OmniX: Any-view and Any-time 4D Reconstruction via Feed-forward Trajectory Fields

Yanqin Jiang[1,2†], Tengfei Wang[3‡], Zhengwei Wang[3], Chenjie Cao[3], Junta Wu[3], Wenhan Luo[4], Weiming Hu[1,2,5,6], Jin Gao[1,2,5‡], and Chunchao Guo[3]

[1] MAIS, Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Tencent Hunyuan
[4] HKUST
[5] Beijing Key Laboratory of Super Intelligent Security of Multi-Modal Information
[6] School of Information Science and Technology, ShanghaiTech University
jiangyanqin2021@ia.ac.cn; tengfeiwang12@gmail.com; jin.gao@nlpr.ia.ac.cn

Abstract. Previous feed-forward 4D reconstruction methods either predict per-frame static point clouds, ignoring foreground motion, or estimate point cloud trajectories while being limited to small camera motions. This limits their ability to aggregate observations over time and reconstruct complete dynamic scenes under large viewpoint changes. To address this, we propose OmniX, a feed-forward 4D reconstruction framework that predicts **dense 3D point trajectories for every pixel** from videos with **large camera motion**. 1) OmniX separates dynamic motion modeling from static geometry prediction and represents motion with a small set of dynamic tokens. Leveraging the sparse and low-rank structure of 3D motion, these tokens generate trajectory fields for all pixels in all images while efficiently preserving global interactions. 2) To facilitate training, we build an automatic UE5-based 4D data engine and introduce a dataset of 80k scenes and 1.28M multi-view videos with full geometric annotations. OmniX achieves state-of-the-art results on dense 3D point trajectory prediction and 3D point tracking, with competitive performance on video depth estimation and camera pose estimation.

Keywords: 4D Reconstruction, 3D Point Tracking, Synthetic Dataset

1 Introduction

Given image sequences of a dynamic scene, 4D reconstruction aims to reconstruct a dynamic 3D representation that is consistent across views and time. Serving as a foundation for dynamic scene understanding, it enables a wide range of downstream applications, including embodied AI simulation, autonomous driving perception, and AR/VR content creation. Thus, this area has attracted growing interest from the research community.

[†] Work done during an internship at Tencent Hunyuan.
[‡] Tengfei Wang and Jin Gao are co-corresponding authors.

Recent progress in 4D reconstruction has been largely driven by feed-forward approaches [5,7,18,32,38,41,46,48]. Most existing methods predict per-frame 3D point clouds from videos [7, 8, 16, 19, 38, 40, 46, 48], without explicitly modeling motion across time. To recover temporal correspondences, motion is often inferred by iteratively querying and tracking points across video frames [12, 25, 26, 41, 42, 44], which is computationally expensive and difficult to scale to dense predictions. A few works attempt to predict dense 3D point trajectories [5,18,32,33], but are typically limited to monocular videos with small camera motion. These methods struggle under large viewpoint changes because static geometry feature matching and dynamic foreground temporal correspondence learning are performed without disentanglement, thereby distracting from each other.

In this work, we present **Omni**$\mathbb{X}$, a 4D reconstruction framework that *predicts dense 3D point trajectories from images captured at arbitrary viewpoints and times*. It is robust to large camera motion and temporally discontinuous inputs by explicitly disentangling dynamic foreground motion from static scene structure. This is achieved by our proposed **Sparse Spatiotemporal Attention** (SSA) mechanism, which shows that, given the sparse and low-rank structure of 3D motion, a small set of dynamic tokens can effectively parameterize trajectory fields for all pixels across views and time. To predict per-pixel trajectories while keeping dynamic token selection in SSA differentiable, we further introduce a **Deformable Trajectory Sampling Head** (DTSH). Together, these designs enable efficient global attention and produce 3D point trajectories for all pixels in all images across all timestamps in one pass.

For training, due to the lack of 4D scene datasets with large camera motion and 360° multi-view coverage, we develop an automatic data engine based on Unreal Engine 5 (UE5). It synthesizes dynamic scenes from static environments and dynamic objects, and renders them with a customized camera system. Using this engine, we construct a dataset with **80K scenes and 1.28M multi-view videos**, annotated with depth, camera poses, and dense 3D point trajectories.

OmniX achieves state-of-the-art performance on 4D point cloud reconstruction and 3D point tracking, while delivering competitive performance on video depth estimation and camera pose estimation. Besides, we achieve an optimal accuracy-efficiency trade-off. These results consistently demonstrate the effectiveness of the proposed reconstruction framework and data generation pipeline.

2 Related Work

2.1 Feedforward 3D and 4D Reconstruction

The pinoreer work of feedfoward 3D reconstruction is DUSt3R [39], which introduces dense point map prediction from unposed images. Subsequent works significantly improve scalability through all-to-all attention [8, 14, 16, 19, 35, 40, 43]. Particularly, several recent works leverage dynamic-scene videos as training data and formulate 4D reconstruction as predicting per-frame 3D point clouds [3, 7, 8, 13, 16, 19, 40, 46, 48, 50]. However, these methods primarily reconstruct per-frame

3D geometry without explicitly establishing point-level correspondences across time. In contrast, our method augments per-frame point-cloud prediction with temporally consistent dense 3D point trajectories in a feed-forward manner.

2.2 3D Point Tracking

Early works achieve 3D point tracking through test-time optimization [36] or by lifting 2D tracks to 3D [15]. SpatialTracker [42] introduces the first feed-forward 3D point tracker, encoding frames into triplane representations and iteratively updating point trajectories in 3D space. Subsequent methods follow this iterative tracking paradigm, improving scalability [41], dense prediction [25, 26], and feature matching [44]. Recent methods [5, 18, 32, 33] directly regress dense 3D point trajectories, substantially improving efficiency. The former two use pairwise prediction, while the latter two predict trajectories for all pixels across all frames. However, these methods mainly adapt the point prediction heads of 3D reconstruction models for trajectory prediction, without explicitly disentangling dynamic motion from static geometry. Another work D4RT [45] proposes a flexible querying mechanism that removes both iterative trajectory updates and self-attention among queries, achieving impressive performance, but the model is not open-sourced. Concurrent works include 4RC [23] and Track4World [22]. 4RC disentangles static and dynamic prediction but does not employ sparse attention. Track4World improves the efficiency of the classic iterative tracking architecture, achieveing high precision, but its inference speed remains limited by iterative updates. In contrast, our method directly regresses dense 3D point trajectories while leveraging sparse attention and large-scale training data, enabling efficient and robust 3D point tracking under large camera motion.

3 Method

We aim at recovering consistent 4D geometry from any-view and any-time visual inputs, spanning the **single** video input, **multi-video** input and even the image-video **combinations**.

3.1 Formulation

We denote the input as a collection of videos $\mathcal{V} = \{\mathcal{V}^{(i)}\}_{i=1}^{N_v}$. The i-th video is represented by $\mathcal{V}^{(i)} = \{\mathbf{I}_j^{(i)}\}_{j=1}^{T^{(i)}}$, where $\mathbf{I}_j^{(i)} \in \mathbb{R}^{H\times W\times 3}$ denotes the j-th image in the i-th video and $T^{(i)}$ denotes the video length.

3D Geometry. Each image $\mathbf{I}_j^{(i)}$ is associated with a depth map $\mathbf{D}_j^{(i)} \in \mathbb{R}^{H\times W}$, a camera ray map $\mathbf{R}_j^{(i)} \in \mathbb{R}^{H\times W\times 3}$, the camera extrinsics $[\mathbf{Q}_j^{(i)} \mid \mathbf{t}_j^{(i)}]$ and intrinsics $\mathbf{K}_j^{(i)}$, where H and W are image height and width. Alternatively, we represent the camera parameters as $\mathbf{v}_j = (\mathbf{t}_j, \mathbf{q}_j, \mathbf{f}_j) \in \mathbb{R}^9$, where $\mathbf{t}_j \in \mathbb{R}^3$ denotes camera

translation, $\mathbf{q}_j \in \mathbb{R}^4$ is a rotation quaternion, and $\mathbf{f}_j \in \mathbb{R}^2$ represents the field-of-view parameters. Following prior work, for each pixel $\mathbf{p}$, we define a camera ray $\mathbf{r} = (\mathbf{t}, \mathbf{d}) \in \mathbb{R}^6$. Here, $\mathbf{t}$ and $\mathbf{d}$ denote the ray origin and direction in world coordinates respectively. The 3D point $\mathbf{P}$ for pixel location (u, v) is computed as

$$\mathbf{P} = \mathbf{t} + \mathbf{D}(u, v) \cdot \mathbf{d}. \tag{1}$$

4D Geometry. We also predict the trajectory of each 3D point over the timestamps of the input observations. Let $\mathbf{P} \equiv \mathbf{P}_{j_0}^{(i_0)}$ be a reference 3D point observed at time (i_0, j_0). We define its trajectory over the whole collection as $\{\mathbf{P}_j^{(i)}\}$. Inspired by Shape-of-Motion [37], we obtain this trajectory by a trajectory transformation $\boldsymbol{\tau} = \{[\mathbf{A}_j^{(i)} \mid \mathbf{b}_j^{(i)}]\}$, where $\mathbf{A}_j^{(i)}$ and $\mathbf{b}_j^{(i)}$ denote the rotation and translation matrices. The transformed point at video i and frame j is

$$\mathbf{P}_j^{(i)} = \mathbf{A}_j^{(i)} \, \mathbf{P}_{j_0}^{(i_0)} + \mathbf{b}_j^{(i)}. \tag{2}$$

Driven by the inherent sparsity of motion, we represent motion via **trajectory field** to ensure computational efficiency. The trajectory field, denoted as $\mathbf{T}_{j_0}^{(i_0)} = \{\boldsymbol{\tau}_{j_0,k}^{(i_0)}\}_{k=1}^{K}$, serves as a set of **trajectory transformation bases** for representing $\boldsymbol{\tau}$ in reference image $\mathbf{I}_{j_0}^{(i_0)}$. That is, we construct the per-pixel trajectory transformation $\boldsymbol{\tau}$ by a weighted combination:

$$\boldsymbol{\tau} = \sum_{k=1}^{K} w_k \, \boldsymbol{\tau}_{j_0,k}^{(i_0)}, \tag{3}$$

where $\{w_k\}$ are learned combination weights predicted from $\mathbf{I}_{j_0}^{(i_0)}$.

3.2 Architecture

Overview. As shown in Fig. 1, building upon the feed-forward 3D reconstruction models [16,19,35], OmniX consists of three main components: a transformer backbone, a DPT head for 3D geometry prediction, and a newly introduced trajectory module for 4D geometry prediction. Given any-view and any-time visual inputs, the transformer backbone encodes all images into tokens, injects sinusoidal encodings to embed timestamps for all images, and performs cross-view self-attention to aggregate multi-view information. The DPT head then predicts depth and ray maps for each image, from which 3D points are recovered. To lift 3D geometry to 4D, we introduce a trajectory module composed of a Sparse Spatiotemporal Attention (SSA) mechanism and a Deformable Trajectory Sampling Head (DTSH). SSA selects a set of dynamic tokens and expands them along the temporal dimension as trajectory queries. These expanded tokens are processed by stacked spatiotemporal cross-attention blocks, where they attend to all image tokens to establish spatiotemporal correspondences and predict trajectory transformation bases. The predicted bases are scattered back to their spatial token locations, forming a sparse trajectory field. Given this field, DTSH samples

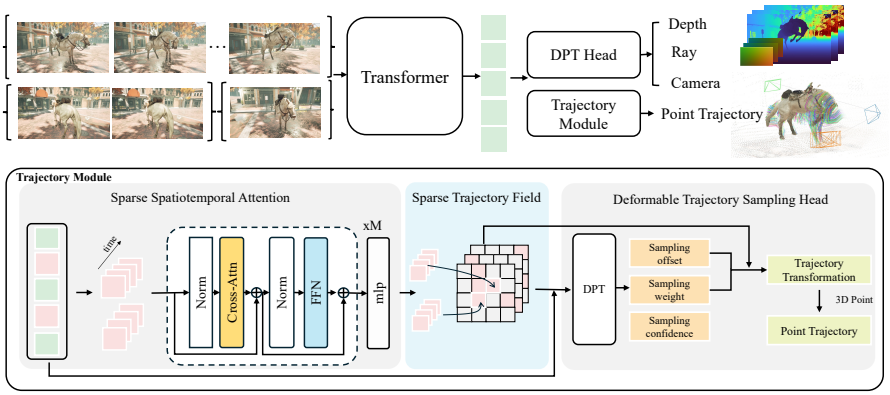

Fig. 1: Framework of OmniX. Built upon DepthAnything3 [16], OmniX predicts depth, ray maps, and camera parameters for the input images, from which 3D points are recovered. A trajectory module then predicts transformations for these 3D points. For efficiency, we propose a Sparse Spatiotemporal Attention (SSA) mechanism, which selects dynamic tokens, expands them across timestamps as queries, and lets them cross-attend to all image tokens to predict trajectory transformation bases. These bases are scattered back to the token grid to form a sparse trajectory field. A deformable trajectory sampling head upsamples this field to pixel-level transformation maps, which are applied to the 3D points to obtain dense trajectories.

neighboring bases for each pixel and combines them with the learned weights to produce pixel-level trajectory transformations, which lift the recovered 3D points to dense 4D trajectories. We next detail SSA, the sparse trajectory field construction, DTSH, and the differentiable dynamic token selection mechanism.

Sparse Spatiotemporal Attention. Given the image tokens $\mathbf{F}_j^{(i)} \in \mathbb{R}^{L\times d}$ from the j-th reference image of the i-th video, we employ an MLP φ to predict the dynamic probability $\boldsymbol{\Theta}_j^{(i)}$ and select the top-$\rho\%$ tokens as dynamic tokens, denoted as $\hat{\mathbf{F}}_j^{(i)} \in \mathbb{R}^{\hat{L}\times d}$. L and $\hat{L}$ are the number of image and dynamic tokens, respectively. Dynamic tokens are processed to predict trajectory transformation bases. We first expand them along temporal dimension by temporal embedding. The temporal embedding is constructed by adding the sinusoidal encodings of the video index and the local frame index, denoted by $\mathbf{e} \in \mathbb{R}^{\mathcal{T}\times d}$, where $\mathcal{T} = \sum_{i=1}^{N_v} T^{(i)}$ is the total number of input images. To capture temporal dynamics more effectively, we modulate the temporal embedding with per-token coefficients predicted from the dynamic tokens. The temporally expanded dynamic tokens $\tilde{\mathbf{F}}_j^{(i)} \in \mathbb{R}^{\mathcal{T}\times\hat{L}\times d}$ are computed as

$$\tilde{\mathbf{F}}_j^{(i)} = \hat{\mathbf{F}}_j^{(i)} + \gamma(\hat{\mathbf{F}}_j^{(i)}) \cdot \beta(\mathbf{e}), \tag{4}$$

where γ and β are MLPs with outputs $\gamma(\hat{\mathbf{F}}_j^{(i)}) \in \mathbb{R}^{\hat{L}\times 1}$ and $\beta(\mathbf{e}) \in \mathbb{R}^{\mathcal{T}\times d}$, respectively. The `broadcasting` along the temporal and token dimensions is omitted for simplicity.

We collect $\tilde{\mathbf{F}}_j^{(i)}$ and $\mathbf{F}_j^{(i)}$ from all input images and stack them along the first dimension, obtaining $\tilde{\mathbf{F}} \in \mathbb{R}^{\mathcal{T}\times\mathcal{T}\times\hat{L}\times d}$ and $\mathbf{F} \in \mathbb{R}^{\mathcal{T}\times L\times d}$. In the spatiotemporal cross-attention module, $\tilde{\mathbf{F}}$ serves as queries to extract geometric and motion cues from $\mathbf{F}$. Self-attention among query tokens is omitted, since experiments in Sec. 4.3 indicates that it provides negligible benefit in this module. For the m-th attention block, let the output be $\mathbf{Z}_c^m$ and the intermediate output be $\mathbf{Z'}_c^m$. The processing flow is formulated as follows:

$$\mathbf{Z'}_c^m = \mathrm{SSA}_c^m\Big(\mathrm{LN}(\mathbf{Z}_c^{m-1}), \mathbf{F}\Big) + \mathbf{Z}_c^{m-1}, \tag{5}$$

where $\mathbf{Z}_c^0 = \tilde{\mathbf{F}}$, and LN denotes layer normalization. The SSA_c^m operator aggregates spatiotemporal information through a multi-head cross-attention mechanism. For clarity, the core scaled dot-product attention operation is defined as follows:

$$\mathrm{SSA}_c^m(\mathbf{Z}^{\mathrm{r}}, \mathbf{F}^{\mathrm{r}}) = \mathrm{Softmax}\left(\frac{(\mathbf{Z}^{\mathrm{r}}\mathbf{W}_Q^m)(\mathbf{F}^{\mathrm{r}}\mathbf{W}_K^m)^T}{\sqrt{d}}\right)(\mathbf{F}^{\mathrm{r}}\mathbf{W}_V^m). \tag{6}$$

Here, $\mathbf{Z}^{\mathrm{r}} \in \mathbb{R}^{\tilde{\updownarrow}\times d}$ and $\mathbf{F}^{\mathrm{r}} \in \mathbb{R}^{\updownarrow\times d}$ are the reshaped output of the queries and image tokens. $\mathbf{W}_Q^m, \mathbf{W}_K^m, \mathbf{W}_V^m \in \mathbb{R}^{d\times d}$ represent the learnable projection matrices for generating queries, keys, and values in the m-th block, respectively. The cross-attention output is further processed by a feed-forward network, denoted as FFN, along with residual connections and layer normalization, as follows:

$$\mathbf{Z}_c^m = \mathbf{Z'}_c^m + \mathrm{FFN}\Big(\mathrm{LN}(\mathbf{Z'}_c^m)\Big). \tag{7}$$

Through this cascaded cross-attention architecture, the model can adaptively retrieve and integrate key geometric constraints and motion cues from the global image features $\mathbf{F}$, guided by the spatiotemporal priors in $\tilde{\mathbf{F}}$, thereby obtaining highly consistent trajectory representations.

Sparse Trajectory Field. The trajectory field $\mathbf{T} \in \mathbb{R}^{\mathcal{T}\times\mathcal{T}\times\hat{L}\times c}$ is predicted as follows:

$$\mathbf{T}^m = \phi^m(\mathbf{Z}_c^m), \tag{8}$$

where ϕ^m denotes a MLP layer. The channel dimension $c = 9$, comprises two components: the first six channels represent the 6D rotation parameters, and the remaining three channels correspond to the 3D translation vector. To enable efficient sampling in subsequent stages, we `scatter` these trajectory transformations back into 2D spatial grids according to the original positions of the dynamic tokens, obtaining a scattered sparse trajectory field. Finally, we reshape the field by merging the temporal dimension with the channel dimension to facilitate grid-based sampling operations, *i.e.*, the sparse trajectory field $\tilde{\mathbf{T}}^m \in \mathbb{R}^{\mathcal{T}\times\mathcal{H}\times\mathcal{W}\times(\mathcal{T}\cdot c)}$, where $\mathcal{H}\times\mathcal{W} = L$.

Deformable Trajectory Sampling Head. To obtain per-pixel trajectory transformations from sparse trajectory fields, we design a deformable sampling head based on the DPT architecture [30]. While the original DPT head operates on multi-level image tokens, we augment these tokens by embedding both the trajectory fields and per-token dynamic scores. Specifically, we aggregate these sparse fields via two pathways: a) a statistical branch that computes the temporal mean and variance of the sparse trajectory field, followed by a MLP projection; and b) a salient branch that projects each field individually before applying temporal max pooling. These embeddings, combined with MLP-projected dynamic scores, are concatenated with the image tokens to generate the sampling parameters, *i.e.*, sampling weights and sampling offsets.

To maintain the *differentiability of dynamic token selection*, we adopt a multi-scale deformable sampling mechanism to jointly sample trajectory transformations and dynamic scores, generating the final trajectories via weighted fusion. Specifically, the head predicts sampling offsets $\Delta\mathbf{p}_q^m$ and weights w_q^m for pixel $\mathbf{p}$, where q indexes the sampling point. Let $\mathcal{B}$ denotes bilinear sampling, we compute the per-pixel trajectory transformation $\boldsymbol{\tau}$ and dynamic score $\boldsymbol{\theta}$ as:

$$\boldsymbol{\tau} = \sum^{m}\sum^{q} w_q^m \mathcal{B}(\tilde{\mathbf{T}}^{\mathbf{m}}, \mathbf{p} + \boldsymbol{\Delta}\mathbf{p}_{\mathbf{q}}^{\mathbf{m}}), \tag{9}$$

$$\boldsymbol{\theta} = \sum^{m}\sum^{q} w_q^m \mathcal{B}(\boldsymbol{\Theta}, \mathbf{p} + \boldsymbol{\Delta}\mathbf{p}_{\mathbf{q}}^{\mathbf{m}}). \tag{10}$$

Here, $\boldsymbol{\Theta} \in \mathbb{R}^{\mathcal{T}\times\mathcal{H}\times\mathcal{W}}$ is the stacked and rearranged result of dynamic probability $\boldsymbol{\Theta}_j^{(i)}$ in SSA, shared by all levels. We compute the point trajectory $\mathbf{P}_{traj} \equiv \{\mathbf{P}_j^{(i)}\}$ following Eq. (2). To allow the trajectory loss to guide the selection of dynamic tokens, we incorporate the dynamic score $\boldsymbol{\theta}$ into the final trajectory prediction:

$$\hat{\mathbf{P}}_{traj} = \mathbf{P} + \boldsymbol{\theta} \cdot (\mathbf{P}_{traj} - \mathbf{P}). \tag{11}$$

The `broadcasting` of $\mathbf{P}$ is omitted for simplicity. This trajectory formulation also encourages background points to remain static during prediction.

3.3 Training Objectives

Following the formulation in Sec. 3.1, our model maps the video collection $\mathcal{V}$ to a depth map $\mathbf{D}$, a ray map $\mathbf{R}$, an optional camera pose $\mathbf{c}$, and a trajectory map $\mathbf{T}$. Additionally, we predict a token-level dynamic score map $\boldsymbol{\Theta}$ and a pixel-level dynamic score map $\boldsymbol{\theta}$. Before loss computation, all ground-truth signals are normalized by the average distance of ground-truth point cloud trajectories, following prior works, to ensure magnitude consistency across different modalities and stabilize the training process.

We supervise the depth, ray map, and point trajectory using confidence-aware losses. While the dynamic score maps do not theoretically require explicit supervision, we incorporate them with a small weighting factor for regularization when ground-truth dynamic masks are available. Let $\mathbf{D}^*$, $\mathbf{R}^*$, $\mathbf{c}^*$, and $\mathbf{T}^*$ denote

the ground-truth counterparts of the predicted variables. The total objective function is defined as:

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_R + \mathcal{L}_T + \beta\mathcal{L}_C + \alpha\mathcal{L}_{grad} + \kappa(\mathcal{L}_\Theta + \mathcal{L}_\theta), \tag{12}$$

where β, α, and κ are hyper-parameters balancing different tasks. For the geometric predictions $\mathbf{X} \in \{\mathbf{D}, \mathbf{R}, \mathbf{T}\}$, we incorporate a confidence-aware supervision mechanism. This allows the model to adaptively re-weight the regression loss based on the estimated uncertainty, thereby alleviating the impact of outliers or ambiguous regions (*e.g.*, occlusions and sky region). Let $\mathbf{C}_\mathbf{X}$ denote the predicted confidence map corresponding to variable $\mathbf{X}$. The regression loss is formulated as:

$$\mathcal{L}_{reg}(\mathbf{X}, \mathbf{X}^*) = \frac{1}{|\Omega|} \sum_{p \in \Omega} \left((1 + \lambda_1 \mathbf{C}_{\mathbf{X},p}) \|\mathbf{X}_p - \mathbf{X}_p^*\|_1 - \lambda_2 \log \mathbf{C}_{\mathbf{X},p} \right), \tag{13}$$

where Ω denotes valid pixels and λ_1 and λ_2 are hyperparameters balancing the weighted reconstruction error and the confidence regularization. This loss is applied as follows:

$$\mathcal{L}_D = \mathcal{L}_{reg}(\mathbf{D}, \mathbf{D}^*), \mathcal{L}_R = \mathcal{L}_{reg}(\mathbf{R}, \mathbf{R}^*), \mathcal{L}_T = \mathcal{L}_{reg}(\mathbf{T}, \mathbf{T}^*). \tag{14}$$

To preserve sharp edges while ensuring spatial consistency in planar regions, we introduce the gradient loss $\mathcal{L}_{grad}$ to penalize discrepancies in depth gradients:

$$\mathcal{L}_{grad}(\mathbf{D}, \mathbf{D}^*) = \|\nabla_x \mathbf{D} - \nabla_x \mathbf{D}^*\|_1 + \|\nabla_y \mathbf{D} - \nabla_y \mathbf{D}^*\|_1, \tag{15}$$

where ∇_x and ∇_y are the horizontal and vertical finite difference operators, respectively. For camera pose optimization, to ensure robustness against outliers, we apply the Huber loss $|\cdot|_\epsilon$:

$$\mathcal{L}_C = |f - f^*|_\epsilon + |\mathbf{q} - \mathbf{q}^*|_\epsilon + |\mathbf{t} - \mathbf{t}^*|_\epsilon, \tag{16}$$

where $|\cdot|_\epsilon$ is the Huber loss with threshold ϵ. Regarding the dynamic score maps, we employ the Binary Cross-Entropy (BCE) loss for regularization at both token and pixel granularities, whereas all other loss terms are formulated based on the ℓ_1 norm. Let $\mathbf{M} \in \{\boldsymbol{\Theta}, \boldsymbol{\theta}\}$ represent a dynamic score map at a specific scale and $\Omega_\mathbf{M}$ be its corresponding spatial domain. The BCE loss is defined as:

$$\mathcal{L}_{\text{BCE}}(\mathbf{M}, \mathbf{M}^*) = -\frac{1}{|\Omega_\mathbf{M}|} \sum_{p \in \Omega_\mathbf{M}} \left[\mathbf{M}_p^* \log \mathbf{M}_p + (1 - \mathbf{M}_p^*) \log(1 - \mathbf{M}_p) \right]. \tag{17}$$

Thus,

$$\mathcal{L}_\Theta = \mathcal{L}_{\text{BCE}}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^*), \quad \mathcal{L}_\theta = \mathcal{L}_{\text{BCE}}(\boldsymbol{\theta}, \boldsymbol{\theta}^*). \tag{18}$$

The joint optimization of loss terms of different tasks facilitates the understanding of the dynamic 3D scene.

4 Experiment

4.1 Setup

Data Preparation. To address the scarcity of large-scale 4D scene datasets featuring full geometric annotations and multi-view configurations, we develop an automated data generation engine based on Unreal Engine 5 [4]. This pipeline constructs dynamic scenes by compositing static environments with dynamic objects. A multi-camera system with diverse camera motion is designed to capture the dynamic scenes. We construct 160k dynamic scenes and render each with 16 cameras. After data cleaning, the dataset contains 80K scenes and approximately 1.28M multi-veiw videos, with 77K scenes used for training and evaluation in this work.

Implementation Details. In addition to our synthetic data, we incorporate eight public datasets to enhance diversity: DynamicReplica [11], PointOdessey [47], Spring [24], OmniGame [49], HOI4D [20], Waymo [34], DL3DV [17], and Stereo4D [9]. The model is configured with 8 SSA blocks. The weight for the dynamic mask loss is set to 0.01. During training, we use image resolution of 280×504 and sequence length of 16 frames; however, we observe that the model can generalize to longer sequences (*e.g.*, 32 frames or more) during inference. We employ the AdamW optimizer [21] with a learning rate of 2e-4. The model is trained for approximately 64k steps on a cluster of 64 GPUs, with a batch size of 1 per GPU. The total training time is approximately 12 days.

Evaluation. We establish a comprehensive benchmark on the validation set of the proposed dataset to evaluate per-pixel and per-frame 3D point trajectory prediction, referred to as dense 3D point trajectory prediction. The evaluation data consist of 600 videos from 40 scenes. This benchmark supports diverse input types, including monocular videos with complex camera motions, temporally disjoint video pairs, and hybrid image-video sets, and provides ground truth of dense 3D point trajectories. Following St4rTrack [5], we report the End-Point Error (EPE) and Average Point Distance (APD3D), both evaluated in the world coordinate and averaged by trajectories from all frames. We compute the two metrics for foreground points (FG) and all points (ALL) independently.

We also evaluate our model on sparse 3D point tracking task, using the TAPVid-3D dataset [15]. We further compare our method with state-of-the-art approaches on video depth estimation and camera pose estimation tasks, utilizing the KITTI [6], Sintel [1], and TUM-dynamic [31] datasets. At last, we evaluate the efficiency of the proposed framework. For all evaluation tasks, videos are clipped into 16-frame sequences to keep evaluation within the sequence length used during the training of our model and also by other dense 3D point trajectory prediction methods [18, 32].

4.2 Comparison with State-of-the-arts

Dense 3D Point Trajectory Prediction. Tab. 1 reports the evaluation results of dense 3D point trajectory prediction on the validation set of our proposed

Table 1: Dense 3D point trajectory prediction evaluation. The evaluated models are trained on different data sources. TraceAnything is trained on data generated by its custom engine. VDPM is trained on 4 datasets, including the unreleased Kubric-G [33]. Our model is trained on 9 datasets, including our custom UE dataset.

Method	Eval Type	Monocular Video				Disjoint Video Pairs				Hybrid Image-video sets			
		APD3D↑		EPE↓		APD3D↑		EPE↓		APD3D↑		EPE↓	
		FG	ALL	FG	ALL	FG	ALL	FG	ALL	FG	ALL	FG	ALL
TraceAnything [18]	first frame	0.062	0.043	0.354	0.316	0.059	0.037	0.380	0.343	0.030	0.027	0.521	0.447
VDPM [32]	first frame	0.187	0.120	0.338	0.287	0.164	0.114	0.341	0.306	0.146	0.117	0.458	0.405
Ours	first frame	**0.284**	**0.391**	**0.133**	**0.116**	**0.322**	**0.444**	**0.101**	**0.094**	**0.332**	**0.456**	**0.146**	**0.133**
VDPM [32]	all frames	0.199	0.118	0.332	0.282	0.143	0.093	0.345	0.305	0.135	0.104	0.341	0.306
Ours	all frames	**0.300**	**0.381**	**0.133**	**0.116**	**0.305**	**0.406**	**0.113**	**0.105**	**0.316**	**0.400**	**0.132**	**0.121**

Table 2: 3D point tracking evaluation on the TAPVid-3D benchmark [15]. Input videos are segmented into 16-frame clips. The evaluated models are trained on different data sources. SpatialTrackerV2 is trained on 17 datasets, including both synthetic and real-world datasets. St4RTrack is trained on 3 synthetic datasets. TraceAnything is trained on data generated by its custom engine. VDPM is trained on 4 datasets, including the unreleased Kubric-G [33]. Our model is trained on 9 datasets, including our UE dataset. Note that VDPM is trained on DriveTrack (Waymo) data, which may overlap with the test data, while we exclude the specific test scenes during training.

Method	**ADT** [27]		**DriveTrack** [34]		**PStudio** [10]	
	APD3D↑	**EPE↓**	**APD3D↑**	**EPE↓**	**APD3D↑**	**EPE↓**
SpatialTrackerV2 [41]	0.236	0.193	0.197	1.651	0.107	<u>0.149</u>
St4RTrack [5]	0.202	0.259	0.124	1.906	0.099	0.191
TraceAnything [18]	0.069	0.297	0.113	2.638	0.097	0.189
VDPM [32]	<u>0.266</u>	<u>0.162</u>	<u>0.228</u>	<u>1.576</u>	<u>0.118</u>	**0.141**
Ours	**0.367**	**0.134**	**0.256**	**1.490**	**0.141**	0.156

dataset. Since TraceAnything [18] does not predict camera parameters, we can only compute its metrics on the first frame rather than over the full sequence. In contrast, VDPM [32] predicts both camera parameters and dense 3D point trajectories, allowing comparison under the full evaluation setting. Our method outperforms both compared methods by a large margin under challenging input settings, especially for *Disjoint Video Pairs* and *Hybrid Image-video Sets*. This is mainly because the compared methods are not trained on videos with large camera motion, whereas our model explicitly includes such data during training. The significant performance gains on ADT [27], a dataset featuring large camera motion, in Tab. 2 further validate this point. We also observe that the foreground-point results of existing methods [18, 32] are better than their all-point results. This is because their inaccurate camera estimation often causes many background points to be projected out of view during metric computation, even after scale alignment, whereas foreground points are typically more spatially concentrated and are therefore less likely to be projected out of view.

Table 3: Video depth estimation on the KITTI dataset [6] (16-frame clips).

Metric	VGGT4D[7]	PAGE4D[48]	π^3[40]	SpatialTrackerV2[41]	DepthAnythingV3[16]	**Ours**
Abs Rel↓	0.041	<u>0.031</u>	0.032	0.049	0.039	**0.024**
$\delta < 1.25$↑	0.972	0.982	**0.990**	0.981	0.987	**0.990**

Table 4: Camera pose estimation (16-frame clips).

Method	**Sintel** [1]			**TUM-dynamics** [31]		
	ATE ↓	RPE_{trans} ↓	RPE_{rot} ↓	ATE ↓	RPE_{trans} ↓	RPE_{rot} ↓
VGGT4D [7]	0.321	0.322	1.165	0.011	<u>0.014</u>	<u>0.461</u>
SpatialTrackerV2 [41]	0.186	0.236	1.645	0.037	0.042	1.067
PAGE4D [48]	0.350	0.262	1.182	0.022	0.024	0.716
VDPM [32]	0.231	0.234	1.048	0.157	0.018	0.517
π^3 [40]	<u>0.157</u>	<u>0.174</u>	**0.627**	0.016	0.016	0.511
DepthAnythingV3 [16]	0.252	0.299	<u>0.710</u>	**0.010**	**0.013**	**0.458**
Ours	**0.108**	**0.152**	0.722	**0.010**	<u>0.014</u>	0.472

Due to space limitations, we omit qualitative comparisons here and instead provide visualizations of our method in Fig. 2, covering both synthetic and real-world dynamic scenes. As shown in Fig. 2, our method can reconstruct dynamic scenes from inputs with camera motions of up to 180° (1st scene), while accurately predicting 3D point trajectories under both non-rigid and rigid motion.

Sparse 3D Point Tracking. In Tab. 2, we report evaluation results on the TAPVid-3D benchmark [15]. Following standard protocols, videos are segmented into continuous 16-frame clips, and we evaluate all visible points starting from the first frame of each clip. Our method achieves state-of-the-art performance across all three datasets. Qualitative comparisons are conducted on the commonly used DAVIS dataset [29] as shown in Fig. 3. For visualization, we select points with the largest motion amplitudes while retaining background points, as background regions are also essential for dense reconstruction. We observe that: 1) In complex walking scenes, such as the first scene, comparison methods often fail to capture leg-crossing motion, resulting in tilting and warping artifacts. These methods tend to overfit to the object's static shape and fail to recover its actual motion. In contrast, our method correctly captures the forward-swinging and crossing motion patterns of the legs. 2) Comparison methods often fail to distinguish foreground objects from the static background and thus assign spurious displacements to background points. In contrast, our per-point dynamic score prediction better separates dynamic foreground points from the static background, reducing such false background displacements.

Other Tasks. In addition to trajectory prediction, we evaluate our model on other tasks such as video depth estimation and camera pose estimation. Videos are segmented to 16-frame clips. As shown in Tab. 3 and Tab. 4, our method achieves comparable performance to current state-of-the-art methods.

Fig. 2: Visualization of dense 3D point trajectory prediction under challenging camera motion. The first two scenes are from our custom dataset, and the last two row features a scene from the DexYCB [2] and Waymo [34] datasets. Each scene is shown as a 3×5 grid, where the top row presents input images at different timesteps. The first column of the second and third rows shows input images from different viewpoints, and columns 2–5 in the same row visualize the predicted trajectories of points predicted from the corresponding first-column image, at the timesteps indicated by the top row.

Fig. 3: Qualitative comparison of 3D point tracking on the DAVIS dataset [29].

Efficiency. Tab. 5 compares the inference efficiency of different methods when evaluated on 16-frame clips. TraceAnything [18] achieves the highest efficiency, attributed largely to its simple architecture, which only adds a tracking head to predict trajectory curves for pixels. However, its accuracy remains much lower than that of other methods, as shown in Tab. 1 and Tab. 2. Our method achieves the second-best efficiency, benefiting from the sparse spatiotemporal attention and a low-dimensional trajectory module (512-dimensional). Our method uses more memory because it natively predicts all trajectories in a single forward pass, rather than relying on an inference-time shortcut. In contrast, VDPM [32] and St4RTrack [5] are trained with sampled frame pairs or timestamps and require iterative inference loops, leading to relatively long runtimes.

4.3 Ablation Study

All ablation models are trained exclusively on our custom UE dataset. We primarily report the foreground APD3D metric since other metrics exhibit marginal

Table 5: Inference efficiency comparison evaluated on 16-frame clips.

Method	FLOPs (T)	Params (M)	Memory (MB)	Runtime (s)
St4RTrack [5]	629.52	575.4	**3273**	20.59
VDPM [32]	333.72	1660.0	15749	11.70
TraceAnything [18]	**8.79**	674.5	<u>12613</u>	**1.05**
Ours w/o sparse design	15.25	**568.7**	23727	4.10
Ours	<u>12.07</u>	**568.7**	21999	<u>2.15</u>

Table 6: Ablation study of SSA components. SelfAttn denotes self attention among temporally expanded dynamic tokens. Due to limited GPU memory, it is implemented sequentially as temporal attention followed by spatial attention.

Components				**APD3D(↑)**		
Eq. (4)-embed.	AdaLN-embed.	SelfAttn	CrossAttn	Monocular	Disjoint	Hybrid
	✓	✓	✓	0.260	0.267	0.291
✓		✓	✓	<u>0.293</u>	**0.298**	**0.324**
✓		✓		0.171	0.197	0.205
✓			✓	**0.297**	**0.298**	**0.324**

variance across different settings, and our primary focus lies in capturing foreground motion dynamics. Tab. 6 summarizes the ablation of core components within the proposed SSA module. 1) Temporal Embedding Strategy: The proposed embedding strategy in Eq. (4) outperforms the widely adopted AdaLN [28] mechanism. 2) Redundancy of Self Attention: We apply self attention to temporally expanded dynamic tokens using a factorized approach, *i.e.*, conducting attention solely along the temporal dimension and then followed by the spatial dimension. Surprisingly, these self-attention layers in the SSA module yield negligible performance gains. This indicates that the essential spatiotemporal reasoning is already effectively captured by other components. Consequently, we discard these layers in our final model to improve computational efficiency. 3) Crucial Role of Cross Attention: Cross attention is vital for motion prediction. Removing it (see last two rows) leads to a drastic performance drop, fundamentally causing the model to fail in capturing foreground dynamics, *i.e.*, mistakenly predicting static foregrounds. Qualitative comparisons for the SSA ablation study are visualized in Fig. 4. Specifically, our model with AdaLN exhibits restricted motion in the bear's foot and erratic movement in the dragon's wings. In contrast, "Ours w/ SelfAttn" yields results similar to "Ours", whereas "Ours w/o CrossAttn" produces nearly static predictions. These results further support our quantitative analysis. Tab. 7 presents the ablation study on the hyper-parameter of top-ρ%, and we use $\rho = 20$ for efficiency during training and inference. Tab. 8 further shows the effect of data scaling, indicating that larger training datasets lead to better accuracy.

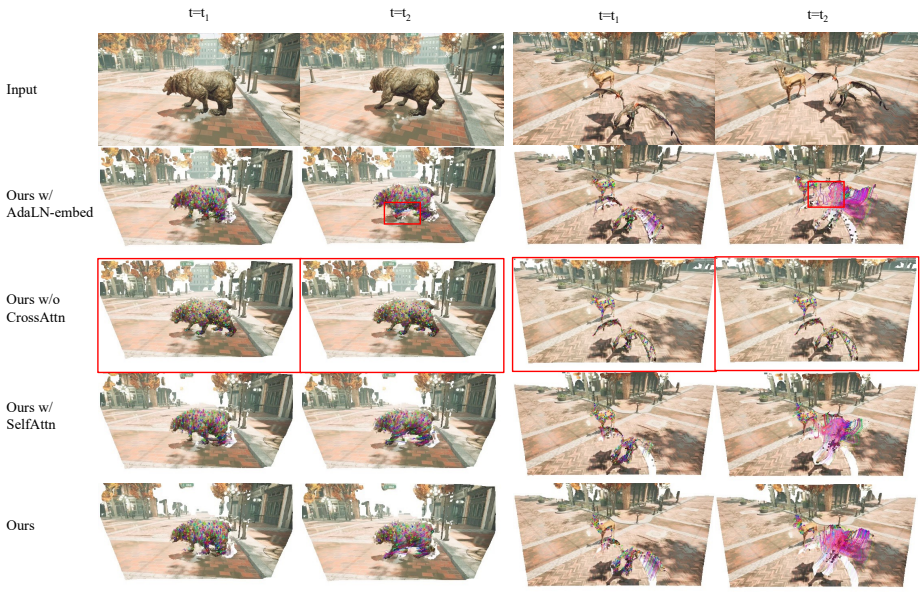

Fig. 4: Qualitative comparison for the ablation study of SSA components.

Table 7: Top-ρ% hyperparameter study on 4-frame monocular videos.

Top-ρ%	10	20	30
APD3D↑	0.3194	0.3356	**0.3454**

Table 8: Data scaling study on 16-frame monocular videos.

Training Size	2.8k	6.9k	13.9k
APD3D↑	0.182	0.210	**0.243**

5 Conclusion

In this paper, we present **OmniX** , a novel feed-forward framework for dense 4D dynamic scene reconstruction. To overcome the limitation of previous methods in handling large camera motions, we explicitly disentangle dynamic foreground motions from static background structures. To address the scarcity of 4D training data, we developed a UE5-based data engine, constructing a large-scale dataset of 80K scenes and 1.28M multi-view videos with dense geometric and trajectory annotations. Extensive experiments demonstrate that OmniX sets a new state-of-the-art in dense 3D point trajectory prediction and 3D point tracking, and meanwhile achieves competitive results on video depth and camera pose estimation. We believe our framework and the newly introduced data engine will serve as a strong foundation for future research in dynamic scene understanding.

Acknowledgments. This work was supported in part by the Beijing Natural Science Foundation (Grant No. L223003), the Natural Science Foundation of China (Grant No. 62422317, U22B2056, 62192782, 62532015, 62403462, U25A20480), and 2025 Tencent AI Lab Rhino-Bird Focused Research Program.

References

1. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Proceedings of the European Conference on Computer Vision. pp. 611–625 (2012)
2. Chao, Y.W., Yang, W., Xiang, Y., Molchanov, P., Handa, A., Tremblay, J., Narang, Y.S., Van Wyk, K., Iqbal, U., Birchfield, S., et al.: DexYCB: A benchmark for capturing hand grasping of objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9044–9053 (2021)
3. Deng, K., Ti, Z., Xu, J., Yang, J., Xie, J.: VGGT-Long: Chunk it, loop it, align it–pushing VGGT's limits on kilometer-scale long RGB sequences. arXiv preprint arXiv:2507.16443 (2025)
4. Epic Games: Unreal Engine 5 (2022), `https://www.unrealengine.com/en-US/unreal-engine-5`, accessed: 2026-06-24
5. Feng, H., Zhang, J., Wang, Q., Ye, Y., Yu, P., Black, M.J., Darrell, T., Kanazawa, A.: St4RTrack: Simultaneous 4D reconstruction and tracking in the world. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8503–8513 (2025)
6. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research **32**(11), 1231–1237 (2013)
7. Hu, Y., Cheng, C., Yu, S., Guo, X., Wang, H.: VGGT4D: Mining motion cues in visual geometry transformers for 4D scene reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 414–424 (2026)
8. HY-World, T., Cao, C., Zuo, X., Wang, Z., Zhang, Y., Wu, J., Liu, Z., Gong, Y., Liu, Y., Yuan, B., et al.: HY-World 2.0: A multi-modal world model for reconstructing, generating, and simulating 3D worlds. arXiv preprint arXiv:2604.14268 (2026)
9. Jin, L., Tucker, R., Li, Z., Fouhey, D., Snavely, N., Holynski, A.: Stereo4D: Learning how things move in 3D from internet stereo videos. arXiv preprint arXiv:2412.09621 (2024)
10. Joo, H., Simon, T., Li, X., Liu, H., Tan, L., Gui, L., Banerjee, S., Godisart, T.S., Nabbe, B., Matthews, I., et al.: Panoptic Studio: A massively multiview system for social interaction. IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (2017)
11. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: DynamicStereo: Consistent dynamic depth from stereo videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13229–13239 (2023)
12. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: CoTracker: It is better to track together. In: Proceedings of the European Conference on Computer Vision. pp. 18–35. Springer (2024)
13. Karhade, J., Keetha, N., Zhang, Y., Gupta, T., Sharma, A., Scherer, S., Ramanan, D.: Any4D: Unified feed-forward metric 4D reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14578–14589 (2026)
14. Keetha, N., Müller, N., Schönberger, J., Porzi, L., Zhang, Y., Fischer, T., Knapitsch, A., Zauss, D., Weber, E., Antunes, N., et al.: MapAnything: Universal feed-forward metric 3D reconstruction. arXiv preprint arXiv:2509.13414 (2025)

15. Koppula, S., Rocco, I., Yang, Y., Heyward, J., Carreira, J., Zisserman, A., Brostow, G., Doersch, C.: TAPVid-3D: A benchmark for tracking any point in 3D. Advances in Neural Information Processing Systems **37**, 82149–82165 (2024)
16. Lin, H., Chen, S., Liew, J., Chen, D.Y., Li, Z., Shi, G., Feng, J., Kang, B.: Depth Anything 3: Recovering the visual space from any views. arXiv preprint arXiv:2511.10647 (2025)
17. Ling, L., Sheng, Y., Tu, Z., Zhao, W., Xin, C., Wan, K., Yu, L., Guo, Q., Yu, Z., Lu, Y., et al.: DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22160–22169 (2024)
18. Liu, X., Xiao, Y., Chen, D.Y., Feng, J., Tai, Y.W., Tang, C.K., Kang, B.: Trace Anything: Representing any video in 4D via trajectory fields. arXiv preprint arXiv:2510.13802 (2025)
19. Liu, Y., Min, Z., Wang, Z., Wu, J., Wang, T., Yuan, Y., Luo, Y., Guo, C.: WorldMirror: Universal 3D world reconstruction with any-prior prompting. arXiv preprint arXiv:2510.10726 (2025)
20. Liu, Y., Liu, Y., Jiang, C., Lyu, K., Wan, W., Shen, H., Liang, B., Fu, Z., Wang, H., Yi, L.: HOI4D: A 4D egocentric dataset for category-level human-object interaction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21013–21022 (2022)
21. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
22. Lu, J., Xu, J., Hu, W., Zhu, R., Zhao, C., Yeung, S.K., Shan, Y., Liu, Y.: Track4World: Feedforward world-centric dense 3D tracking of all pixels. arXiv preprint arXiv:2603.02573 (2026)
23. Luo, Y., Zhou, S., Lan, Y., Pan, X., Loy, C.C.: 4RC: 4D reconstruction via conditional querying anytime and anywhere. arXiv preprint arXiv:2602.10094 (2026)
24. Mehl, L., Schmalfuss, J., Jahedi, A., Nalivayko, Y., Bruhn, A.: Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4981–4991 (2023)
25. Ngo, T., Zhuang, P., Kalogerakis, E., Gan, C., Tulyakov, S., Lee, H.Y., Wang, C.: DELTA: Dense efficient long-range 3D tracking for any video. In: Proceedings of the International Conference on Learning Representations (2025)
26. Ngo, T.D., Mirzaei, A., Qian, G., Liang, H., Gan, C., Kalogerakis, E., Wonka, P., Wang, C.: DELTAv2: Accelerating dense 3D tracking. arXiv preprint arXiv:2508.01170 (2025)
27. Pan, X., Charron, N., Yang, Y., Peters, S., Whelan, T., Kong, C., Parkhi, O., Newcombe, R., Ren, Y.C.: Aria Digital Twin: A new benchmark dataset for egocentric 3D machine perception. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20133–20143 (2023)
28. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4195–4205 (2023)
29. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 724–732 (2016)
30. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12179–12188 (2021)

31. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 573–580 (2012)
32. Sucar, E., Insafutdinov, E., Lai, Z., Vedaldi, A.: V-DPM: 4D video reconstruction with dynamic point maps. arXiv preprint arXiv:2601.09499 (2026)
33. Sucar, E., Lai, Z., Insafutdinov, E., Vedaldi, A.: Dynamic Point Maps: A versatile representation for dynamic 3D reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7295–7305 (2025)
34. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., 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)
35. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: VGGT: Visual geometry grounded transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5294–5306 (2025)
36. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19795–19806 (2023)
37. Wang, Q., Ye, V., Gao, H., Zeng, W., Austin, J., Li, Z., Kanazawa, A.: Shape of Motion: 4D reconstruction from a single video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9660–9672 (2025)
38. Wang, Q., Zhang, Y., Holynski, A., Efros, A.A., Kanazawa, A.: Continuous 3D perception model with persistent state. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10510–10522 (2025)
39. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: DUSt3R: Geometric 3D vision made easy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20697–20709 (2024)
40. Wang, Y., Zhou, J., Zhu, H., Chang, W., Zhou, Y., Li, Z., Chen, J., Pang, J., Shen, C., He, T.: π^3: Permutation-equivariant visual geometry learning. arXiv preprint arXiv:2507.13347 (2025)
41. Xiao, Y., Wang, J., Xue, N., Karaev, N., Makarov, Y., Kang, B., Zhu, X., Bao, H., Shen, Y., Zhou, X.: SpatialTrackerv2: Advancing 3D point tracking with explicit camera motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6726–6737 (2025)
42. Xiao, Y., Wang, Q., Zhang, S., Xue, N., Peng, S., Shen, Y., Zhou, X.: SpatialTracker: Tracking any 2D pixels in 3D space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20406–20417 (2024)
43. Yang, J., Sax, A., Liang, K.J., Henaff, M., Tang, H., Cao, A., Chai, J., Meier, F., Feiszli, M.: Fast3R: Towards 3D reconstruction of 1000+ images in one forward pass. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21924–21935 (2025)
44. Zhang, B., Ke, L., Harley, A., Fragkiadaki, K.: TAPIP3D: Tracking any point in persistent 3D geometry. Advances in Neural Information Processing Systems **38**, 135284–135303 (2026)
45. Zhang, C., Le Moing, G., Koppula, S., Rocco, I., Momeni, L., Xie, J., Sun, S., Sukthankar, R., Barral, J.K., Hadsell, R., et al.: Efficiently reconstructing dynamic scenes one D4RT at a time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7382–7392 (2026)
46. Zhang, J., Herrmann, C., Hur, J., Jampani, V., Cole, F., Sun, D., Yang, M.H., et al.: MonST3R: A simple approach for estimating geometry in the presence of motion. In: Proceedings of the International Conference on Learning Representations (2025)

47. Zheng, Y., Harley, A.W., Shen, B., Wetzstein, G., Guibas, L.J.: PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19855–19865 (2023)
48. Zhou, K., Wang, Y., Chen, G., Chang, X., Beaudouin, G., Zhan, F., Liang, P.P., Wang, M.: PAGE-4D: Disentangled pose and geometry estimation for 4D perception (2026)
49. Zhou, Y., Wang, Y., Zhou, J., Chang, W., Guo, H., Li, Z., Ma, K., Li, X., Wang, Y., Zhu, H., et al.: OmniWorld: A multi-domain and multi-modal dataset for 4D world modeling. arXiv preprint arXiv:2509.12201 (2025)
50. Zhuo, D., Zheng, W., Guo, J., Wu, Y., Zhou, J., Lu, J.: Streaming 4D visual geometry transformer. arXiv preprint arXiv:2507.11539 (2025)