Towards Self-Supervised, Generalizable and Decomposable 4D Driving Scene Reconstruction

James Tu[1,2*] **Anqi Joyce Yang**[1,2*]
Jingkang Wang[†] **Siva Manivasagam**[1,2] **Raquel Urtasun**[1,2]
Waabi[1] University of Toronto[2]
{jtu, jyang, siva, urtasun}@waabi.ai

Abstract

Reconstructing high-fidelity and controllable digital twins from multi-modal sensory observations is a fundamental problem in physical AI applications. While existing per-scene optimization methods handle dynamic driving scenes well, they are slow to optimize, have artifacts at novel viewpoints, and require costly manual annotations for decomposing the scene into controllable instances. Self-supervised generalizable reconstruction methods enable faster and more scalable reconstructions by learning from large datasets, but existing approaches do not fully leverage multi-modal inputs (*e.g.*, camera and LiDAR) and lack decomposition capabilities. To address these limitations, we propose *STRIDE*, a self-supervised, generalizable and decomposable 4D driving scene reconstruction method. By processing multi-modal sensory inputs in a common 3D space with a Point-Transformer, *STRIDE* efficiently discovers spatio-temporal correspondences necessary for accurate flow prediction. By incorporating latent instance tokens, *STRIDE* is the first to perform feed-forward reconstruction and learnable decomposition without manual annotations. Experiments on public driving datasets show *STRIDE* achieves state-of-the-art performance on recovering geometry, appearance, and 3D flow. Moreover, we demonstrate how learned decompositions can enable dynamic instance manipulation and controllable simulation. Project website is at https://waabi.ai/stride/.

1 Introduction

Reconstructing dynamic 3D scenes from sensor data is a fundamental task in physical AI domains such as self-driving and robotics. It enables creating high-fidelity digital replicas of the physical world that support realistic and diverse simulation for validating and training autonomous systems. To be useful, a 4D reconstruction method should accurately capture scene geometry and dynamics, effectively integrate multi-modal sensor data (*e.g.*, camera and LiDAR), generate and render scene representations efficiently, and produce decomposable representations for downstream editing.

Recently, 3D Gaussian Splatting (3DGS) [19] has emerged as an effective representation, offering real-time rendering and high-quality novel view synthesis by modeling scenes as a set of learnable Gaussian primitives. Building on 3DGS, existing methods [62, 20, 16] have enabled reconstruction of dynamic driving scenes by decomposing them into a static background and a set of dynamic actors using human annotations such as 3D tracklets or segmentation masks. While achieving impressive results, these methods optimize a separate set of Gaussian primitives for each individual scene, which is time-consuming. As a result, they tend to memorize the input views rather than learning the underlying scene structure, leading to artifacts under novel viewpoints. Moreover, their reliance on costly annotations for separating static and dynamic elements restricts scaling to unlabeled data.

*Equal contribution. † Work done at Waabi.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

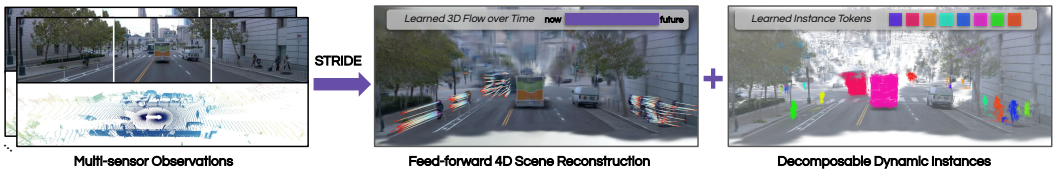


Figure 1: *STRIDE* performs *self-supervised*, *generalizable* and *decomposable* 4D driving scene reconstruction.

To address these limitations, *generalizable* reconstruction [7, 44, 28, 52, 45] is an alternative paradigm where a neural network is trained across a large dataset of dynamic scenes to predict Gaussian primitives directly from raw sensor inputs. This approach produces robust representations under viewpoint extrapolation [8, 45] and imperfect calibration [23, 55]. It also enables efficient reconstruction, since inference typically is only a single network pass. By leveraging *self-supervised* learning, recent works [45, 52] bypass the need for dense 4D annotations, enabling scaling to unlabeled data. However, existing methods remain limited in the sensor modalities they can exploit. Camera-only approaches such as STORM [52] and DrivingRecon [28] do not leverage input LiDAR data, and reason in 2D image space where perspective distortion complicates geometry and motion modeling. Conversely, the LiDAR-centric Flux4D [45] reasons in 3D to achieve strong geometric fidelity, but the inherent sparsity of LiDAR-dependent Gaussian initialization leaves distant regions under-represented. Additionally, existing methods are limited by their architectural designs. Convolutional backbones [45, 28] suffer from limited receptive fields that cannot capture object motion across sparse frames. Conversely, fully-connected transformers [52] perform inefficient global attention for scene reasoning.

Beyond sensor modality and architectural design, achieving *decomposable* scene representations remains a major challenge for 4D reconstruction of dynamic scenes. Many downstream applications, such as scene editing and controllable simulation, require the dynamic scene to be parsed into distinct object instances rather than an unstructured set of Gaussian primitives [33]. Recent self-supervised methods [29, 38, 25] have shown that instance-level decomposition can emerge without explicit labels, but they rely on initializations from pre-trained 2D instance segmentation models and per-scene optimization with scene-specific regularization [38, 25], or on post-processing heuristics (*e.g.*, DBSCAN clustering on LiDAR points [29]) that require careful parameter tuning during inference time, limiting their scalability to diverse real-world scenes.

To address these challenges, we propose *STRIDE*, a self-supervised, generalizable and decomposable multi-modal 4D spatio-temporal reconstruction model capable of performing precise reconstruction and dynamic instance decomposition in a single forward pass. *STRIDE* leverages sparse LiDAR to accurately anchor dense multi-image observations in a shared 3D space, creating a unified spatio-temporal representation which is then processed by a Point Transformer [48] to efficiently aggregate local contextual cues and predict accurate scene geometry and flow. Importantly, by associating each dynamic Gaussian primitive with a latent instance token, *STRIDE* generates a decomposable 4D scene. Extensive experiments on the Waymo Open Dataset [39] and PandaSet [49] demonstrate that *STRIDE* achieves state-of-the-art performance on 4D scene reconstruction with multi-modal inputs, outperforming existing approaches in scene flow estimation, image fidelity, and geometric accuracy, while also learning dynamic instance decompositions for scene editing. Overall, this work proposes a novel multi-modal reconstruction architecture and introduces learnable instance decomposition.

2 Related Work

Per-scene Reconstruction Differentiable rendering approaches such as NeRF [31] and 3DGS [19] were first introduced for high-quality static scene reconstruction and later extended to dynamic driving scenes. A common paradigm decomposes the scene into a static background and a set of dynamic actors using human annotations (*e.g.*, 3D bounding boxes or semantic masks) [33, 54, 50, 62, 20, 16, 10, 42]. While these methods achieve impressive visual fidelity, they optimize a separate representation for each scene, which limits generalization to novel viewpoints and incurs high computational cost. Moreover, their reliance on 3D annotations increases cost and limits scalability to large unlabeled datasets. To eliminate annotation dependence, self-supervised per-scene methods leverage motion cues, cycle consistency, and physics-informed priors [53, 9, 17, 34, 47, 40] to decompose static and dynamic elements directly from data. However, they typically require complex multi-stage training such as separately estimating geometry and motion [34, 40] or pre-trained vision

models [32, 59] to provide pseudo labels, making them sensitive to hyperparameters and difficult to scale.

Generalizable Reconstruction To overcome the lack of efficiency and generalization bottlenecks of per-scene optimization, generalizable reconstruction methods train networks across many scenes to predict scene representations directly from observations [5, 46, 4]. In the driving domain, several feed-forward approaches have been proposed to handle dynamic urban scenes [41, 30, 6], but they still rely on 3D tracklets for dynamic actor handling, limiting their scalability to large unlabeled datasets. To eliminate this annotation dependence, self-supervised approaches predict 3D Gaussians and their velocities without explicit labels [52, 28, 45, 58]. However, these methods suffer from limitations in both sensor modality and architectural design. STORM [52] and DrivingRecon [28] are camera-centric and cannot leverage LiDAR, a ubiquitous sensor in self-driving that provides accurate depth measurements. They also reason densely in 2D image space rather than in the native 3D space of Gaussian primitives, limiting their ability to recover accurate 3D geometry and scene flow. Conversely, Flux4D [45] decodes Gaussians directly from LiDAR points, achieving strong geometry but struggling in regions (*e.g.*, distant buildings) with sparse LiDAR coverage. Our approach addresses these limitations by reasoning in 3D via Point Transformers [48] over image pixels lifted with learned dense depth without requiring any annotations.

Instance Decomposition for Driving Scenes Beyond reconstruction quality, applications such as scene editing and simulation require a *decomposable* representation where Gaussian primitives are parsed into distinct object instances typically via 3D annotations [54, 30]. Most recently, self-supervised methods [38, 25] demonstrate that instance-level decomposition can emerge without explicit 3D cuboid labels. Specifically, DIAL-GS [38] leverages a pre-trained 2D instance segmentation model [14] and 2D tracking [1] to initialize instances, and then exploits appearance-position inconsistency to identify dynamic instances and assign instance-aware 4D Gaussians. IDSplat [25] uses zero-shot video tracking anchored by LiDAR with a coordinated-turn smoothing scheme to obtain instance trajectories. UniRe [29] adopts DBSCAN clustering on LiDAR points. However, all these approaches rely on pre-trained instance models as initialization and per-scene optimization with scene-specific regularization, or on post-processing heuristics which require careful parameter tuning during inference time, limiting their scalability to diverse real-world scenes. In contrast, although our method uses DBSCAN as initialization, it learns to refine the imperfect initialization during training through a set of learnable instance tokens that are jointly optimized with reconstruction, and it does not require any test-time optimization or hyperparameter tuning at inference time.

Image and Point Cloud Processing Architectures Convolutional Neural Networks (CNNs) [22] were the first widely adopted architectures for 2D image processing [18, 15, 37, 21], due to their parameter efficiency, local inductive bias, and translation invariance. These networks were subsequently extended to 3D to operate on voxelized point clouds [51, 57], often enhanced by sparse convolution operations [26]. However, CNNs struggle to capture long-range dependencies due to their limited receptive fields. More recently, Vision Transformers (ViTs) [12], which process a tokenized image input via self-attention, have emerged as a more scalable alternative for 2D images. Standard ViTs suffer from a critical computational bottleneck due to the quadratic cost of modeling pairwise interactions between all tokens. Efficient variants [27, 11] were later introduced, restricting attention operations to token subsets. However, these architectures rely on the image's underlying grid structures, making them unsuitable for unstructured 3D points. To bridge this gap, recent methods [48, 60] introduce structure to point clouds via serialization, allowing more efficient self-attention operations within serialized chunks. Our method adopts PTv3 [48] to efficiently process spatio-temporal observations.

3 High-Fidelity Decomposable 4D Scene Representations with *STRIDE*

Given a sequence of posed camera images and LiDAR point clouds, we aim to reconstruct the underlying 4D scene in a feed-forward manner. This representation should capture scene geometry, appearance, motion, and dynamic instance decompositions. To this end, we propose *STRIDE*, a multi-modal transformer-based architecture that predicts Gaussian primitives with associated flow and dynamic instance assignments. *STRIDE* consists of three key stages, illustrated by Fig. 2. First, a multi-modal feature extraction stage encodes LiDAR and camera inputs into a unified spatio-temporal feature representation. Next, a 4D scene reasoning stage processes this representation to decode a set of Gaussian primitives. Finally, a learnable decomposition stage associates dynamic Gaussian

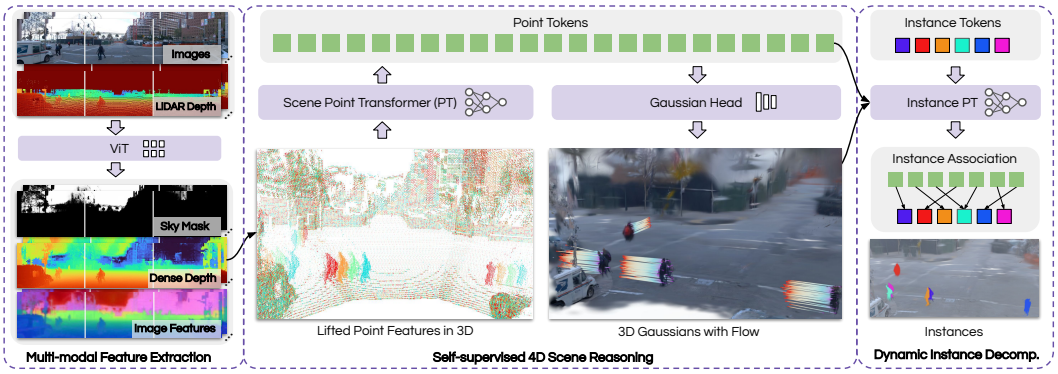


Figure 2: Overview of *STRIDE*. Given multi-view images and LiDAR depths over T source timestamps, *STRIDE* predicts dense depths, sky masks and image features with a vision transformer. We then lift and aggregate foreground non-sky image features across frames (colors represent times), and use a 3D point transformer to output foreground 3DGS. A parallel sky point transformer (not shown here in the figure) processes the sky region. The dynamic instance decomposition stage then learns to associate each dynamic 3DGS primitive to an instance token.

primitives to a set of latent tokens, creating instances based on token assignments. We now detail the problem formulation, architecture of each stage, and self-supervised training objectives.

3.1 Problem Formulation

Our model takes as input multi-modal sensory observations from a modern self-driving platform, consisting of 3D LiDAR point clouds $\mathbf{P}^{(t)} \in \mathbb{R}^{N_t \times 3}$ and multi-view camera images $\mathbf{I}^{(t,v)} \in \mathbb{R}^{H \times W \times 3}$ captured across timesteps t and views v, along with calibration parameters to transform between the 3D space and each 2D image space. Following [52, 45], our model outputs a set of 3D Gaussians $\{\mathbf{g}_i\}$ to model the underlying scene. Each Gaussian $\mathbf{g}_i$ is parameterized by its centroid position ($\mu_i \in \mathbb{R}^3$), scale ($\sigma_i \in \mathbb{R}^3$), orientation quaternion ($q_i \in \mathbb{R}^4$), color ($z_i \in \mathbb{R}^3$), opacity ($\alpha_i \in \mathbb{R}$) and a velocity vector ($\mathbf{v}_i \in \mathbb{R}^3$). Our model additionally learns to decompose the scene into a static component and a set of dynamic instances. Specifically, Gaussians in the scene are partitioned into either the static set $\mathcal{G}_{\text{static}}$ or one of M dynamic instance sets $\{\mathcal{G}^m_{\text{dyn}}\}_{m=1}^{M}$.

3.2 Multi-Modal Feature Extraction with Vision Transformers

To reconstruct high-fidelity 4D scene representations, it is crucial to fully leverage image observations for their dense appearance and semantic cues. However, lifting them to 3D requires accurate per-pixel depth. On the other hand, LiDAR provides accurate but sparse geometric measurements. To exploit both, our approach performs LiDAR-guided depth estimation to physically anchor the visual features in 3D. For each camera image $\mathbf{I}^{(t,v)}$, we project the LiDAR points $\mathbf{P}^{(t)}$ onto the image to derive a sparse LiDAR depth map $\mathbf{D}^{(t,v)}_{\text{lidar}} \in \mathbb{R}^{H \times W}$. We additionally define a LiDAR depth availability mask $\mathbf{M}^{(t,v)}_{\text{lidar}} \in \{0,1\}^{H \times W}$ to explicitly distinguish pixels without a LiDAR depth measurement. We concatenate $(\mathbf{I}^{(t,v)}, \mathbf{D}^{(t,v)}_{\text{lidar}}, \mathbf{M}^{(t,v)}_{\text{lidar}})$ to form a 5-channel input, which we then process using a vision transformer (ViT) encoder [12]. The ViT first encodes the image into a sequence of positionally encoded tokens. The image tokens are subsequently processed with multiple self-attention layers [43], and decoded by parallel multi-layer perceptron (MLP) layers to obtain per-pixel depth estimation $\mathbf{D}^{(t,v)} \in \mathbb{R}^{H \times W}$, sky mask $\mathbf{S}^{(t,v)} \in [0,1]^{H \times W}$ and feature maps $\mathbf{F}^{(t,v)} \in \mathbb{R}^{H \times W \times C}$. We use the predicted sky mask to decouple the sky and foreground entities.

3.3 4D Scene Reasoning with Point Transformers

Foreground Point Feature Construction We now lift foreground image features into a unified 3D feature representation. Reasoning explicitly in 3D is essential for accurately predicting 3D scene flow, as it naturally resolves 2D perspective distortions and operates on metric distances. Specifically, we back-project each foreground pixel feature in $\mathbf{F}^{(t,v)}$ using the predicted depth and known calibration parameters to obtain a 3D point feature, which is subsequently augmented with embeddings of 3D

xyz coordinates, camera time t, view v, and pixel ray directions [35]. We aggregate these 3D point features across all views v and times t to obtain a unified 3D feature point cloud $\mathcal{F}_{\text{fg}}$.

Point-Transformer Backbone We adopt a point transformer backbone [48] to predict scene appearance, geometry, and flow from $\mathcal{F}_{\text{fg}}$. Prior feed-forward reconstruction works predominantly rely on sparse 3D convolution backbones [45, 36], which suffer from limited receptive fields due to their local convolution kernels, and thus struggle to capture long-range correspondences necessary for accurate flow estimation (*e.g.* on high-speed actors). In contrast, point transformers efficiently capture non-local dependencies via multi-scale patch-based attention. *STRIDE* treats foreground 3D features $\mathcal{F}_{\text{fg}}$ as point tokens and processes them with a point transformer. The refined point features $\mathcal{F}'_{\text{fg}}$ are fed into MLP decoders to predict Gaussian parameters for each 3D foreground point.

Sky Modeling While foreground pixels can be assigned a specific 3D position, the depths of sky pixels are typically not well-defined or significantly farther away. Therefore, we model the sky with a sky dome, following [45]. Specifically, we initialize a template Fibonacci point cloud sphere with a pre-defined radius of 400 meters, and use the learned sky mask to filter out points that do not project onto the sky regions of any input image. Each point's feature is initialized as its 3D xyz coordinates, rgb values via image projection, and a 3D scale based on the point's average distance to its three nearest neighbors. This constructs a $N_{\text{sky}} \times 9$ input. We adopt a separate lightweight point transformer to process the initial sky points and decode the parameters for N_{sky} sky Gaussians.

3.4 Dynamic Object Decomposition

In this stage, we organize the foreground Gaussian primitives into a more structured object-centric representation. Crucially, Gaussians moving together belong to the same underlying instance. Therefore, dynamic instance decomposition can emerge from the same motion cues that self-supervise scene flow. To this end, this stage introduces a set of latent dynamic instance tokens, where each token represents a single moving entity in the scene. Each Gaussian primitive is then associated to a single instance token, creating dynamic instance decompositions based on token associations.

Initialization To efficiently decompose dynamic instances, we first isolate the dynamic Gaussians. For each foreground Gaussian $\mathbf{g}_{\text{fg},i}$ with an associated velocity $\mathbf{v}_i$, we apply a velocity threshold τ and obtain a set of N_{dyn} dynamic point features $\mathcal{F}_{\text{dyn}} = \{\mathbf{f}'_{\text{fg},i} \mid \|\mathbf{v}_i\|_2 > \tau\}$. Next, we initialize a set of spatial instance tokens by first aligning the dynamic Gaussians to a common reference time and then clustering them based on their spatial proximity via DBSCAN [3], yielding a set of cluster centroids $\{\mathbf{c}_j\}_{j=1}^{M}$. From each cluster centroid, we initialize an instance token with the centroid's 3D position and a learnable token embedding, yielding an initial set of M instance tokens $\{\mathbf{t}_j\}_{j=1}^{M}$.

Token-Point Association We associate dynamic Gaussians to tokens via a differentiable similarity matrix. To compute this, we concatenate the spatial point features and instance tokens, and process them jointly through another Point Transformer responsible for decomposition. The output is an updated set of point features $\{\mathbf{f}'_{\text{dyn},i}\}_{i=1}^{N_{\text{dyn}}}$ and token features $\{\mathbf{t}_j\}_{j=1}^{M}$. Next, we compute the feature similarity between Gaussian i and token j:

$$d_{\text{feat}}(i,j) = 0.5 \cdot \left(\frac{\mathrm{H}_p(\mathbf{f}'_{\text{dyn},i})^\top \mathrm{H}_t(\mathbf{t}_j)}{\|\mathrm{H}_p(\mathbf{f}'_{\text{dyn},i})\| \|\mathrm{H}_t(\mathbf{t}_j)\|} + 1 \right), \tag{1}$$

where H_p and H_t are MLPs projecting the point and token features into a shared latent space, and the normalization ensures the similarity score is in the range [0, 1]. In addition to feature-based similarity, we inject a spatial bias term, as intuitively, an instance token should not govern faraway Gaussians. To achieve this, we compute a spatial alignment term between per-Gaussian centroid μ_i and token centroid $\mathbf{c}_j$: $d_{\text{spatial}}(i,j) = \|\mu_i - \mathbf{c}_j\|_2$. Finally, we also predict an existence term $\zeta_j = \text{Sigmoid}(\text{MLP}(\mathbf{t}_j))$ for each token, allowing our model to directly suppress redundant tokens. The final similarity score is computed as $s_{i,j} = \zeta_j \Big(d_{\text{feat}}(i,j) - \min(1, \lambda_{\text{spatial}} d_{\text{spatial}}(i,j)) \Big)$, where clamping is applied to handle instances with large spatial extent. Based on these similarity scores, we construct an explicit decomposition of the dynamic Gaussian primitives: $\mathcal{G}^m_{\text{dyn}} = \{\mathbf{g}_i \mid \arg\max_j s_{i,j} = m\}$.

3.5 Self-Supervised Training Objectives

Our model is trained with self-supervised training objectives in two stages. In the first stage, we train the multi-modal feature extraction and 4D scene reasoning modules to decode accurate geometry, appearance and motion of the scene. In the second stage, we train the dynamic instance decomposition module while keeping the previous modules frozen.

Reconstruction Supervision Following previous work [52], we use the predicted velocities to move all Gaussians with a linear motion model to render at a target time step. We employ standard photometric reconstruction losses $\mathcal{L}_{\text{photo}}$ and a velocity regularization term $\mathcal{L}_{\text{v-reg}} = \sum_i^N \|\mathbf{v}_i\|_1$. The photometric loss is a combination of L2 rendered RGB loss, perceptual loss [61], and L1 rendered depth loss. Following [52], we also employ an opacity loss $\mathcal{L}_{\text{opacity}}$ to encourage zero rendered opacity in the sky region. In addition, we incorporate an L1 loss $\mathcal{L}_{\text{mde}}$ to supervise $\mathbf{D}^{(t,v)}$ against sparse $\mathbf{D}^{(t,v)}_{\text{lidar}}$, as well as a binary cross-entropy sky mask loss $\mathcal{L}_{\text{sky}} = \sum_{t,v} \text{BCE}(\mathbf{S}^{(t,v)}, \mathbf{S}^{*(t,v)})$, supervised by off-the-shelf sky segmentation masks $\mathbf{S}^{*(t,v)}$ following [52]. The final loss term in this stage is a weighted sum $\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{photo}} + \lambda_{\text{v-reg}}\mathcal{L}_{\text{v-reg}} + \lambda_{\text{opacity}}\mathcal{L}_{\text{opacity}} + \lambda_{\text{mde}}\mathcal{L}_{\text{mde}} + \lambda_{\text{sky}}\mathcal{L}_{\text{sky}}$.

Decomposition Supervision To supervise the assignment similarity $s_{i,j}$ without human-annotated instance labels, we leverage the intuition that Gaussians belonging to the same dynamic instance should move together. Therefore, motion cues from $\mathcal{L}_{\text{recon}}$ should naturally provide a supervision signal for learning dynamic instance decomposition as well. To achieve this, we simply need each Gaussian's velocity to be a differentiable function of its instance assignment. Thus, for each instance token j with feature $\mathbf{t}_j$, we use an MLP to decode a velocity $\mathbf{v}_{\text{tok},j} \in \mathbb{R}^3$. Then, for each dynamic Gaussian i, we assign an instance-dependent velocity

$$\mathbf{u}_i = \sum_j w_{i,j}\mathbf{v}_{\text{tok},j} + \tanh(\text{MLP}(\mathbf{f}'_{\text{dyn},i})), \tag{2}$$

where the first term decodes velocity from token associations with $\mathbf{w}_i = \text{one_hot}(\arg\max(\mathbf{s}_i))$, and the second term introduces a per-Gaussian residual velocity to capture non-rigid deformations. We constrain the residual velocity with a $\tanh$ activation such that $\mathbf{u}_i$ should be primarily dominated by the token velocity. By replacing the original per-Gaussian velocity $\mathbf{v}_i$ with the token-based per-point velocity $\mathbf{u}_i$, the reconstruction loss $\mathcal{L}_{\text{recon}}$ supervises both the token velocities and assignments, naturally clustering Gaussians that move together into the same instance. Since $w_{i,j}$ is a discrete argmax assignment, we adopt the Straight-Through Estimator (STE) [2] in the backward pass to enable our instance learning formulation.

The reconstruction objective alone, however, may not be sufficient to discourage over-decomposition, where excess token initialization may lead to multiple tokens governing different parts of the same instance. To mitigate this, we introduce a spatial smoothness prior that encourages spatially close Gaussians with similar velocities to share the same token assignment. For each dynamic Gaussian i, we first find the most popular instance assignment among its spatial and kinematic neighbors:

$$\hat{y}_i = \texttt{Mode}\left(\left\{\arg\max(\mathbf{s}_k) \;\middle|\; \|\mu_i - \mu_k\|_2 < \tau_{\text{dist}}, \|\mathbf{v}_i - \mathbf{v}_k\|_2 < \tau_{\text{vel}}\right\}\right), \tag{3}$$

where τ_{dist} and τ_{vel} are pre-defined thresholds. We then align the point's soft assignment distribution with this local consensus via a cross-entropy loss: $\mathcal{L}_{\text{smooth}} = \sum_i \text{CE}(\text{softmax}(\mathbf{s}_i), \text{one_hot}(\hat{y}_i))$. By penalizing deviations from the local majority, this loss suppresses over-decomposition and encourages kinematically cohesive regions to collapse onto a single instance token. The overall loss term in this stage is $\mathcal{L}_{\text{decomp}} = \mathcal{L}_{\text{photo}} + \lambda_{\text{v-reg}}\mathcal{L}_{\text{v-reg}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}$.

4 Experiments

We now evaluate *STRIDE* in terms of 4D scene reconstruction and dynamic instance decomposition. First, *STRIDE* improves 3D flow estimation and dynamic actor reconstruction, primarily through leveraging multi-modal inputs with a point-transformer backbone. Second, our learnable decomposition module can further improve over a strong clustering-based baseline [3]. We detail experimental setup, baselines, and results in this section. Please refer to the appendix for more details.

Method	Model Input		Dynamic			Full Image†			Scene Flow			
	Camera	LiDAR	PSNR↑	SSIM↑	D-RMSE↓	PSNR↑	SSIM↑	D-RMSE↓	EPE3D↓	Acc 5↑	Acc 10↑	Angle↓
STORM [52]	✓		22.10	0.624	7.50	26.38	0.794	5.48	0.276	81.12	85.61	0.658
STORM+* [52]	✓	✓	22.57	0.661	4.20	**27.34**	**0.830**	2.22	0.200	86.27	89.39	0.485
Flux4D-ff* [45]	✓	✓	20.62	0.548	8.71	23.16	0.707	4.98	0.246	89.85	91.62	0.436
Ours	✓	✓	**23.16**	**0.698**	**3.89**	27.21	0.829	**2.14**	**0.161**	**91.32**	**93.05**	**0.307**

Table 1: [WOD] 4D Reconstruction results. * denotes our re-implementation. † denotes non-sky region.

Split	Method	Dynamic				Full Image†			
		PSNR↑	SSIM↑	D-MAE↓	V-RMSE↓	PSNR↑	SSIM↑	D-MAE↓	V-RMSE↓
Interp.	STORM* [52]	17.65	0.367	5.24	0.203	20.79	0.508	4.80	0.238
	STORM+* [52]	18.39	0.382	2.59	0.195	21.82	0.520	1.38	0.204
	Flux4D-ff [45]	18.62	**0.454**	2.06	0.181	20.88	0.575	1.37	0.211
	Ours	**18.92**	0.444	**1.54**	**0.131**	**22.39**	**0.583**	**0.91**	**0.154**
Extrap.	STORM* [52]	16.61	0.338	5.82	0.218	19.63	0.489	5.19	0.251
	STORM+* [52]	17.72	0.361	3.23	0.192	21.02	0.505	1.69	0.215
	Flux4D-ff [45]	17.88	0.403	2.62	0.171	20.04	**0.540**	1.65	0.199
	Ours	**18.65**	**0.414**	**1.74**	**0.134**	**21.45**	0.529	**1.09**	**0.154**

Table 2: [PandaSet] 4D Reconstruction results on interpolated frames and extrapolated frames. STORM* results are from [45]. STORM+* is our multi-modal re-implementation. † denotes non-sky region.

4.1 Self-Supervised and Generalizable 4D Scene Reconstruction

Experimental Setup We evaluate 4D reconstruction on two real-world driving datasets, Waymo Open Dataset [39] (WOD) and PandaSet [49]. On WOD, we follow STORM's sparse-input setup and metrics [52], where we train and evaluate on 2-second clips consisting of 20 frames, with frame 0, 5, 10, 15 as input and evaluation done on the remaining frames. On PandaSet, we strictly follow Flux4D's dense-input setup and metrics [45], where we use 1.5-second clips, with frame 0, 2, 4, 6, 8, 10 as input and evaluate on frames 1, 3, 5, 7, 9 for interpolation and frames 11-15 for extrapolation. We report PSNR, SSIM, depth and velocity errors for full images and dynamic regions. Please refer to the appendix for more details.

Baselines We compare with self-supervised generalizable methods STORM [52] and Flux4D [45]. Since STORM only uses LiDAR for supervision, we implement a multi-modal variant *STORM+* that takes LiDAR depth maps and availability masks as input. For Flux4D, we use only its feed-forward stage, denoted *Flux4D-ff*, for a fair comparison to feed-forward methods.

4D Reconstruction Improvements Results on WOD (Table 1) show that our method produces significantly better scene flow as indicated by the EPE3D and flow angle metrics, enabling better reconstruction quality on dynamic regions. While STORM+ has marginally better full image PSNR and SSIM, it is significantly worse in dynamic regions and flow estimation, indicating a lower quality scene representation. As Flux4D-ff relies solely on LiDAR for Gaussian initialization, it has weaker performance in the sparse view setting. On the denser-input PandaSet setting, *STRIDE* again achieves



Figure 3: Qualitative comparisons of rendered RGB and flow images against SoTA. Our method has better image quality in dynamic regions, and produces more accurate flows, especially for the pedestrians on the right.

Method	Precision @ IoU				Recall @ IoU			
	Mean↑	0.1↑	0.3↑	0.5↑	Mean↑	0.1↑	0.3↑	0.5↑
DBSCAN [3]	51.03	58.27	52.29	42.54	71.43	82.62	72.30	59.39
Mask R-CNN [14]	63.17	71.62	65.23	52.66	71.88	76.36	73.55	65.74
Mask R-CNN [14] + Tracker [1]	47.25	55.32	48.72	37.71	73.11	83.73	76.19	59.42
DIAL-GS Dynamic Instance Labels [38]	33.93	38.38	34.59	28.81	23.88	26.69	24.47	20.48
Ours (Learned)	**53.40**	**62.03**	**55.33**	**42.84**	**74.23**	**85.33**	**77.23**	**60.13**

Table 3: [WOD] 2D Dynamic instance segmentation metrics.

Method	Precision @ IoU				Recall @ IoU			
	Mean↑	0.1↑	0.3↑	0.5↑	Mean↑	0.1↑	0.3↑	0.5↑
DBSCAN [3]	44.23	58.35	45.14	29.19	52.25	69.28	52.78	**34.70**
Mask R-CNN [14]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mask R-CNN [14] + Tracker [1]	43.33	58.97	43.86	27.18	52.86	70.50	54.34	33.73
DIAL-GS Dynamic Instance Labels [38]	33.54	43.34	35.13	22.15	16.39	20.57	17.32	11.30
Ours (Learned)	**46.98**	**63.64**	**47.87**	**29.41**	**53.43**	**71.15**	**54.74**	34.41

Table 4: [WOD] 3D Dynamic instance segmentation metrics.

the best overall performance on both interpolation and extrapolation views (Table 2). Fig. 3 shows a qualitative example, where we have better image quality and more accurate flow estimation, especially for the pedestrians on the right.

4.2 Dynamic Instance Decomposition

Setup A new capability enabled by our method is learnable dynamic instance decomposition. We evaluate the per-frame dynamic instance decomposition quality in the same WOD setting, where our model takes frame 0, 5, 10, 15 as input, and predicts 3DGS with velocities and instance IDs. We adopt both 2D and 3D instance segmentation metrics on rendered 2D instance masks and 3D instance segmentations respectively. Similar to the flow evaluation [52], we evaluate on input context frames.

Baselines As we are the first to propose the dynamic instance decomposition task in the self-supervised, generalizable 4D reconstruction setting, we use the clustering-based initialization introduced in Section 3.4 as a strong baseline. This also directly ablates the learning component of our instance decomposition module. Unlike token initialization, which only uses cluster centroids, the baseline directly uses the clustering outputs as the final instance decomposition. We additionally design a few more baselines based on the per-scene instance-aware method DIAL-GS [38]. We run the pre-trained 2D instance segmentation model Mask R-CNN [14] and a 2D tracker [1] to derive per-pixel instance IDs, and then associate each 3DGS reconstructed by STRIDE with its source pixel's instance ID. On top of this, we evaluate DIAL-GS's dynamic instance labels, which employ a dynamic instance filtering heuristic after applying Mask R-CNN and tracking. As a reference, we also directly evaluate the Mask R-CNN instances in 2D, which, unlike our method and other baselines which perform instance segmentation in 4D, only provides per-frame 2D instances in the image space and cannot be evaluated with 3D metrics.

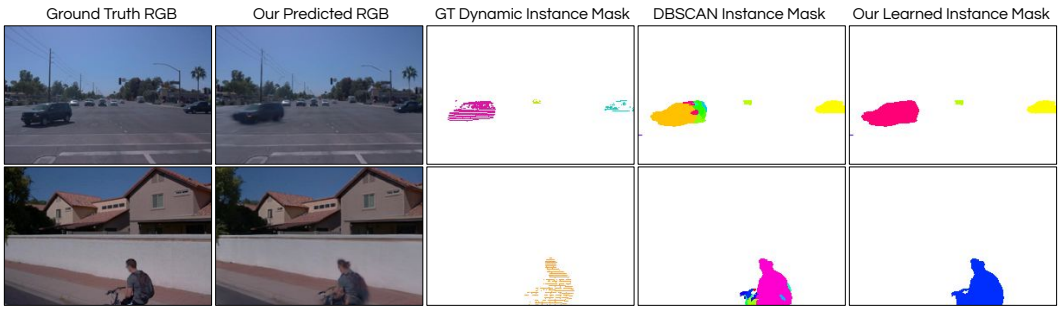


Figure 4: Qualitative comparisons of rendered dynamic instance masks. While DBSCAN over-decomposes on fast-moving and irregularly shaped actors, our method outputs a coherent mask for each instance.

2D Backbone	3D Backbone	Dynamic			Full Image†			Flow			
		PSNR↑	SSIM↑	D-RMSE↓	PSNR↑	SSIM↑	D-RMSE↓	EPE3D↓	Acc 5↑	Acc 10↑	Angle↓
Cross-time ViT	N/A	22.08	0.629	4.46	**26.84**	0.816	2.40	0.207	87.59	89.98	0.593
Per-image ViT	SparseConvUNet	21.01	0.572	5.46	25.85	0.794	2.60	0.385	87.38	88.70	0.750
Per-image ViT	PTv3	**22.67**	**0.667**	**4.17**	26.78	**0.818**	**2.20**	**0.180**	**91.39**	**91.84**	**0.382**

Table 5: [WOD] Backbone ablations. † denotes non-sky region.

Image	LiDAR Depths	LiDAR Points	Dynamic			Full Image†			Flow			
			PSNR↑	SSIM↑	D-RMSE↓	PSNR↑	SSIM↑	D-RMSE↓	EPE3D↓	Acc 5↑	Acc 10↑	Angle↓
✓			21.45	0.587	7.96	25.56	0.770	5.80	0.313	85.81	87.96	0.742
✓	✓		**22.67**	**0.667**	4.17	26.78	0.818	2.20	**0.180**	91.39	91.84	**0.382**
✓	✓	✓	22.65	**0.667**	**4.15**	**26.97**	**0.821**	**2.19**	**0.180**	**91.67**	**93.03**	0.389

Table 6: [WOD] Input modality ablations. † denotes non-sky region.

Metrics We use the standard IoU-based precision and recall metrics for 2D and 3D dynamic instance segmentation evaluation. For 2D metrics, we construct the ground-truth 2D dynamic instance masks using ground-filtered LiDAR points inside human-annotated 3D bounding boxes of dynamic objects. We match the predicted instance masks with the ground-truth instance masks via Hungarian matching, and then compute precision and recall at different IoU thresholds. We report the mean precision and recall over IoU thresholds 0.1, 0.3, 0.5. For 3D instance segmentation metrics, we similarly assign each LiDAR point a ground-truth instance ID if it is inside a labelled cuboid, and not part of the ground annotations. We then assign each LiDAR point a predicted instance ID based on the ID of the 3DGS with the highest opacity at the queried LiDAR point. Each ID then corresponds to a point set for us to measure 3D point-based IoU and derive per-frame instance metrics. The metrics treat all static ground-truth actors as ignores, so the Mask R-CNN-related baselines are not penalized for including static instances. Please refer to the appendix for more details.

Results As shown in Table 3 and Table 4, our learnable instance module consistently outperforms all 4D instance segmentation baselines in almost all precision and recall metrics for both 2D and 3D. Since the DBSCAN baseline is identical to the instance token initialization, this comparison shows that improvements come purely from training with self-supervised 4D reconstruction signals and the smoothness prior. Fig. 4 shows a qualitative example, where our method produces more coherent instance masks for the cyclist and cars. This confirms our intuition that using the same motion cues that supervise scene flow helps improve dynamic instance decomposition. In addition, Mask R-CNN + tracker has limited occlusion reasoning and tracking in 2D. DIAL-GS's view warping-based dynamic filtering heuristic aggressively filtered out many dynamic actors, severely hurting both precision and recall as the number of true positive matches decreased. With our learned instances, *STRIDE* enables controllable scene manipulation, such as actor removal, injection, and placement as seen in Fig. 5.

4.3 Ablation

We perform ablations on backbone design, the multi-modal setting, and instance learning. For efficiency, all ablations are on WOD and use a reduced training schedule (0.5× steps and batch size).

Backbone Ablations To justify our ViT and PTv3-based backbone design, we ablate with the backbones used in STORM [52] and Flux4D [45]. We replace our entire backbone with STORM's cross-time ViT backbone that directly reasons over 2D image-view tokens. Alternatively, we replace our PTv3 with the 3D sparse-convolution UNet used in Flux4D, while keeping our multi-modal ViT feature extractor. As shown in Table 5, our method outperforms both backbones in almost all metrics. Cross-time ViT is slightly better on full image PSNR, but is noticeably worse on dynamic PSNR and flow angles. SparseConvUNet performs worse than PTv3 for reasoning on sparse multi-modal inputs.

Input Modality Ablations Table 6 ablates our use of multi-modal inputs. We start with the image-only setting, and then add the LiDAR depth and availability maps as input, which is our main formulation. The third row further shows the results where we additionally feed image-feature-



Figure 5: Applications of *STRIDE*: Our instance decomposition enables scene editing such as actor removal, manipulation, and injection, enabling creation of new scenarios for autonomy development.

decorated LiDAR points as input to the PTv3-based backbone. The image-only model result highlights the importance of LiDAR inputs for better 4D reconstruction, while the additional LiDAR point input gives a marginal performance gain at a higher computational cost.

Instance Learning Ablations Table 7 ablates our instance learning module. M_1 is the DBSCAN baseline without any instance learning. M_2 aims to learn only with the smoothness loss, which collapses to a degenerate under-decomposing solution that assigns all spatially close 3DGS to the same instance, resulting in high precision but low recall. M_3 learns only with the 4D reconstruction loss through the instance token velocities, and it achieves a small gain over the initialization, suggesting that the 4D reconstruction-based self-supervision signal is helpful, but M_5 shows that it is the best when combined with the smoothness loss. Finally, M_4 initializes the instance tokens with the simple Farthest Point Sampling heuristic [13] instead of DBSCAN, indicating the importance of initialization.

Method	Init	$\mathcal{L}_{\text{smooth}}$	$\mathcal{L}_{\text{recon}}$	mPrec ↑	mRec↑
M_1	✓			51.03	71.43
M_2	✓	✓		+17.77	-19.42
M_3	✓		✓	+0.83	+1.34
M_4		✓	✓	-23.50	-3.18
M_5	✓	✓	✓	+2.37	+2.80

Table 7: Instance ablations. M_2–M_5 show Δ w.r.t. M_1.

5 Limitations

STRIDE achieves significant improvements on dynamic scene reconstruction and pioneers a learnable approach to obtain decomposable scene representations. However, several limitations remain: (1) complex (*e.g.* non-linear) object motions cannot be modeled by our constant-velocity motion model; (2) our decomposition relies on motion cues, meaning it cannot decompose static objects; (3) our backbone operates on all input observations aggregated across time, preventing seamless long-horizon reconstruction due to memory constraints. These remain as exciting future directions to address.

6 Conclusion

We present *STRIDE*, a self-supervised, generalizable model for reconstructing large-scale driving scenes from multi-modal sensory inputs. By processing a unified spatio-temporal representation with 3D Point Transformers, our framework achieves superior reconstruction quality and scene flow estimation, highlighting the importance of our 3D architectural design. Furthermore, we pioneer a self-supervised approach to learn decomposable scene reconstructions in the feed-forward setting, demonstrating that standard reconstruction signals can improve dynamic instance segmentation. Overall, this work provides important architectural insights for 4D reconstruction, and establishes a foundational step towards learnable object-centric scene decomposition.

Acknowledgments and Disclosure of Funding

We sincerely thank our anonymous reviewers for their thoughtful reviews and feedback. We also thank Yun Chen and Jiawen Chen for helpful discussions in the early stage of this paper, and the rest of the Waabi team for their valuable assistance and support.

References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[3] Ricardo JGB Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):5, 2015.

[4] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.

[5] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

[6] Xiaoxue Chen, Ziyi Xiong, Yuantao Chen, Gen Li, Nan Wang, Hongcheng Luo, Long Chen, Haiyang Sun, Bing Wang, Guang Chen, Hongyang Li, Ya-Qin Zhang, Hangjun Ye, and Hao Zhao. Dggt: Feedforward 4d reconstruction of dynamic driving scenes using unposed images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1276, June 2026.

[7] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, 2024.

[8] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3R: Gradient guided generalizable reconstruction. In *ECCV*, 2024.

[9] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *Int. J. Comput. Vision*, 134(3), January 2026.

[10] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. Omnire: Omni urban scene reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12124–12134, 2022.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[16] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11982–11992, June 2025.

[17] Nan Huang, Xiaobao Wei, Wenzhao Zheng, Pengju An, Ming Lu, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Shanghang Zhang. S3gaussian: Self-supervised street gaussians for autonomous driving. *arXiv preprint arXiv:2405.20323*, 2024.

[18] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. In *TOG*, 2023.

[20] Mustafa Khan, Hamidreza Fazlali, Dhruv Sharma, Tongtong Cao, Dongfeng Bai, Yuan Ren, and Bingbing Liu. Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8315–8321, 2025.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[22] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[23] Samuel Li, Pujith Kachana, Prajwal Chidananda, Saurabh Nair, Yasutaka Furukawa, and Matthew Brown. Rig3r: Rig-aware conditioning and discovery for 3d reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[24] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. In *Advances in Neural Information Processing Systems*, 2021.

[25] Carl Lindström, Mahan Rafidashti, Maryam Fatemi, Lars Hammarstrand, Martin R. Oswald, and Lennart Svensson. Idsplat: Instance-decomposed 3d gaussian splatting for driving scenes. *arXiv preprint arXiv:2511.19235*, 2025.

[26] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–814, 2015.

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[28] Hao Lu, Tianshuo Xu, Wenzhao Zheng, Yunpeng Zhang, Wei Zhan, Dalong Du, Masayoshi Tomizuka, Kurt Keutzer, and Ying-Cong Chen. Drivingrecon: Large 4d gaussian reconstruction model for autonomous driving. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] Yunxuan Mao, Rong Xiong, Yue Wang, and Yiyi Liao. Unire: Unsupervised instance decomposition for dynamic urban scene reconstruction. *arXiv preprint arXiv:2504.00763*, 2025.

[30] Sheng Miao, Sijin Li, Pan Wang, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Evolsplat4d: Efficient volume-based gaussian splatting for 4d urban scene synthesis. *arXiv preprint arXiv:2601.15951*, 2026.

[31] 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. In *ECCV*, 2020.

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. Featured Certification.

[33] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs. In *CVPR*, 2021.

[34] Chensheng Peng, Chengwei Zhang, Yixiao Wang, Chenfeng Xu, Yichen Xie, Wenzhao Zheng, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. Desire-gs: 4d street gaussians for static-dynamic decomposition and surface reconstruction for urban driving scenes. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 6782–6791, June 2025.

[35] Julius Plücker. XVII. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, 155:53–76, 1865.

[36] Chen Shi, Shaoshuai Shi, Xiaoyang Lyu, Chunyang Liu, Kehua Sheng, Bo Zhang, and Li Jiang. Unisplat: Unified spatio-temporal fusion via 3d latent scaffolds for dynamic driving scene reconstruction. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

[38] Chenpeng Su, Wenhua Wu, Chensheng Peng, Tianchen Deng, Zhe Liu, and Hesheng Wang. Dial-gs: Dynamic instance aware reconstruction for label-free street scenes with 4d gaussian splatting. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13962–13969, 2026.

[39] 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. *CVPR*, 2020.

[40] Su Sun, Cheng Zhao, Zhuoyang Sun, Yingjie Victor Chen, and Mei Chen. Splatflow: Self-supervised dynamic gaussian splatting in neural motion flow field for autonomous driving. In *CVPR*, 2025.

[41] Qijian Tian, Xin Tan, Yuan Xie, and Lizhuang Ma. Drivingforward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input. In *AAAI*, 2025.

[42] Haithem Turki, Qi Wu, Xin Kang, Janick Martinez Esturo, Shengyu Huang, Ruilong Li, Zan Gojcic, and Riccardo de Lutio. Simuli: Real-time lidar and camera simulation with unscented transforms. In *International Conference on Learning Representations (ICLR)*, 2026.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000—-6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[44] 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.

[45] Jingkang Wang, Henry Che, Yun Chen, Ze Yang, Lily Goli, Sivabalan Manivasagam, and Raquel Urtasun. Flux4d: Flow-based unsupervised 4d reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[46] 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 *CVPR*, 2021.

[47] Xiaobao Wei, Qingpo Wuwu, Zhongyu Zhao, Zhuangzhe Wu, Nan Huang, Ming Lu, Ningning Ma, and Shanghang Zhang. Emd: Explicit motion modeling for high-quality street gaussian splatting. In *ICCV*, 2025.

[48] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024.

[49] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *ITSC*, 2021.

[50] 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. In *ECCV*, 2024.

[51] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[52] Jiawei Yang, Jiahui Huang, Yuxiao Chen, Yan Wang, Boyi Li, Yurong You, Maximilian Igl, Apoorva Sharma, Peter Karkus, Danfei Xu, Boris Ivanovic, Yue Wang, and Marco Pavone. Storm: Spatio-temporal reconstruction model for large-scale outdoor scenes. In *ICLR*, 2025.

[53] 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 *ICLR*, 2024.

[54] 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 *CVPR*, 2023.

[55] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2026.

[56] 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. *JMLR*, 2025.

[57] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[58] Zhongrui Yu, Zhao Wang, Yijia Xie, Yida Wang, Xueyang Zhang, Yifei Zhan, and Kun Zhan. StreetForward: Perceiving dynamic street with feedforward causal attention. In *ECCV*, 2026.

[59] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2D Feature Representations by 3D-Aware Fine-Tuning. In *ECCV*, 2024.

[60] Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler. LitePT: Lighter Yet Stronger Point Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[61] 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.

[62] 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 *CVPR*, 2024.

Appendix

In the appendix, we first provide more method details and implementation details. Then, we provide additional details about the experimental setups, metrics, and additional evaluation results that were not included in the main paper due to space limit. Finally, we provide more qualitative results.

A Method Details

A.1 4D Reconstruction Stage

We hereby provide more details about the 4D reconstruction backbone. Our multi-modal feature extractor ViT first applies a downsampling 2D convolution with stride size p to the input image, which gives us a feature map of size $\frac{H}{p} \times \frac{W}{p} \times C$. We then add learnable positional encodings based on the image patch locations. We adopt a ViT-S with feature dimension 384 and 6 attention heads, and apply 12 layers of self-attention to the input features. We use $p = 4$ for WOD and $p = 8$ for PandaSet. Our parallel MLP decoders consist of two layers with the GELU activation function. Each MLP decodes $p \times p$ values from each image token, which get reshaped to the input image resolution for depth maps, sky masks and image features. We additionally apply the Sigmoid function on top of the sky logits to obtain the soft binary sky masks. During the feature lifting process, we use the ground-truth sky masks during training, and we apply a threshold of 0.5 to binarize the predicted sky masks during inference.

For all point transformers, we use the standard implementation of PTv3 [48] (`https://github.com/pointcept/pointtransformerv3`). For the 3D point transformer on lifted foreground pixels, we set the grid size to 0.1m, feature dimensions to 128, stride size to 2, and use 4 encoder stages and 3 decoder stages with patch size 128. For the sky point transformer, we set the grid size to 0.1m, feature dimensions to 64, stride size to 2, and use 3 encoder stages and 2 decoder stages with patch size 128.

For the sky dome, we initialize a Fibonacci sphere with radius 400m and 600^2 samples. We project the template points into the images, and take the rgb color from the earliest image that has a valid projection in the sky region defined by the sky mask.

For the foreground Gaussian MLP decoders, we use a linear layer to directly output Gaussian parameters for each point feature. For the sky Gaussians, the sky PTv3 directly outputs the Gaussian parameters in the last layer.

A full overview of the method is illustrated in Figure 6, which contains the sky branch as well.

During rendering, we combine all foreground Gaussians and sky Gaussians together to render RGB color maps. We only use the foreground Gaussians to render depths.

A.2 Dynamic Instance Decomposition Stage

We set the velocity filtering threshold to $\tau = 1.0$. We use a lightweight instance point transformer with grid size 4cm, 128 feature dimensions, 256 patch size, and 8 PTv3 blocks without downsampling or upsampling. We cap the max number of instance tokens to 200, using farthest point sampling to subsample. We set λ_{spatial} to 0.1. For DBSCAN we adopt the hierarchical DBSCAN implementation [3], and we set min_samples to 5.

A.3 Self-Supervised Training Objectives

Dynamic Instance Decomposition Supervision For instance-based velocity assignment, we use a weight $w_{i,j}$ to indicate which token velocity each Gaussian i should take. A linear combination based on similarity scores $s_{i,j}$ would yield a soft assignment over instance tokens, while a direct argmax would not be differentiable with respect to the full similarity matrix. Therefore, we use a Straight-Through Estimator (STE) [2] over the softmax of similarity scores. Let $\mathbf{s}_i \in \mathbb{R}^M$ be the

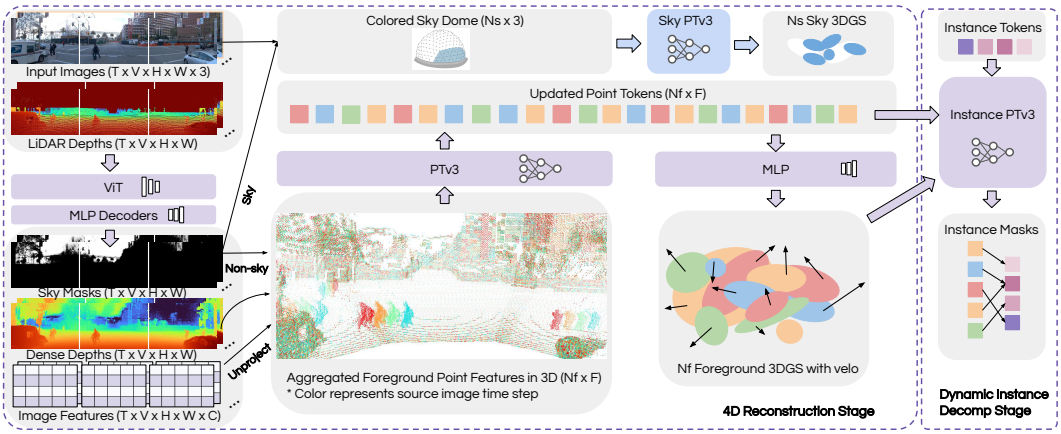


Figure 6: Overview of *STRIDE*. In the 4D reconstruction stage, given multi-view images and LiDAR depths over T source timestamps, we use a vision transformer to predict dense depths, sky masks and image features. We then lift and aggregate foreground non-sky image features from all frames (colors represent input image times), and use a 3D point transformer to output foreground 3DGS. A parallel sky point transformer processes the sky region. The dynamic instance decomposition stage then learns to associate each dynamic 3DGS primitive to an instance token.

vector of logits for Gaussian i. The discrete assignment vector $\mathbf{w}_i$ is formulated as:

$$\mathbf{w}_i^{\text{soft}} = \text{softmax}(\mathbf{s}_i) \tag{4}$$

$$\mathbf{w}_i^{\text{hard}} = \text{one_hot}(\arg\max(\mathbf{s}_i)) \tag{5}$$

$$\mathbf{w}_i = \mathbf{w}_i^{\text{hard}} + \mathbf{w}_i^{\text{soft}} - \text{sg}(\mathbf{w}_i^{\text{soft}}) \tag{6}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. As a result, during the forward pass, each Gaussian i would only take the velocity of the instance token with the highest similarity score, while during the backward pass, the gradients would be back-propagated to all instance tokens based on the soft assignment weights.

Hyperparameters In practice, we set $\lambda_{\text{v-reg}} = 0.005$, $\lambda_{\text{opacity}} = 0.1$, $\lambda_{\text{sky}} = 0.02$, $\lambda_{\text{mde}} = 0.02$. For the smoothness loss, we set $\lambda_{\text{smooth}} = 0.0001$, and the thresholds $\tau_{\text{dist}} = 0.3$ and $\tau_{\text{vel}} = 3.0$. We use the same hyperparameters for both WOD and PandaSet.

Training Details On WOD, following STORM [52], we train with 8x downsampled images with the same training schedule of effective batch size 64 for 100k iterations. We use the AdamW optimizer with a learning rate of 1e-4 and a weight decay of 5e-3. We apply learning rate warm up for the first 2k iterations and a cosine learning rate schedule with the minimum learning rate of 1e-5. On PandaSet, we train with a batch size of 32 for 40k iterations. Since Flux4D evaluates on the full resolution images, we take 4x downsampled images with 270×480 as input resolution and render the Gaussians at the full 1080×1920 resolution for supervision.

B Experiment Details

B.1 Experimental Setup Details

Our evaluation of 4D scene reconstruction quality focuses on the *self-supervised* and *generalizable* setting, comparing against SoTA methods STORM [52] and Flux4D [45]. STORM mainly conducts experiments on the Waymo Open Dataset [39] (WOD) with sparse input frames, while Flux4D primarily evaluates on dense input frames on PandaSet [49]. To showcase that *STRIDE* is robust to both input frame densities, we evaluate with STORM's setting on WOD and Flux4D's setting on PandaSet.

The Waymo Open Dataset [39] contains 1000 20-second sequences: 798 for training and 202 for validation. We strictly follow STORM for the experimental setup and metrics. We use the 10Hz 360°

LiDAR and three frontal cameras at an 8× downsampled resolution (160×240). We conduct training and evaluation on 2-second clips, which consist of 20 frames. For each clip, we use frame 0, 5, 10, and 15 as input frames, and evaluate on remaining frames with PSNR, SSIM and Depth RMSE as metrics for both full images and dynamic regions. To evaluate 3D flow, we report End-Point-Error (EPE) in 3D (EPE3D), Acc_5, Acc_{10}, and flow angle errors [24, 52] on the input context frames.

PandaSet [49] contains 103 8-second sequences and strictly following Flux4D, we use 93 sequences for training and 10 for validation. We use the front camera and the 360° LiDAR. We conduct training and evaluation on 1.5-second clips where the input frames are frame 0, 2, 4, 6, 8, 10 and evaluation is on frame 1, 3, 5, 7, 9 for interpolated novel views and frame 11-15 for extrapolated novel views. Different from STORM's setting, here the input frames are closer in time and the evaluation is performed with the full resolution (1080×1920) images. Following [45], we report PSNR, SSIM, Depth MAE and Velocity RMSE for both full images and dynamic regions.

B.2 Dynamic Instance Decomposition Metrics

We construct the ground-truth 2D dynamic instance masks as follows: (1) for each image under evaluation, we first project all associated LiDAR points at the same timestamp to the image, which gives us a LiDAR-point availability mask $\mathbf{M}_{avail} \in \{0,1\}^{H\times W}$; (2) we then take the LiDAR points inside all object bounding boxes, and project them to the image to create a set of instance masks $\mathbf{M}_{inst} \in \{0,1,\dots,N\}^{H\times W}$, where N is the number of instances in the image, and 0 represents not associated to a dynamic instance. Note that since not all objects are dynamic, we further define which ground-truth objects should be ignored. If a ground-truth instance has very small velocity (with velocity magnitude under 1m/s), we mark it as ignore. If a ground-truth instance has too few valid pixels (≤ 16) in the 2D instance mask, we also mark it as ignore.

For our predicted dynamic Gaussians with instance IDs, we render the 2D predicted instance mask as follows. For each Gaussian, we construct a feature vector $\in \mathbb{R}^M$, where M is the total number of predicted dynamic instances. If the Gaussian is static, we assign this feature vector to all zeros. Otherwise, we assign the feature vector with a one-hot encoding based on its instance ID. We then render the Gaussian primitives with these feature vectors to the image plane, which gives us a rendered instance mask $\mathbf{M}_{pred} \in \mathbb{R}^{H\times W\times M}$. We take the argmax over the feature dimension to get the final predicted instance mask $\mathbf{M}_{pred} \in \{0,1,\dots,M\}^{H\times W}$, where 0 represents static Gaussians that are not associated to any dynamic instance. If the max of the rendered feature vector is under a certain threshold (0.1), we also mark the pixel as static with instance ID 0.

We evaluate on all pixels that are marked as 1 in the LiDAR availability mask. In other words, for each predicted instance k, we only consider the pixels where $\mathbf{M}_{avail}(i,j) = 1 \wedge \mathbf{M}_{pred}(i,j) = k$. For each ground-truth instance, as the instance mask is created from LiDAR points directly, using the $\mathbf{M}_{inst}$ alone is sufficient to find all valid pixels associated to each instance.

We conduct two rounds of matching between ground-truth instances and predicted instances. In the first round of matching, we consider ground-truth instances that are not marked as ignore. For each non-ignore ground-truth instance i and predicted instance j, we compute the intersection-over-union (IoU) overlap of their available pixels in the image. We then use the Hungarian algorithm to find the optimal matching between ground-truth and predicted instances based on the IoU scores, and we consider a predicted instance as a true positive if its IoU with the matched ground-truth instance is over a threshold (we consider thresholds 0.1, 0.3, 0.5). In the second round of matching, we consider all valid ground-truth actors that are marked as ignore and match them with the remaining unmatched predicted instances using the same Hungarian matching process. The number of true positives P comes from the first round of matching. The number of matched ignores comes from the second round of matching, which we denote as Q. The precision is then computed as $\frac{P}{M-Q}$, where M is the total number of predicted instances. The recall is computed as $\frac{P}{N-Q'}$, where N is the total number of ground-truth instances and Q' is the total number of ground-truth ignore instances.

Following the flow metrics in STORM [52], we evaluate the instance metrics on all input frames. We report the precision and recall under different IoU thresholds (0.1, 0.3, 0.5) and report the average precision and average recall across all thresholds. We report the metrics only on frames with at least one valid ground-truth instance, and we report the average metrics across all valid frames.

For 3D segmentation metrics, the process is similar except how we construct ground-truth 3D segmentation labels and predicted 3D instance masks. Since we only use three front cameras, we only consider LiDAR points that are visible in these camera frustums, by projecting them to each camera and only consider points with projected depth > 0.5m in at least one of the cameras. We assign each LiDAR point a ground-truth instance ID only if it is both inside a labelled cuboid and not part of the ground. We set the point to ignore if the ground-truth instance is static. For each LiDAR point, we then assign its predicted instance ID by considering 10 nearest 3DGS primitives, and for each 3DGS, we compute the opacity at the query LiDAR point with the full anisotropic density $\text{density}_i(\mathbf{x}) = \text{opacity}_i \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right)$. We then assign the point the instance ID of the max-density Gaussian if the density is above a certain threshold (0.05), and compute the 3D point-based IoU for evaluation.

B.3 Model Size

We compare the model size and inference speed of our method with STORM [52] and Flux4D [45]. For model size, we compare the number of parameters in the 4D reconstruction model. STORM has 100.6M training parameters, Flux4D has 4.29M training parameters, and ours has 41.13M training parameters.

B.4 Inference Speed and Memory Scaling

Our PTv3 model leverages local patch-based attention, which offers improved runtime efficiency compared to fully-connected attention transformers, employed by STORM. We benchmark the inference speed of a single forward pass in the sparse-input WOD setting on a single NVIDIA A6000 GPU with a batch size of 1. Note the forward pass ends at 3DGS generation, not including rendering. We measure runtime and memory across different numbers of input context frames. The results are shown in Table 8. Here we observe that inference speed scales quadratically with the number of input context frames in STORM. On the other hand, the scaling is linear for our PTv3 backbone. Memory scaling is linear for both, and higher in STRIDE by design, as it is able to maintain more fine-grained activations throughout the network to yield superior reconstruction quality.

	STORM		STRIDE	
# Ctx	Inf.(ms)	Mem.(GB)	Inf.(ms)	Mem.(GB)
1	42	0.7	142	1.4
4	313	2.0	486	4.9
8	1125	3.7	953	9.2
12	2454	4.9	1430	13.5

Table 8: Runtime and memory comparisons.

B.5 Backbone Ablations Setting

Since STORM's backbone has 100M parameters while our model has 41M parameters, in the backbone ablation for fair comparison we reduced the ViT channel size from 768 to 512, and number of attention heads from 12 to 8, resulting in 43M model parameters. We additionally modified the original Flux4D backbone to increase the depth and feature dimensions of the SparseConvUNet model, resulting in 42M model parameters. For all 4D reconstruction ablation experiments, we halve both the batch size and the training iterations for efficiency. We keep the rest of the training settings (e.g., learning rate, optimizer, and loss weights) the same as the main *STRIDE* model.

B.6 Instance Learning Ablations

Method	Init	$\mathcal{L}_{\text{smooth}}$	$\mathcal{L}_{\text{recon}}$	Precision @ IoU				Recall @ IoU			
				Mean↑	0.1↑	0.3↑	0.5↑	Mean↑	0.1↑	0.3↑	0.5↑
M_1	✓			51.03	58.27	52.29	42.54	71.43	82.62	72.30	59.39
M_2	✓	✓		68.80	86.00	72.16	48.24	52.01	60.81	54.42	40.81
M_3	✓		✓	51.86	60.49	53.74	41.36	72.77	85.24	75.49	57.57
M_4		✓	✓	27.53	34.90	29.84	17.84	68.25	86.67	76.48	41.59
M_5	✓	✓	✓	53.40	62.03	55.33	42.84	74.23	85.33	77.23	60.13

Table 9: [WOD] Ablations on different components of the learned instance decomposition module.

Due to space constraints in the main paper, we only report the mean precision and recall of different rows. Table 9 shows the full results including precision and recall under different IoU thresholds, which draw the same conclusions as the main paper.

B.7 DBSCAN Initialization Ablations

DBSCAN provides the initial proposals used to initialize the instance tokens, which are subsequently refined by our learnable decomposition module. To evaluate sensitivity to this initialization, we conducted additional experiments by varying the DBSCAN parameters to produce different levels of decomposition on our WOD evaluation set. The results are shown in Table 10. At the recall-optimal DBSCAN setting (row 2), STRIDE consistently improves both precision and recall. The improvement is much larger under the more fragmented initialization (row 1), where mPrecision increases from 33.3 to 70.4 while mRecall also improves from 60.5 to 62.2, showing that instance learning effectively suppresses redundant tokens and consolidates over-segmented proposals. Row 3 is an extreme under-decomposed setting with only three median clusters, where DBSCAN recall has already dropped from 71.4 to 58.7. Although STRIDE is not designed to split merged proposals, it remains stable and yields slight improvements over DBSCAN in both precision and recall in this challenging setting. Overall, STRIDE is robust across variations in initialization and is particularly effective at refining over-decomposed proposals.

	DBSCAN		STRIDE	
# Cluster	mPrec	mRecall	mPrec	mRecall
40	33.3	60.5	70.4	62.2
10	51.0	71.4	53.4	74.2
3	69.6	58.7	69.8	59.0

Table 10: Ablation on DBSCAN initialization.

B.8 Instance Learning Hyperparameter Ablations

We ablated several key hyperparameters used during instance learning, specifically the distance threshold τ_{dist} and the velocity threshold τ_{vel} used in the smoothness loss, the spatial weight λ_{spatial} used to anchor the similarity scores, and the weight λ_{smooth} for the smoothness loss. Table 11 shows the results when we vary each of these hyperparameters, and the last row reflects the default setting used in our main experiments. The table shows the method is generally not very sensitive to τ_{dist} and τ_{vel}, as with varying values they still outperform the DBSCAN initialization. The ablation shows that spatial weight is important, as setting it to zero leads to a significant drop in both precision and recall, while increasing it to 1.0 has similar performance as the default 0.1. In addition, varying smoothness loss weight shows a similar trend as the ablation study in Table 9. With a lower smoothness loss weight (1e-5), supervision is more dominated by the reconstruction loss; the performance drops compared to the default setting but is still better than using reconstruction loss alone (M_3 in Table 9). On the other hand, increasing the smoothness loss weight to 1e-3 gives more power to the smoothness loss, leading to higher precision but lower recall, which matches well with M_2 in Table 9.

τ_{dist}	τ_{vel}	λ_{spatial}	λ_{smooth}	mPrec	mRecall
0.5	3.0	0.1	1e-4	53.3	73.6
1.0	3.0	0.1	1e-4	52.9	74.0
0.3	1.0	0.1	1e-4	54.1	73.7
0.3	5.0	0.1	1e-4	53.6	73.4
0.3	3.0	0.0	1e-4	36.1	58.4
0.3	3.0	1.0	1e-4	52.7	74.4
0.3	3.0	0.1	1e-5	52.9	73.0
0.3	3.0	0.1	1e-3	63.2	64.0
0.3	3.0	0.1	1e-4	53.4	74.2

Table 11: Ablation on instance hyperparameters.

C Additional Qualitative Results

4D Reconstruction We provide additional qualitative comparisons of the rendered RGB and flow images against SoTA methods in Fig. 7 and Fig. 8.

Dynamic Instance Decomposition We provide additional qualitative examples of the instance decomposition results in Fig. 9. We additionally randomly select a few qualitative examples and present them in Fig. 10, which show general success and failure cases of our method.

Scene Editing We provide additional qualitative examples of the scene editing results in Fig. 11. With our dynamic instance decomposition capability, we can easily remove objects from the scene,

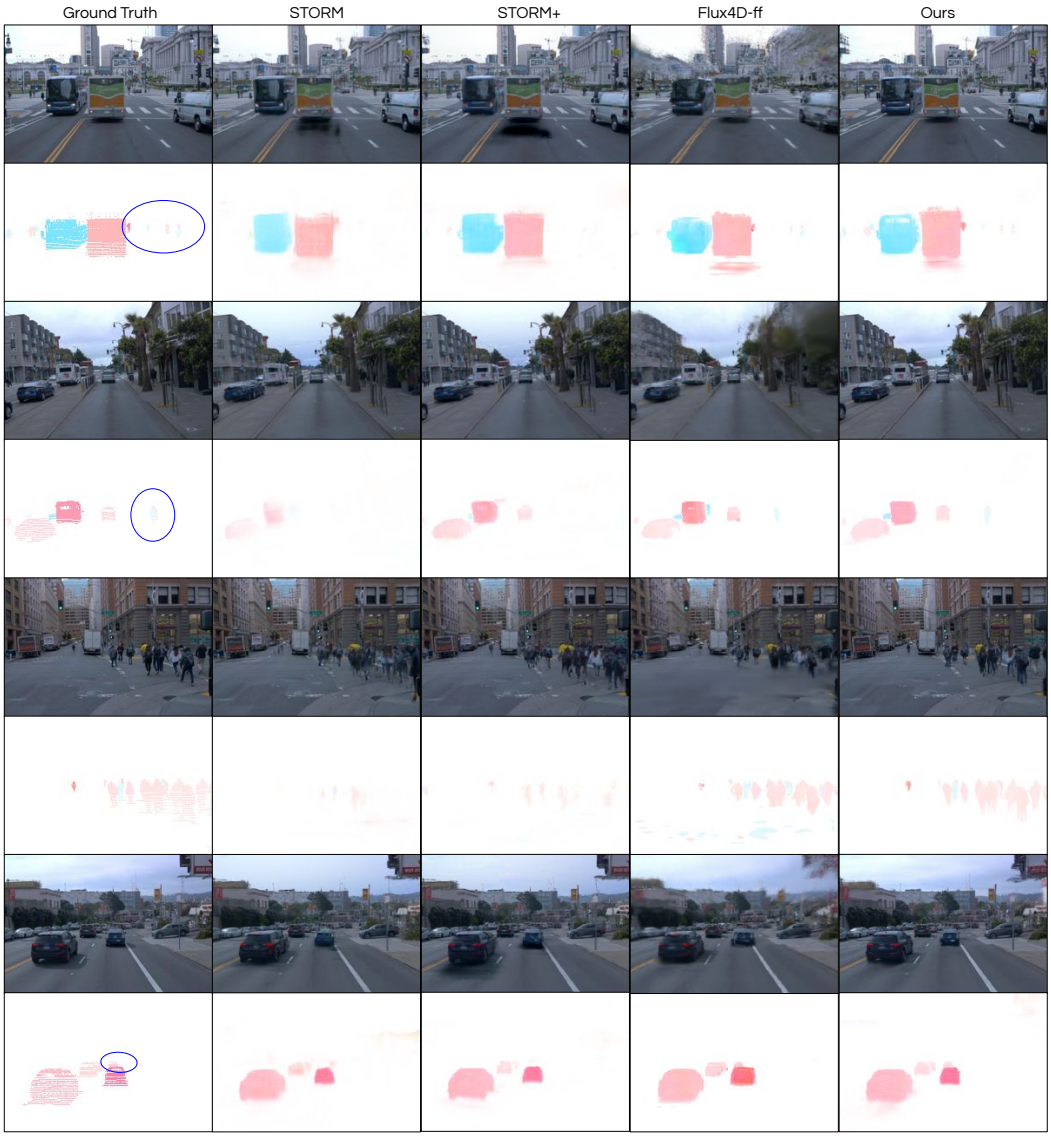


Figure 7: Qualitative comparisons of rendered RGB and flow images against SoTA. Our method has better image quality in dynamic regions, and produces more accurate flows. Flow color hues represent flow directions, and flow color intensities represent flow magnitudes. Our gains are more apparent for smaller objects, slower objects, and/or faraway objects.

change their placements, and inject new object instances (borrowed from other scenes) into the scene. These editing capabilities are crucial for simulation and downstream autonomy applications.

D Broader Impact

STRIDE is a self-supervised, generalizable, and decomposable 4D reconstruction method for dynamic driving scenes. Its ability to model dynamic urban environments from multi-modal sensory inputs without manual annotations makes it useful for downstream tasks such as reactive simulation. Additionally, *STRIDE* supports object-centric decomposition, enabling scene editing operations such as actor removal, manipulation, and injection, which may help generate diverse safety-critical scenarios for autonomy development. Creating digital twins of real-world locations may raise privacy concerns. Additionally, our system may exhibit unstable performance or unintended behavior on different datasets, particularly when sensory data are sparse, noisy, or out of distribution. Its predicted flow

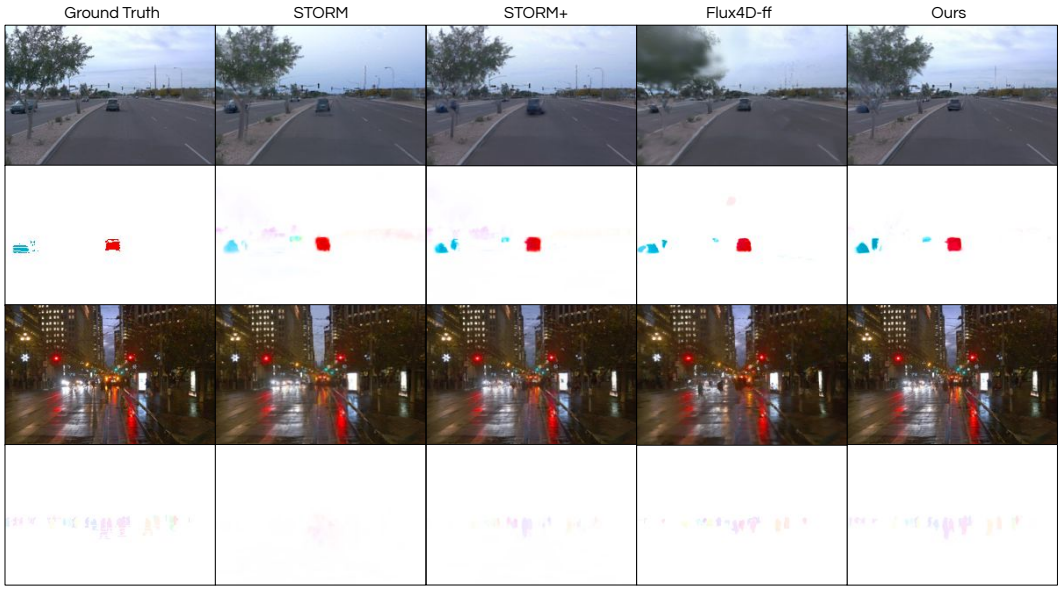


Figure 8: Qualitative comparisons of rendered RGB and flow images against SoTA. Our method has better image quality in dynamic regions, and produces more accurate flows. Flow color hues represent flow directions, and flow color intensities represent flow magnitudes. We have better reconstruction quality in the dynamic regions. In the challenging nighttime scene, we successfully reconstruct the flows of many pedestrians, while the image-only STORM fails to detect any. This example highlights the importance of using LiDAR.

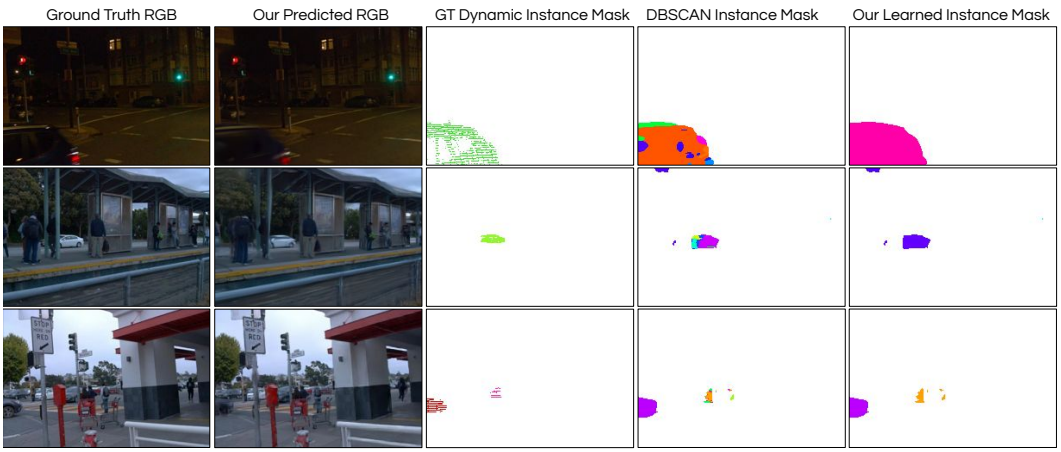


Figure 9: Qualitative examples of instance decomposition results. Our learned approach successfully improved over the clustering-based approach, which suffers from over-decomposition from noisy flow predictions and partial observations of actors. The learned approach can leverage motion cues and spatial priors to group the primitives into more complete instances.

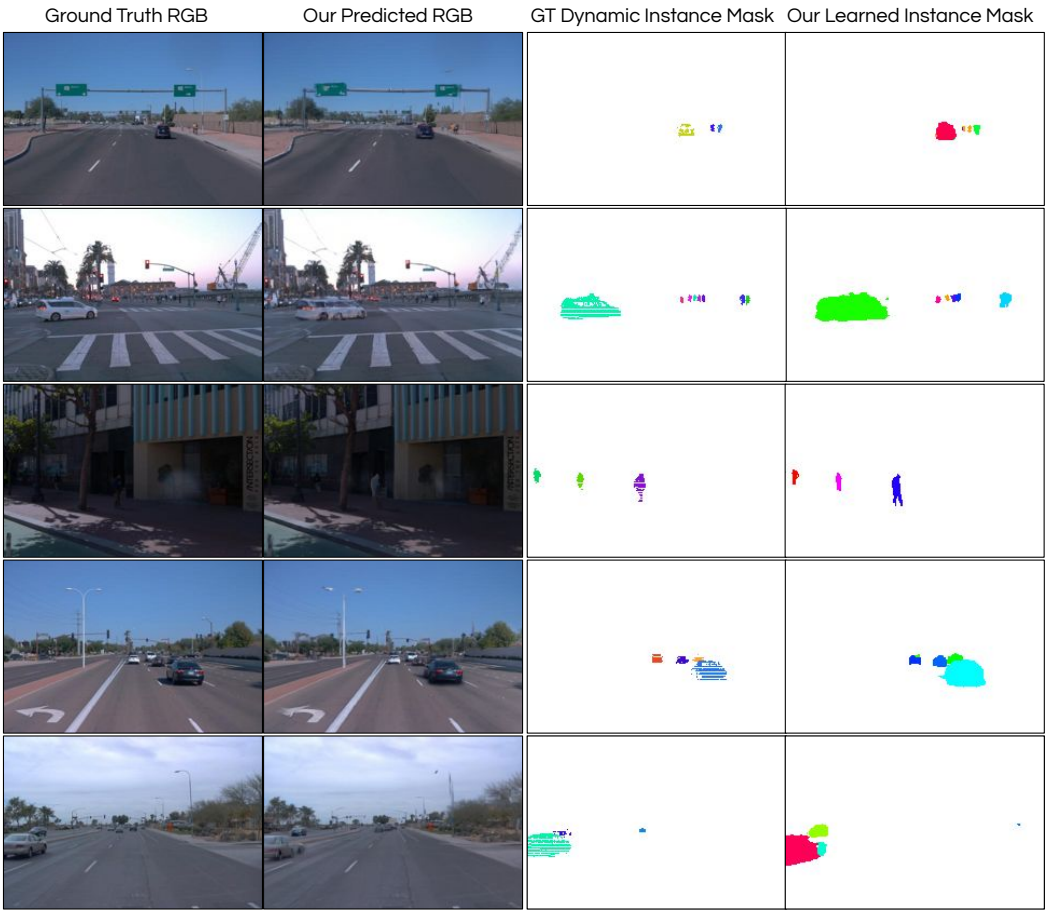


Figure 10: Qualitative examples of our instance decomposition results. General qualitative results randomly chosen from the validation set show that our method can successfully decompose dynamic instances in various scenes. Some failure cases are also observed. For example, in the second row, we group the two pedestrians on the right together, which is likely due to their proximity and similar motion. In the last row, there is a slight over-decomposition on the red car.



Figure 11: Qualitative examples of scene editing results for actor removal, manipulation and new actor injection.

and decompositions may also be imperfect, which could lead to inaccurate or misleading outcomes if directly used by downstream systems without validation.

E Licenses of Assets

We summarize the licenses and terms of use for all assets (datasets, software, code) in Table 12.

Assets	License	URL
Datasets		
PandaSet [49]	CC BY 4.0	https://scale.com/open-av-datasets/pandaset
WOD [39]	Custom	https://waymo.com/open/terms/
Codebases		
Point Transformer V3 [48]	MIT	https://github.com/Pointcept/PointTransformerV3
gsplat [56]	Apache-2.0	https://github.com/nerfstudio-project/gsplat
Baseline Codebases (Comparison purposes only)		
STORM [52]	NVIDIA License	https://github.com/NVlabs/GaussianSTORM
Flux4D [45]	N/A	https://waabi.ai/research/flux4d

Table 12: Summary of the licenses of assets.