omnire: Omni urban scene reconstruction. *arXiv preprint arXiv:2408.16760*, 2024.

Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7911–7920, 2021.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, 2023.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4970–4980, 2023.

Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pp. 1–9, 2022.

Tobias Fischer, Jonas Kulhanek, Samuel Rota Bulò, Lorenzo Porzi, Marc Pollefeys, and Peter Kontschieder. Dynamic 3d gaussian fields for urban areas. *arXiv preprint arXiv:2406.03175*, 2024a.

Tobias Fischer, Lorenzo Porzi, Samuel Rota Bulo, Marc Pollefeys, and Peter Kontschieder. Multi-level neural scene graphs for dynamic urban environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21125–21135, 2024b.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12479–12488, 2023.

Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5712–5721, 2021.

Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024a.

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024b.

Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *arXiv preprint arXiv:2306.04988*, 2023.

Nan Huang, Xiaobao Wei, Wenzhao Zheng, Pengju An, Ming Lu, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Shanghang Zhang. s^3 gaussian: Self-supervised street gaussians for autonomous driving. *arXiv preprint arXiv:2405.20323*, 2024.

Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, and Konrad Schindler. Dynamic 3d scene analysis by point cloud accumulation. In *European Conference on Computer Vision*, pp. 674–690. Springer, 2022.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18365–18375, 2022.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023.

Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024.

Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021a.

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6498–6508, 2021b.

Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4273–4284, 2023.

Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5166–5175, 2024.

Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13–23, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

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.

Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *European Conference on Computer Vision*, pp. 424–443. Springer, 2022.

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2856–2865, 2021.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Julius Plucker. Xvii. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, (155):725–791, 1865.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327, 2021.

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10901–10911, 2021.

Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12932–12942, 2022.

Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4gm: Large 4d gaussian reconstruction model, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*, pp. 156–174. Springer, 2022.

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*, pp. 2446–2454, 2020.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *arXiv preprint arXiv:2406.04343*, 2024a.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10208–10217, 2024b.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.

Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. Neurad: Neural rendering for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14895–14904, 2024.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12959–12970, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Letian Wang, Seung Wook Kim, Jiawei Yang, Cunjun Yu, Boris Ivanovic, Steven L Waslander, Yue Wang, Sanja Fidler, Marco Pavone, and Peter Karkus. Distillnerf: Perceiving 3d scenes from single-glance images by distilling neural fields and foundation model features. *arXiv preprint arXiv:2406.12095*, 2024a.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2021.

Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. *arXiv preprint arXiv:2306.05422*, 2023.

Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. 2024b.

Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. *arXiv preprint arXiv:2403.16292*, 2024.

Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, et al. fvdb: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024.

Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024a.

Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21551–21561, 2024b.

Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *Advances in Neural Information Processing Systems*, 35:32653–32666, 2022.

Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, et al. Mars: An instance-aware, modular and realistic simulator for autonomous driving. In *CAAI International Conference on Artificial Intelligence*, pp. 3–15. Springer, 2023.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9421–9431, 2021.

Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023.

Yinghao Xu, Zifan Shi, Wang Yifan, Sida Peng, Ceyuan Yang, Yujun Shen, and Wetzstein Gordon. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arxiv: 2403.14621*, 2024.

Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024.

Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=ycv2z8TYur.

Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1389–1399, 2023a.

Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023b.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20331–20341, 2024b.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4578–4587, 2021.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *arXiv preprint arXiv:2404.19702*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21336–21345, 2024a.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21634–21643, 2024b.

A IMPLEMENTATION DETAILS

In this section, we discuss the implementation details of STORM. Our code and pre-trained models will be released. See our anonymous page for more results.

A.1 STORM IMPLEMENTATION DETAILS

Gaussian Parameterization. Our Transformer architecture and output mapping largely follow (Zhang et al., 2024). The input images are normalized to the range of [-1, 1]. Recall that each Gaussian is defined as $\mathbf{g} \equiv (\boldsymbol{\mu}, \mathbf{R}, \mathbf{s}, o, \mathbf{c})$, where $\boldsymbol{\mu} \in \mathbb{R}^3$ and $\mathbf{R} \in \mathbb{SO}(3)$ represent the center and orientation, $\mathbf{s} \in \mathbb{R}^3$ indicates the scale, $o \in \mathbb{R}^+$ denotes the opacity, and $\mathbf{c} \in \mathbb{R}^3$ corresponds to the color. Below, we describe the activations or normalizations applied to the raw outputs to derive these parameters.

For coordinates $\boldsymbol{\mu}$, we first compute ray origins and directions from the camera's intrinsic and extrinsic parameters. We then compute $\boldsymbol{\mu} = \text{ray}o + d \cdot \text{ray}dir$, where d is the 1-channel ray distance predicted by our model. Depth d is computed as $d = \text{near} + \sigma(d) * (\text{far} - \text{near})$, where σ represents the sigmoid function, and near and far are hyperparameters. In all our experiments, we set near to 0.1 and far to 400.0.

For rotation $\mathbf{R}$, we parameterize it with 4-dimensional quaternion vectors and apply L_2 normalization to ensure they are unit vectors.

For scale $\mathbf{s}$, we compute $\mathbf{s} = \min(\exp(\mathbf{s}' - 2.3), 0.5)$, following Zhang et al. (2024), where $\mathbf{s}'$ represents the outputs before normalization. This regularization limits the maximum size of Gaussians and improves training stability.

For opacity o, we compute $o = \sigma(o' - 2.0)$, again following Zhang et al. (2024).

For color $\mathbf{c}$, we do not apply any activation or normalization during training. However, at test time, we clamp the values to be within the range of -1 to 1. When computing PSNR, we map color back to [0, 1] range.

Sky MLP. The modulated sky MLP predicts sky color from view directions by conditioning a sky token $\mathbf{c}_{sky}$ through a modulated linear layer. Specifically, frequency-embedded viewing directional vectors $\gamma(\mathbf{d})$ are linearly projected to 64 dimensions, then normalized using LayerNorm without affine parameters. The sky token outputted from the Transformer serves as the conditioning vector $\mathbf{c}_{sky}$ (768 dims) is mapped to 64 dimensions and used in an adaptive layer normalization (AdaLN) process, where $\mathbf{c}_{sky}$ is transformed to produce shift and scale vectors, each of size 64, modulating the normalized features by $\mathbf{x} = \mathbf{x} \cdot (1 + \text{scale}) + \text{shift}$. Finally, the modulated features are linearly transformed to an output of 3-dimensional color.

Mask Decoder. The convolution layers in the mask decoder are similar to those used in SAM Kirillov et al. (2023), which is defined as:

```
self.output_upscaling = nn.Sequential(
  nn.ConvTranspose2d(embed_dim, 512, kernel_size=2, stride=2),
  LayerNorm2d(512),
  nn.GELU(),
  nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2),
  LayerNorm2d(256),
  nn.GELU(),
  nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2),
  nn.GELU()
)
```

The input to this decoder is the ViT output feature maps. After they are upsampled by the mask decoder, they are projected into a 32-dimensional space using a linear layer. Additionally, motion tokens are mapped into a 32-dimensional space using a set of three-layer MLPs, following the design in SAM.

Training. We train our model for 10,000 iterations with a global batch size of 64 on NVIDIA A100 GPUs, using a learning rate of 4×10^{-4}. The training process utilizes the AdamW optimizer (Loshchilov & Hutter, 2019) along with a cosine learning rate scheduler that includes a linear warmup phase over the first 5,000 iterations. We enable the LPIPS loss (Zhang et al., 2018) only after 5,000 iterations, as we find this approach stabilizes training. Gradient checkpointing is enabled by default to reduce memory usage. Behind the scene, we observe that `STORM` benefits from longer training durations and larger model sizes. We maintain the default setup to ensure alignment with our baseline in this work. However, an attractive direction for future work is to explore the scaling laws of `STORM` (Zhai et al., 2022).

Point trajectory estimation. We visualize the trajectories of dynamic Gaussians. These trajectories are obtained by chaining per-frame scene flows. Specifically, for each frame at t, we use the predicted scene flow to transform Gaussians to its next frame $t+1$ to obtain its estimated destination. Then, for every Gaussian at $t+1$, we find its nearest Gaussians transformed from t and connect them to visualize the trajectories. This process is recursively applied to all frames to obtain the final trajectories.

A.2 Loss function

We present more details about our loss function here. Given the rendered images $\hat{\mathbf{I}}$, depth maps $\hat{\mathbf{D}}$, opacity maps $\hat{\mathbf{O}}$, and velocities for all 3D Gaussians $\mathbf{v}$, along with the corresponding observed camera images $\mathbf{I}$, depth maps $\mathbf{D}$, and sky masks $\mathbf{M}$ that are predicted by a pre-trained segmentation model (Chen et al., 2022), we compute the overall loss as:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{sky}} + \mathcal{L}_{\text{reg}}, \tag{A1}$$

where the reconstruction loss combines RGB loss, depth loss, and LPIPS loss (Zhang et al., 2018):

$$\mathcal{L}_{\text{recon}} = \|\hat{\mathbf{I}} - \mathbf{I}\|_2 + \|(\hat{\mathbf{D}} - \mathbf{D})/\max(\mathbf{D})\|_1 + \lambda_{\text{lpips}} \cdot \text{LPIPS}\left(\hat{\mathbf{I}}, \mathbf{I}\right), \tag{A2}$$

and the sky loss and velocity regularization loss are MSE losses that encourage sparsity:

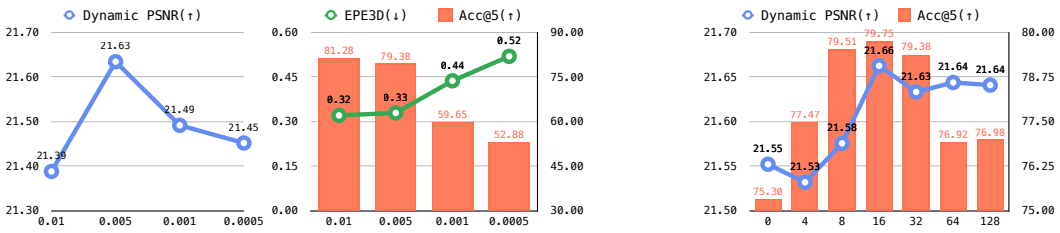
$$\mathcal{L}_{\text{sky}} = \lambda_{\text{sky}} \cdot \|\hat{\mathbf{O}} - (\mathbf{1} - \mathbf{M})\|_2, \quad \mathcal{L}_{\text{reg}} = \|\mathbf{v}\|_2. \tag{A3}$$

Here, the λ terms control the relative weighting of each loss component. For the LPIPS loss, we utilize a VGG-19-based (Simonyan & Zisserman, 2014) implementation. We set λ_{lpips} to 0.05, λ_{sky} to 0.1, and λ_{reg} to 10^{-3} in all experiments.

A.3 Baseline Implementations

For per-scene optimization 3DGS-based methods, we use the recently open-sourced codebase `DriveStudio` from Chen et al. (2024), which includes implementations for PVG Chen et al. (2023), DeformableGS (Yang et al., 2024b), and 3DGS (Kerbl et al., 2023) on the Waymo Open Dataset. For EmerNeRF (Yang et al., 2024a), we directly modify their officially released code. Since our task is to reconstruct short-sequenced dynamic scenes from sparse observations (3 cameras $\times$ 4 timesteps), the original training recipes designed for long-sequenced dense views (3 cameras $\times$ 200 timesteps) are no longer appropriate. Therefore, we reduce the training iterations for all methods from 20,000 to 5,000 and linearly scale down all iteration-based hyperparameters. In our preliminary experiments, we did not observe significant differences between training for 20,000 versus 5,000 iterations, as there are only limited training views available, while 5,000 iterations provide a much faster training time.

For generalizable approaches, LGM (Tang et al., 2024) has open-sourced their code and pre-trained models, whereas GS-LRM (Zhang et al., 2024) has not. However, LGM is originally trained on an object-centric synthetic dataset, which has a significant domain gap compared to our problem. Therefore, we followed their official code to reimplement their model within our codebase to eliminate potential misalignments due to differences in data processing, learning rate scheduling, supervision, and optimizers. For GS-LRM (Zhai et al., 2022), we implemented the model according to the descriptions provided in their paper. We train these models on the same dataset as ours with the same color, depth, perceptual and sky supervision, and the same number of iterations. Since these models do not inherently support sky processing, we modify them to predict the depth of the sky



(a) Effect of flow regularization loss coefficient λ_{reg}

(b) Effect of number of motion tokens M

Figure B.1: **Ablation study on velocity regularization and motion tokens. (a)** Effect of velocity regularization coefficient λ_{reg}: We evaluate rendering quality using dynamic PSNR and flow estimation performance using EPE3D and Acc_5. We find that excluding this regularization frequently leads to gradient explosion and NaN loss. **(b)** Impact of motion token count: We study how the number of motion tokens M influences rendering and motion estimation performance.

as the far plane, a predefined hyperparameter. This adjustment already enhances the performance of these methods. The number of trainable parameters of these models are controlled to be similar, *i.e.*, GS-LRM has 86.68M parameters, LGM has 103.29M parameters, while our default `STORM` has 100.60M parameters.

B Additional Results

B.1 Comparison on additional datasets

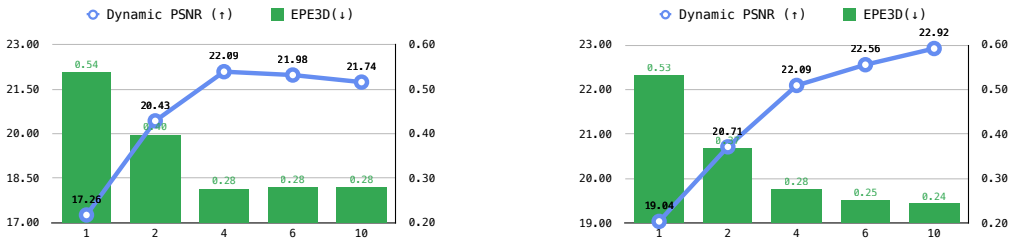
NuScenes. The NuScenes dataset (Caesar et al., 2020) contains 1000 driving scenes, each lasting 20 seconds, collected in Boston and Singapore, and captured at 12Hz frame rate. These scenes are divided into 700, 150, and 150 scenes for training, validation, and testing, respectively. Similar to our evaluation protocol for the Waymo Open Dataset (Sun et al., 2020), we use 3 frontal cameras at a roughly $5.5\times$ downsampled resolution (160×288) and leveraging both `sample` (key frames) and `sweep` data. Models are trained on the training set and evaluated on the validation set with *unchanged hyperparameters*. Results are presented in Table 2. Note that while performance on dynamic regions is not measured due to the additional process required to extract dynamic instances, we expect higher performance in these regions, as gains are driven by improvements in reconstructing dynamic content. Static reconstruction performance is expected to slightly surpass previous results since the artifacts caused by dynamic instances are addressed.

Argoverse2. The Argoverse2 dataset (Wilson et al.) contains 1,000 driving scenes, split into 700 for training, 150 for validation, and 150 for testing. It includes data from seven ring cameras, providing a 360-degree view. For training and evaluation, we use the three frontal cameras, resampled to a 192×256 resolution. The original central camera resolution is 2048×1550, while other cameras are 1550×2048, resulting in reversed aspect ratios. We do not apply special adjustments for this discrepancy and resize all images uniformly to 192×256. Results are presented in Table 2. To simplify processing, performance is measured on full images without extracting dynamic instances. Notably, depth RMSE is higher for this dataset compared to NuScenes and Waymo, which is expected due to Argoverse2's larger sensing range of over 200m, in contrast to the approximately 80m range of the other datasets.

B.2 Ablation Study

We study the effect of different components of our method here. All experiments here are conducted on the Waymo Open Dataset. To manage the computational demands of these extensive experiments, models studied in Fig. B.1 are trained with a global batch size of 32 for 100k iterations. For the models examined in Fig. B.2-(b), we use a global batch size of 64 over the same number of iterations.

Velocity regularization coefficient λ_{reg}. The impact of the velocity regularization coefficient is illustrated in Fig. B.1-(a). We observe that omitting this term often results in gradient explosion and NaN loss. Thus, this regularization term is indispensable. In this controlled setting, the optimal



(a) Zero-shot transfer to different number of testing input timesteps

(b) Effect of number of training input timesteps

Figure B.2: **Ablation study on input timesteps. (a)** Zero-shot transfer evaluation: We test STORM pre-trained with 4 input timesteps with varying test-time input timestep configurations without re-training. **(b)** Scaling training timesteps: We train STORM with different numbers of input views to assess its adaptability to changes in input timesteps.

coefficient is found to be 5×10^{-3}, as it achieves the best dynamic PSNR and ranks second in flow estimation performance.

Number of motion tokens M. We evaluate the effect of the number of motion tokens on rendering and flow estimation performance in Fig. B.2-(b). When no motion token ($M = 0$) is used, the dynamic mask decoder directly predicts pixel-aligned velocities from image embeddings, keeping the number of learnable parameters nearly constant to ensure fair comparison. As shown in Fig. B.2-(b), STORM demonstrates robust performance across different motion token counts, with the best results obtained at $M = 16$.

Number of input timesteps. Our default configuration trains and tests STORM with 4 input timesteps. Leveraging the sequence-to-sequence nature of our Transformer-based model, we can flexibly adjust the number of input timesteps during both training and testing by appending tokens from more input timesteps or dropping tokens from existing input timesteps. This flexibility enables us to study the effect of input sparsity on STORM's performance. We conduct two ablation studies in Fig. B.2. First, we test STORM trained with 4 input timesteps under varying test-time input timestep configurations without re-training. This zero-shot transfer experiment demonstrates that STORM generalizes well to unseen input configurations, though it achieves peak performance with 4 timesteps, as expected. In the second study, we re-train STORM with different numbers of input timesteps and evaluate their performance. Results indicate that increasing the number of input timesteps improves performance. Notably, when trained with a single timestep, STORM transitions into a future prediction framework. Even in this configuration, it significantly outperforms per-scene optimization approaches that utilize 4 input timesteps for reconstruction, achieving a 2 to 4 PSNR improvement on dynamic regions (*cf.* Table 1).

C Limitation

Limitations. While our model benefits from the scalability and flexibility of Transformer architectures, it comes with certain trade-offs. One limitation is the processed sequence length. STORM typically operates on images downsampled by a factor of 8, using inputs from three cameras and four timesteps, which results in around 7k tokens. Although we have fine-tuned STORM to handle up to 32,000 tokens in *preliminary* experiments to enable higher resolution images, longer temporal windows, or additional camera views, this comes with a non-trivial increase in computational costs for both training and inference. Another limitation is that our model requires camera intrinsic and extrinsic parameters as inputs. While these parameters are readily accessible in autonomous vehicle datasets, they may be more difficult to obtain in other domains, potentially limiting the ability to directly train and test STORM on those data domains without additional effort or preprocessing. Future works to address these limitations include better Transformer architecture with reduced complexity, joint optimization of camera parameters, and the use of geometric foundation models.