TrackCraft3R: Repurposing Video Diffusion Transformers for Dense 3D Tracking

Jisu Nam[1] **Jahyeok Koo**[1] **Soowon Son**[1] **Jaewoo Jung**[1]
Honggyu An[1] **Junhwa Hur**[2†] **Seungryong Kim**[1†]

[1]KAIST AI [2]Google DeepMind

https://cvlab-kaist.github.io/TrackCraft3r

Abstract

Dense 3D tracking from monocular video is fundamental to dynamic scene understanding. While recent 3D foundation models provide reliable per-frame geometry, recovering object motion in this geometry remains challenging and benefits from strong motion priors learned from real-world videos. Existing 3D trackers either follow iterative paradigms trained from scratch on synthetic data or fine-tune 3D reconstruction models learned from static multi-view images, both lacking real-world motion priors. Pre-trained video diffusion transformers (video DiTs) offer rich spatio-temporal priors from internet-scale videos, making them a promising foundation for 3D tracking. However, their *frame-anchored* formulation, which generates each frame's content, is fundamentally mismatched with *reference-anchored* dense 3D tracking, which must follow the same physical points from a reference frame across time. We present TrackCraft3R, the first method to repurpose a video DiT as a feed-forward dense 3D tracker. Given a monocular video and its frame-anchored reconstruction pointmap, TrackCraft3R predicts a reference-anchored tracking pointmap that follows every pixel of the first frame across time in a single forward pass, along with its visibility. We achieve this through two designs: (i) a *dual-latent representation* that uses per-frame geometry latents and reference-anchored track latents as dense queries, and (ii) *temporal RoPE alignment*, which specifies the target timestamp of each track latent. Together, these designs convert the per-frame generative paradigm of video DiTs into a reference-anchored tracking formulation with LoRA fine-tuning. TrackCraft3R achieves state-of-the-art performance on standard sparse and dense 3D tracking benchmarks, while running $1.3\times$ faster and using $4.6\times$ less peak memory than the strongest prior method. We further demonstrate robustness to large motions and long videos.

1 Introduction

Recovering dense 3D trajectories from monocular video [7, 13, 57, 73, 80] is a fundamental building block for robotic manipulation [2, 38], dynamic scene reconstruction [17, 45], and controllable video generation [15, 55]. Because apparent motion is often dominated by camera ego-motion rather than object motion, accurate tracking requires reasoning in a 3D world coordinate frame in which camera motion is canceled out. Recent advances in monocular depth and pose estimation [18, 25, 44, 46] now provide reliable 3D geometry for arbitrary videos, enabling 3D trackers [57, 80, 88] to operate in a world coordinate frame where only residual object motion remains to be recovered.

Early 3D trackers [56, 57, 73, 79, 80] follow the 2D tracker paradigm such as CoTracker paradigm [33, 34], which iteratively updates trajectories based on local 3D correlation features, and is trained from

[†]Co-corresponding.

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

scratch on synthetic 4D datasets [16, 32, 91]. More recent feed-forward approaches [13, 35, 49, 68] instead fine-tune pre-trained 3D reconstruction models [37, 42, 76]. While their pre-trained models offer strong spatial priors, they are learned from static multi-view images, lack rich temporal priors from real-world videos.

On the other hand, recent works demonstrate that pre-trained video diffusion models [3, 81], especially video diffusion transformers (DiTs) [40, 72, 85], already encode strong spatio-temporal priors from internet-scale real videos and effectively transfer to perception tasks such as video depth [24, 65, 90], camera pose [29], and pointmap estimation [53].

This motivates a key question: *can we leverage the spatio-temporal priors of video DiTs for dense 3D tracking?* This is challenging because existing diffusion-based perception models produce *frame-anchored* outputs (*i.e.*, predictions defined independently at each frame [24, 29, 53, 65, 90]), whereas dense 3D tracking requires *reference-anchored* representations (*i.e.*, tracking the same physical points from a reference frame across time). A concurrent work, MotionCrafter [92], repurposes a video diffusion U-Net [3] for 4D reconstruction, but predicts *frame-anchored* scene flow between adjacent frames, requiring temporal chaining for dense 3D tracking and potentially leading to error accumulation, especially under occlusion.

In this paper, we introduce **TrackCraft3R**, the first method that repurposes a video diffusion transformer [72] as a feed-forward dense 3D tracker. Given a monocular video and its *frame-anchored* reconstruction pointmap in world coordinates [25, 44, 46], TrackCraft3R predicts, in a single forward pass, a *reference-anchored* tracking pointmap that tracks every pixel in the first frame across time, along with its visibility.

We achieve this by repurposing two core components of the video DiTs. First, we introduce a **dual-latent representation** consisting of (i) *geometry latents*, which encode each frame's RGB and reconstruction pointmap, and (ii) *first-frame anchored track latents*, which encode the reference frame's RGB and pointmap. The track latents act as dense query points defined in the first frame, while geometry latents represent 3D geometry over time in a shared world coordinate frame. Through full 3D attention, each track latent attends to geometry latents across frames to determine *where* its corresponding point is and *what* 3D position it should take. Second, we propose a **temporal RoPE alignment**, repurposing rotary positional embedding (RoPE) [67] to encode the target timestamp of each track latent, specifying *when* it attends to geometry latents. Together, TrackCraft3R enables dense 3D tracking with LoRA [23] fine-tuning, effectively converting the per-frame generative paradigm of video DiTs into a reference-anchored dense tracking paradigm.

TrackCraft3R achieves state-of-the-art performance on standard 3D sparse and dense tracking benchmarks [16, 31, 32, 41, 58, 91]. Notably, TrackCraft3R runs $1.3\times$ faster and uses $4.6\times$ less peak memory than the state-of-the-art 3D tracker DELTAv2 [57]. We further demonstrate robustness to large motions and long videos, and extensive ablations validate our design choices.

In summary, our contributions are threefold: (1) we present **TrackCraft3R**, the first method to repurpose a video diffusion transformer for feed-forward dense 3D tracking; (2) we propose a dual-latent representation and temporal RoPE alignment to convert frame-anchored generation into first-frame-anchored dense 3D tracking; and (3) we achieve state-of-the-art performance on standard 3D tracking benchmarks, while demonstrating robustness to large temporal strides and long videos.

2 Related Work

3D Point Tracking. Point tracking aims to recover long-range motion trajectories in videos. Early 2D tracking methods [6, 11, 12, 19, 33, 34, 63] iteratively refine trajectories within sliding temporal windows. To extend this to 3D, several works incorporate monocular depth [46, 84] and track in camera coordinates [56, 57, 73, 79], while others [80, 88] further utilize camera poses [25, 44, 46] to operate in a world coordinate frame, where camera motion is explicitly compensated. However, these methods rely on iterative trajectory updates and are trained from scratch on synthetic 4D datasets [16, 32, 91].

Recent feed-forward approaches [13, 17, 30, 35, 45, 49, 68] instead propose to fine-tune pre-trained 3D reconstruction models [37, 42, 76, 83] on synthetic 4D data. While these methods benefit from strong spatial priors of pre-trained models, they still lack strong temporal priors from real-world video dynamics. A concurrent work, MotionCrafter [92], incorporates temporal priors by

repurposing a video diffusion U-Net [3] for 4D reconstruction. However, it predicts frame-anchored scene flow between adjacent frames, requiring temporal chaining that accumulates errors under occlusion. In contrast, TrackCraft3R repurposes a video diffusion transformer to directly produce reference-anchored tracking pointmap in a single forward pass, avoiding temporal chaining.

Video Diffusion Models for Frame-Anchored Perception. Image diffusion models have been successfully repurposed for a wide range of perception tasks, including depth estimation [21, 36], surface normal prediction [14, 21], dense correspondence [22, 54, 71], and optical flow [64]. This paradigm has naturally extended to the video domain, where video diffusion models provide robust spatio-temporal priors. Early works repurpose video diffusion U-Nets [3, 81] for temporally consistent video depth estimation [24, 65], per-frame pointmap estimation [82], and joint estimation of depth, pointmaps, and ray maps [29]. Recently video diffusion transformers (DiTs) [40, 72, 85] has driven performance improvement across multiple tasks: DVD [90] repurposes the Wan 2.1 DiT [72] for video depth, and Sora3R [53] adapts an OpenSora DiT for pointmap prediction.

Despite the diversity of tasks, all these methods produce frame-anchored outputs, where predictions are tied to the content and timestamp of individual frames. Dense 3D tracking, by contrast, requires reference-anchored predictions that follow the same physical content from a reference frame across time. To the best of our knowledge, TrackCraft3R is the first to repurpose a video DiT for reference-anchored dense 3D tracking. A recent work [66] leverages video DiT features for sparse 2D point tracking. However, this method adds a tracking head (*e.g.*, a CoTracker head [34]) on top of the video DiT features, rather than repurposing the video DiT itself.

3 Preliminaries

Variational Autoencoder (VAE). A VAE encoder $\mathcal{E}$ maps a video $\mathbf{V} \in \mathbb{R}^{(1+F)\times H\times W\times 3}$ into a latent representation $\mathbf{z} = \mathcal{E}(\mathbf{V}) \in \mathbb{R}^{(1+f)\times h\times w\times c}$, where H, W, and $(1+F)$ denote the spatial resolution and number of frames, and h, w, and $(1+f)$ denote their spatially and temporally downsampled counterparts. c is the latent channel dimension. Here, temporal downsampling is applied only to the F frames, while the first frame is preserved. A decoder $\mathcal{D}$ reconstructs the video from $\mathbf{z}$.

Prior works show that VAEs pre-trained on RGB videos can be repurposed to encode and decode geometric modalities such as pointmaps [53, 82], depth maps [24, 90], and camera rays [28, 29], enabling diffusion models to operate in this latent space for geometric prediction.

Video Diffusion Transformers (DiTs). The latent $\mathbf{z}$ is patchified and projected, and a transformer f_θ is trained with rectified flow matching [48] to predict the velocity field along a linear interpolation between noise and data. The model applies full 3D attention, where each token i produces query $\mathbf{q}_i$, key $\mathbf{k}_i$, and value $\mathbf{v}_i$, and attends to all the other tokens j with weights proportional to $\mathbf{q}_i^\top \mathbf{k}_j/\sqrt{d_k}$, where d_k is the key dimension.

In this work, following [21, 90], we repurpose f_θ as a feed-forward regressor rather than a multi-step denoiser, enabling efficient inference without iterative sampling.

3D Rotary Positional Embedding (3D RoPE). To encode relative spatio-temporal structure, video DiTs employ 3D RoPE [67]. The channel dimension of each query and key vector is partitioned into temporal and spatial groups, and axis-specific rotation matrices are applied on each token's 3D position $\mathbf{p}_i = (x_i, y_i, t_i)$, where (x_i, y_i) denote spatial coordinates and t_i denotes the temporal index. Under RoPE, the attention score between tokens i and j becomes

$$\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{k}}_j = \mathbf{q}_i^\top \mathbf{R}_{\mathbf{p}_j - \mathbf{p}_i} \mathbf{k}_j, \tag{1}$$

where $\tilde{\mathbf{q}}_i$ and $\tilde{\mathbf{k}}_j$ denote the query and key vectors after applying RoPE. $\mathbf{R}_{\mathbf{p}_j-\mathbf{p}_i}$ is a block-diagonal rotation matrix parameterized by the relative offset $\mathbf{p}_j - \mathbf{p}_i$. Thus, attention depends only on relative positions, *i.e.*, tokens with similar t_i interact more strongly.

4 Video Diffusion Transformer for Dense 3D Tracking

We present a novel framework that densely tracks dynamic video content in a 3D world coordinate frame in a single forward pass. Recent 3D foundation models for depth and camera pose [25, 44, 46] provide reliable 3D scene geometry in world coordinates for arbitrary videos. Building on the pre-

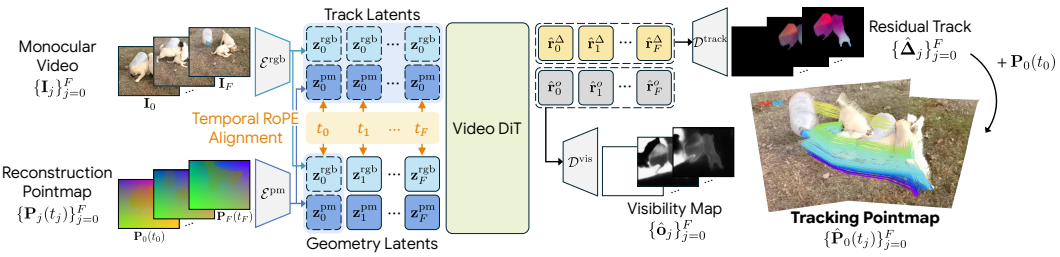


Figure 1: **Overall architecture.** Each RGB frame $\mathbf{I}_j$ and its reconstruction pointmap $\mathbf{P}_j(t_j)$ are encoded into RGB and pointmap latents using separate VAE encoders. A geometry latent is formed by channel-wise concatenation, and a track latent replicates the first-frame geometry latent across all frames. The latents are concatenated along the token dimension and processed by a video DiT, where RoPE assigns the same temporal index to each frame. The track latent outputs are decoded using separate VAE decoders into a residual track $\hat{\mathbf{\Delta}}_j$ and visibility $\hat{\mathbf{o}}_j$.

trained spatio-temporal priors of video diffusion transformers (DiTs), we leverage this 3D geometry as input and repurpose a video DiT to regress dense 3D tracks directly in this coordinate frame.

Specifically, we adopt two pointmap representations [13, 17, 68] that encode 3D geometry and motion: a *frame-anchored* pointmap as input and a *reference-anchored* pointmap as output. In Sec. 4.1, we formulate these pointmaps and define the problem.

However, the frame-anchored generative paradigm of video DiTs is fundamentally misaligned with dense 3D tracking, which requires reference-anchored predictions of the same physical points across time. To address this, we repurpose a video DiT with *dual-latent representation* and *temporal RoPE alignment*. Sec. 4.2 provides further details on the model architecture.

4.1 Problem Formulation

Following [13, 68], given a monocular video $\mathbf{V} = \{\mathbf{I}_t\}_{t=0}^{F} \in \mathbb{R}^{(1+F)\times H\times W\times 3}$, we define a time-dependent pointmap $\mathbf{P}_i(t_j) \in \mathbb{R}^{H\times W\times 3}$ as the 3D positions of the physical content observed in frame $\mathbf{I}_i$ at timestamp t_j. This provides a unified representation of dynamic scenes, jointly encoding 3D geometry and motion. All pointmaps are expressed in a shared world coordinate frame (we use the first frame as the reference frame), and we omit the coordinate index for simplicity.

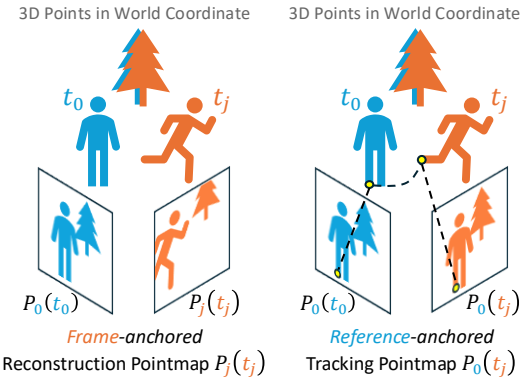


Figure 2: **Pointmap Representations.** Given 3D points of a dynamic scene in world coordinates, the reconstruction pointmap represents 3D points of $\mathbf{I}_j$'s content at t_j, while the tracking pointmap represents 3D points of $\mathbf{I}_0$ (reference frame) at t_j, so that all 3D points of $\mathbf{I}_0$ are tracked across time.

Reconstruction Pointmap. Each frame $\mathbf{I}_j \in \mathbb{R}^{H\times W\times 3}$ is lifted to 3D using depth and camera intrinsics, and transformed into the shared world coordinate frame via camera extrinsics. This yields a *frame-anchored* reconstruction pointmap $\mathbf{P}_j(t_j) \in \mathbb{R}^{H\times W\times 3}$, which represents the 3D positions of the content in frame $\mathbf{I}_j$ at its own timestamp t_j. Note that such pointmaps can be readily obtained either from ground-truth [16, 32, 91] or from estimated depth and camera pose using recent 3D foundation models [25, 44, 46].

Tracking Pointmap. To enable tracking, we define a *reference-anchored* tracking pointmap $\mathbf{P}_0(t_j) \in \mathbb{R}^{H\times W\times 3}$, which represents the 3D positions of the content originally observed in the reference frame $\mathbf{I}_0$ at timestamp t_j. Here, the reference index is fixed to 0 while time varies, so the same physical points from $\mathbf{I}_0$ are tracked consistently across frames. Fig. 2 illustrates both pointmaps.

Our Objective. Given a video $\mathbf{V} = \{\mathbf{I}_j\}_{j=0}^{F}$ and its reconstruction pointmaps $\{\mathbf{P}_j(t_j)\}_{j=0}^{F}$, which provide per-frame 3D geometry in a shared world coordinate frame, our goal is to predict the tracking pointmaps $\{\mathbf{P}_0(t_j)\}_{j=0}^{F}$ that establish dense 3D correspondences across time by tracking the physical

content of the reference frame $\mathbf{I}_0$ throughout the sequence. In addition, we predict visibility maps $\{\mathbf{o}_j\}_{j=0}^{F}$, where $\mathbf{o}_j \in [0,1]^{H\times W}$ indicates whether each tracked point from $\mathbf{I}_0$ is visible at time t_j.

4.2 Model Architecture

An overview of our architecture is shown in Fig. 1. Given a video $\{\mathbf{I}_j\}_{j=0}^{F}$ and its reconstruction pointmaps $\{\mathbf{P}_j(t_j)\}_{j=0}^{F}$, we encode each RGB frame and pointmap independently using separate VAE encoders $\mathcal{E}^{\text{rgb}}$ and $\mathcal{E}^{\text{pm}}$, yielding per-frame RGB latents $\mathbf{z}_j^{\text{rgb}}$ and pointmap latents $\mathbf{z}_j^{\text{pm}}$:

$$\mathbf{z}_j^{\text{rgb}} = \mathcal{E}^{\text{rgb}}(\mathbf{I}_j) \in \mathbb{R}^{h\times w\times c}, \quad \mathbf{z}_j^{\text{pm}} = \mathcal{E}^{\text{pm}}(\mathbf{P}_j(t_j)) \in \mathbb{R}^{h\times w\times c}. \tag{2}$$

To preserve per-frame spatial precision, we bypass temporal compression in the original 3D VAE by treating the temporal dimension as a batch dimension [55] (see the ablation in Tab. 3).

Point Map Normalization. Prior to VAE encoding, each pointmap is normalized by subtracting the mean and dividing by the maximum distance from the mean, both computed over points whose depths fall within the 2%–98% percentile range across all frames to exclude outliers. As a result, the normalized values lie approximately within $[-1, 1]$.

Dual-Latent Representation. To repurpose a video DiT for reference-anchored 3D tracking, we define two types of latents for the model input: a *geometry latent* $\mathbf{g}_j$, which encodes 3D geometry at timestamp t_j, and a *first-frame-anchored track latent* $\mathbf{r}_j$, which serves as a dense query anchored to the reference frame $\mathbf{I}_0$ for tracking across time.

To explicitly couple RGB appearance $\mathbf{z}_j^{\text{rgb}}$ and 3D geometry $\mathbf{z}_j^{\text{pm}}$ at each spatial location, the geometry latent $\mathbf{g}_j$ is formed by channel-wise concatenation at timestamp t_j. To anchor tracking to the reference frame, the track latent $\mathbf{r}_j$ is obtained by replicating the first-frame geometry latent across all timestamps:

$$\mathbf{g}_j = [\mathbf{z}_j^{\text{rgb}};\ \mathbf{z}_j^{\text{pm}}] \in \mathbb{R}^{h\times w\times 2c}, \qquad \mathbf{r}_j = \mathbf{g}_0 \in \mathbb{R}^{h\times w\times 2c}, \tag{3}$$

where $[\cdot;\cdot]$ denotes channel-wise concatenation.

We concatenate the geometry and track latents along the token dimension and process them with a video DiT f_θ:

$$\{\hat{\mathbf{r}}_j\}_{j=0}^{F} = f_\theta\big([\{\mathbf{g}_j\}_{j=0}^{F},\ \{\mathbf{r}_j\}_{j=0}^{F}]\big), \tag{4}$$

where $[\cdot,\cdot]$ denotes concatenation along the token sequence dimension. The outputs corresponding to the track latents, $\hat{\mathbf{r}}_j \in \mathbb{R}^{h\times w\times 2c}$, are used for tracking pointmap and visibility prediction.

Intuitively, RGB latents provide cues for spatial matching, while pointmap latents store the associated 3D positions. Once $\mathbf{z}_0^{\text{rgb}}(u_r, v_r)$ in the track latent $\mathbf{r}_j$ matches the same physical point as $\mathbf{z}_j^{\text{rgb}}(u_g, v_g)$ in the geometry latent $\mathbf{g}_j$ via attention, the corresponding pointmap latent $\mathbf{z}_j^{\text{pm}}(u_g, v_g)$ directly provides its 3D position $\mathbf{P}_j(t_j)(u_g, v_g)$, which defines the tracked point $\mathbf{P}_0(t_j)(u_r, v_r)$. Here, (u_r, v_r) denotes spatial coordinates in the track latent, and (u_g, v_g) denotes the corresponding spatial coordinates in the geometry latent.

To convert the video DiT into a one-step regressor, we fix the diffusion timestep to zero and use a null text prompt. We further evaluate the inference efficiency of our one-step model in Tab. 6.

Temporal RoPE Alignment. To ensure that each track latent attends to the geometry latent at the correct timestamp, we utilize the temporal axis of 3D RoPE [67]. As illustrated in Fig. 1, we assign both $\mathbf{g}_j$ and $\mathbf{r}_j$ the same temporal RoPE index t_j (Eq. 1). Since RoPE encodes relative position, tokens with identical temporal indices exhibit stronger attention. Consequently, each track latent $\mathbf{r}_j$ attends to the geometry latent $\mathbf{g}_j$ at timestamp t_j, retrieving the corresponding 3D position.

Fig. 3(a) visualizes the query–key attention from $\mathbf{r}_5$ to $\{\mathbf{g}_k\}_{k=0}^{F}$, showing that attention is predominantly localized on $\mathbf{g}_5$, confirming that temporal RoPE alignment correctly specifies the target timestamp. Fig. 3(b) further visualizes the attention between $\mathbf{r}_5$ and $\mathbf{g}_5$ across different transformer layers, showing that full 3D attention effectively establishes accurate correspondences between track and geometry latents under motion. Full attention visualizations and additional discussion are provided in the Appendix E.

Trajectory and Visibility Prediction. We decode the video DiT outputs corresponding to the track latents, $\hat{\mathbf{r}}_j$, into a tracking pointmap $\hat{\mathbf{P}}_0(t_j)$ and a visibility map $\hat{\mathbf{o}}_j$. The latent $\hat{\mathbf{r}}_j \in \mathbb{R}^{h\times w\times 2c}$ is

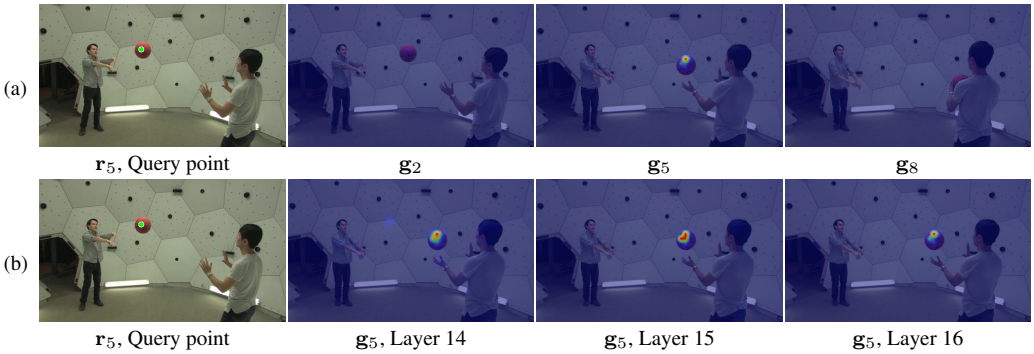


Figure 3: **Query-key attention visualization.** The query point is marked as a green circle. (a) Attention from track latents $\mathbf{r}_5$ at timestamp t_5 to geometry latents $\{\mathbf{g}_j\}_{j=0}^{F}$. Attention is predominantly localized on $\mathbf{g}_5$, showing that RoPE correctly assigns a target timestamp to each track latent. (b) Within $\mathbf{g}_5$, attention aligns with the same physical point under motion, demonstrating accurate dense correspondence between track and geometry latents.

channel-wise partitioned into two components: the first half is used for pointmap prediction, and the second half for visibility prediction.

Instead of directly regressing $\mathbf{P}_0(t_j)$, we predict a residual track with respect to the reference frame:

$$\mathbf{\Delta}_j = \mathbf{P}_0(t_j) - \mathbf{P}_0(t_0). \tag{5}$$

This residual formulation stabilizes training and improves accuracy (see Tab. 3), as $\mathbf{\Delta}_j = \mathbf{0}$ for static regions while non-zero values capture motion-induced displacement.

We decode $\hat{\mathbf{r}}_j$ using two separate VAE decoder heads:

$$\hat{\mathbf{\Delta}}_j = \mathcal{D}^{\text{track}}(\hat{\mathbf{r}}_j^{\Delta}), \qquad \hat{\mathbf{o}}_j = \mathcal{D}^{\text{vis}}(\hat{\mathbf{r}}_j^{o}), \tag{6}$$

where $\hat{\mathbf{r}}_j^{\Delta}$ and $\hat{\mathbf{r}}_j^{o}$ denote the channel-wise partitions. Here, $\hat{\mathbf{\Delta}}_j \in \mathbb{R}^{H\times W\times 3}$ is defined in the normalized pointmap space, and $\hat{\mathbf{o}}_j \in [0,1]^{H\times W}$ denotes visibility.

Since the VAE decoder produces three-channel outputs, the visibility map is broadcast to three channels to match the output dimensionality [36]. For pointmap normalization, we use the same factors (mean and maximum distance) as those of $\mathbf{P}_j(t_j)$ to ensure that the same physical point has the same 3D position after normalization.

Finally, the tracking pointmap is recovered as:

$$\hat{\mathbf{P}}_0(t_j) = \mathbf{P}_0(t_0) + \hat{\mathbf{\Delta}}_j. \tag{7}$$

Long-Video Inference. Our model is trained on clips of $1{+}F$ frames. To handle longer videos at inference time, we adopt a strided sliding window strategy with the first frame as a fixed anchor.

Given a test video of L frames, we compute the stride as $s = \lceil (L{-}1)/F \rceil$ and partition the frames $\{1, \ldots, L{-}1\}$ into s non-overlapping groups. Each forward pass processes the anchor frame $\mathbf{I}_0$ together with F frames sampled from one group, resulting in s passes that cover the entire sequence. For each pass, we assign consecutive RoPE temporal indices $\{0, 1, \ldots, F\}$ as in training, regardless of the original frame indices. As in [13, 20], the model is trained with various temporal strides and naturally generalizes to non-consecutive frames. The predicted pointmaps are consistent across passes without post-processing, as all inputs $\mathbf{P}_j(t_j)$ share a common world coordinate frame. Fig. 5 further evaluates the robustness of our method on long videos and large temporal strides.

5 Experiment

5.1 Implementation Details

Architecture. We fine-tune Wan 2.1-T2V [72] using LoRA [23]. Because the input and output token channel dimensions are doubled, we duplicate the DiT input projection weights [5]. For the output

Table 1: **3D tracking comparison.** We report AJ, APD_{3D}, and OA after Sim(3) alignment. The best and second-best results are highlighted in dark and light blue, respectively.

Method	ADT [58]			PStudio [31]			DR [32]			PO [91]			Kubric [16]			Average		
	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
(i) Iterative dense 3D trackers																		
DELTA [56] + ViPE [25]	0.5088	0.6949	0.8142	0.4987	0.7810	0.6959	0.4049	0.5847	0.7638	0.4559	0.6286	0.8116	0.2894	0.3721	0.9630	0.4315	0.6123	0.8097
DELTAv2 [57] + ViPE [25]	0.5135	0.7066	0.8038	0.5353	0.8026	0.7284	0.4167	0.5888	0.7825	0.4459	0.6246	0.8011	0.2860	0.3692	0.9564	0.4395	0.6184	0.8144
DELTAv2 [57] + DA3 [46]	0.6150	0.8219	0.8125	0.5571	0.8496	0.7087	0.4494	0.6217	0.7817	0.5304	0.7251	0.8020	0.3354	0.4106	0.9592	0.4975	0.6858	0.8128
(ii) Feed-forward dense 3D trackers based on 3D reconstruction models																		
St4RTrack [13]	0.5929	0.7683	0.8323	0.5723	0.7552	0.8099	0.3534	0.5710	0.6836	0.3968	0.6579	0.6860	0.1193	0.1896	0.7703	0.4069	0.5884	0.7564
Any4D [35]	0.4646	0.6134	0.8358	0.4222	0.5707	0.8128	0.4414	0.6955	0.6801	0.4387	0.6830	0.7353	0.3887	0.4967	0.8826	0.4311	0.6119	0.7893
TraceAnything [49]	0.5929	0.7634	0.8412	0.5225	0.6928	0.8130	0.2072	0.3549	0.7332	0.2041	0.3649	0.6930	0.2418	0.3252	0.8198	0.3537	0.5002	0.7800
(iii) Feed-forward dense 3D trackers based on video generative models																		
MotionCrafter [92]	0.4460	0.6037	0.8036	0.5044	0.6659	0.8142	0.4926	0.6172	0.9172	0.4197	0.6412	0.7301	0.2176	0.3011	0.8730	0.4161	0.5658	0.8276
TrackCraft3R + ViPE [25]	0.6683	0.7688	0.9405	0.6803	0.8163	0.8937	0.5842	0.7034	0.9408	0.5836	0.7262	0.8943	0.3032	0.3940	0.9597	0.5639	0.6817	0.9258
TrackCraft3R + DA3 [46]	0.8626	0.9510	0.9445	0.7287	0.8713	0.8891	0.6518	0.7706	0.9388	0.7288	0.8678	0.8938	0.4208	0.5047	0.9587	0.6785	0.7931	0.9250

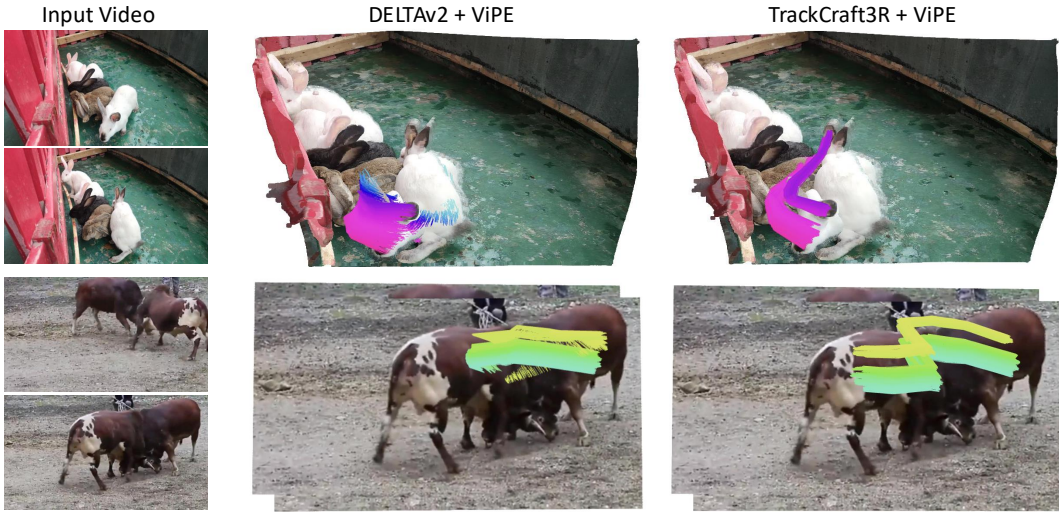


Figure 4: **Qualitative comparison on ITTO [10] videos.** TrackCraft3R accurately estimates dense 3D trajectories on real-world videos under large object dynamics and occlusion.

projection, we retain the pre-trained weights for the first half of the channels and zero-initialize the remaining half. All VAE components are initialized from the pre-trained Wan VAE weights.

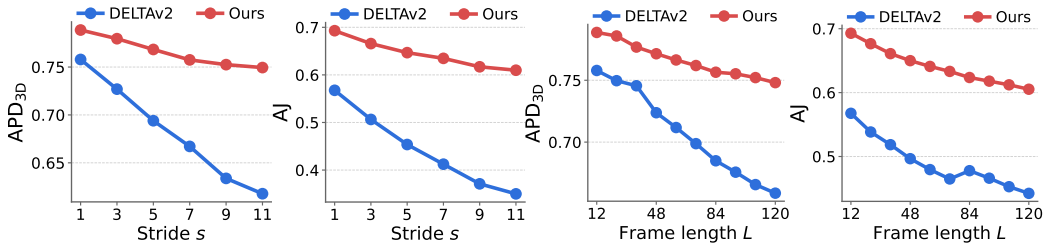
Training. All models are trained at a resolution of 480×832 on 12-frame clips using 8 H200 GPUs. Training proceeds in two stages. In Stage 1, we train the DiT with LoRA and input/output projection layers, with VAEs being frozen. We use AdamW [50] with a learning rate of 1e-4 and a global batch size of 80 for 3 days. In Stage 2, we unfreeze all VAE encoders and decoders, $\mathcal{E}^{\text{rgb}}, \mathcal{E}^{\text{pm}}, \mathcal{D}^{\text{track}}, \mathcal{D}^{\text{vis}}$, and continue end-to-end training with learning rates of 3e-5 for the DiT and 1e-5 for the VAE, using a global batch size of 64 for an additional 2 days.

Training Objective. We minimize an MSE loss [57] on the predicted residual $\hat{\mathbf{\Delta}}_j$ in normalized pointmap space, combined with a BCE loss [34] on visibility $\hat{\mathbf{o}}_j$, weighted by 0.1.

Dataset. Following [13], we train our model on Kubric [16], PointOdyssey [91], and Dynamic Replica [32], which provide ground-truth 3D trajectories from mesh vertices. We also include TartanAir [77], a static-scene dataset with large camera motion, to improve robustness to ego-motion. More details are provided in Appendix A.

5.2 Evaluation Settings

Evaluation Datasets. We evaluate our method on both 3D sparse and dense tracking benchmarks, with all metrics computed in a world coordinate frame. For *3D sparse tracking*, following [13], we use two real-world datasets from TAPVid-3D [41], Aerial Digital Twin (ADT) [58] and Panoptic Studio (PStudio) [31], along with two synthetic test dataset, Point Odyssey (PO) [91] and Dynamic Replica (DR) [32]. Each dataset provides sparse ground-truth 3D trajectories, and we evaluate the first 84 frames. For *3D dense tracking*, following [35, 56, 57], we use the held-out Kubric [16] test



(a) APD_{3D}, varying s (b) AJ, varying s (c) APD_{3D}, varying L (d) AJ, varying L

Figure 5: **Robustness to large inter-frame motion (a, b) and long videos (c, d). TrackCraft3R**'s performance drops much more slowly than **DELTAv2** as stride s or frame length L grows.

split consisting of 50 sequences. This dataset provides dense ground-truth 3D trajectories defined for every pixel in the reference frame, and we evaluate the first 24 frames following [57].

Evaluation Metrics. Following TAPVid-3D [41], we report three metrics: (i) *average percentage of points within* δ_{3D} (APD_{3D}), defined as the percentage of points whose 3D end-point error is below a threshold $\delta_{3D} \in \{0.1, 0.3, 0.5, 1\}$m [13], averaged over thresholds; (ii) *occlusion accuracy* (OA), which measures the accuracy of occlusion prediction as a binary classification task; and (iii) *average Jaccard* (AJ), which jointly measures 3D point accuracy and occlusion prediction. Predicted trajectories are aligned to the ground truth using Sim(3) alignment [13, 80].

Baselines. We compare our method against recent dense 3D trackers, grouped into three categories. *(i) Iterative Dense 3D Trackers*: DELTA [56] and DELTAv2 [57], which condition on external depth and use camera poses to transform tracks into world coordinates. *(ii) Feed-forward Dense 3D Trackers Based on 3D Reconstruction Models*: St4RTrack [13], Any4D [35], and TraceAnything [49] are built upon pre-trained 3D reconstruction backbones [37, 42, 83], which are pre-trained to predict camera poses, depth, or pointmaps. *(iii) Feed-forward Dense 3D Trackers Based on Video Generative Models*: MotionCrafter [92], based on a video diffusion U-Net [3], where tracks are obtained by chaining scene flow across adjacent frames. Since *(ii)* does not output visibility, we project the predicted track pointmaps into each frame and consider a point visible if the projected pixel lies within the image bounds and its projected depth is within a 10% tolerance of the per-frame depth. We use ViPE [25] and DA3 [46] to provide input geometry for DELTA, DELTAv2, and TrackCraft3R.

Quantitative Comparison. In Tab. 1, TrackCraft3R achieves state-of-the-art performance across all benchmarks, with the best average AJ, APD_{3D}, and OA. TrackCraft3R + ViPE surpasses the strongest iterative dense 3D tracker, DELTAv2 [57] + ViPE, as well as all feed-forward baselines. TrackCraft3R + ViPE is even competitive with DELTAv2 [57] + DA3. With the stronger 3D foundation model DA3, TrackCraft3R + DA3 further surpasses DELTAv2 + DA3 and outperforms all other feed-forward baselines by a large margin. More quantitative comparisons with a dense 2D tracker [20], sparse 3D trackers [80, 88], and the concurrent work V-DPM [68] are provided in Appendix Secs. B, C, and D.

Qualitative Comparison. Fig. 4 compares 3D trajectories predicted by TrackCraft3R and DELTAv2 [57] on real-world ITTO [10] videos. TrackCraft3R produces accurate dense trajectories under large object dynamics and occlusion, where DELTAv2 often fails. Additional results on ITTO [10] and DAVIS [59] are provided in Appendix Sec. G and the supplementary video.

Robustness to Large Motion and Long Videos. We further evaluate the robustness of our method with respect to *motion* and *video length*. For *large motion*, we fix the clip length to 12 frames and increase the temporal stride s from 1 to 12 (in steps of 1), enlarging per-frame displacement. For *long videos*, we fix the stride to s=1 and increase the sequence length L from 12 to 120 (in steps of 12). The resulting APD_{3D} and AJ curves are shown in Fig. 5, averaged over sparse tracking benchmarks. TrackCraft3R consistently widens the gap with DELTAv2 [57] in both settings, indicating that the learned motion prior enables robust tracking under large displacements and generalizes to long-horizon videos beyond the training sequence length (12 frames).

5.3 Ablation Study

All ablation studies report the average AJ, APD_{3D}, and OA across all benchmarks, and use ViPE [25] for input geometry, unless otherwise specified.

Table 2: **Ablation on spatio-temporal priors.**

Initialization	AJ↑	APD_{3D}↑	OA↑
Random	0.4698	0.6312	0.8271
Pre-trained (Ours)	**0.5639**	**0.6817**	**0.9258**

Spatio-Temporal Prior in Video DiT. We compare our model against an identical architecture trained from scratch, both to convergence on the same data. Tab. 2 shows that random initialization substantially degrades all metrics, confirming that the pre-trained spatio-temporal prior is critical.

Model Design. Tab. 3 ablates four core design components by removing each from the full model. For this ablation, we train each model with the VAE frozen. *(a) w/o First-frame anchoring*: setting $\mathbf{r}_j = \mathbf{g}_j$ instead of $\mathbf{r}_j = \mathbf{g}_0$, removing reference-frame anchoring. *(b) w/o Temporal RoPE alignment*: assigning a constant temporal index t_0 to all $\mathbf{r}_j$. *(c) w/o Residual displacement*: directly regressing the 3D track pointmap $\mathbf{P}_0(t_j)$ instead of predicting residual displacements $\mathbf{\Delta}_j$. *(d) w/ VAE temporal compression*: using the original 3D VAE temporal downsampling instead of processing frames independently. All four components contribute. (a) causes consistent drops across all metrics and (b) causes the largest AJ drop, indicating that (a) and (b) jointly contribute to reference-anchored correspondence at the correct target timestamp (Fig. 3). (c) specifically drops APD_{3D}, since residual prediction stabilizes pointmap prediction. (d) drops all metrics consistently, as VAE temporal compression affects both the pointmap and visibility decoders.

Table 3: **Ablation on model components.**

Configuration	AJ↑	APD_{3D}↑	OA↑
(a) w/o First-frame anchoring	0.5135	0.6535	0.8778
(b) w/o Temporal RoPE alignment	0.4450	0.6317	0.8031
(c) w/o Residual displacement	0.5007	0.6172	0.9159
(d) w/ VAE temporal compression	0.4487	0.6325	0.8148
Full model (Ours)	**0.5609**	**0.6790**	**0.9225**

Input Geometry Quality. In Tab. 4, we study the impact of input geometry quality by using depth and camera poses from DA3 [46] and ground truth (GT). Metrics are averaged over the synthetic datasets [16, 32, 91], for which GT is available. Without any retraining, replacing DA3 with GT consistently improves all metrics, providing an upper bound for our method and suggesting that future advances in 3D geometry estimation can directly translate to better tracking performance. Note that performing tracking with input geometry from off-the-shelf estimators is becoming a common practice [56, 57, 80, 88]. In Appendix Sec. C, we further compare TrackCraft3R with recent sparse 3D trackers [80, 88] under the same input geometry.

Table 4: **Ablation on input geometry.**

Configuration	AJ↑	APD_{3D}↑	OA↑
DELTAv2 [57] + DA3 [46]	0.4384	0.5858	0.8476
TrackCraft3R + DA3 [46]	**0.6005**	**0.7144**	**0.9304**
DELTAv2 [57] + GT	0.5590	0.7169	0.8380
TrackCraft3R + GT	**0.7649**	**0.8635**	**0.9353**

LoRA Rank and VAE Finetuning. In Tab. 5, increasing the LoRA rank from 64 to 1024 consistently improves performance, indicating that the DiT benefits from more expressive low-rank updates. Unfreezing the VAEs in Stage 2 yields further gains, confirming the benefit of adapting them to the pointmap and visibility domains.

Table 5: **Ablation on LoRA rank and VAE finetuning.**

LoRA rank	VAE finetuning	AJ↑	APD_{3D}↑	OA↑
64	✗	0.5025	0.6430	0.8779
256	✗	0.5399	0.6623	0.9112
1024	✗	0.5609	0.6790	0.9225
1024	✓	**0.5639**	**0.6817**	**0.9258**

Inference Efficiency. Tab. 6 compares inference time and peak GPU memory of TrackCraft3R, DELTA [56], and DELTAv2 [57] at 448×448 resolution for 12- and 23-frame clips on a single NVIDIA A6000 GPU. For 12 frames, TrackCraft3R is **1.3× faster** and uses **4.6× less peak memory** than DELTAv2. Specifically, DELTA and DELTAv2 (1) perform iterative refinement (six steps) and (2) construct 4D correlation features between dense queries and multi-scale image features. In contrast, TrackCraft3R (1) predicts trajectories in a *single forward pass* and (2) replaces explicit 4D correlation features with full 3D attention in a $1/16$ *spatially compressed latent space*, which is effectively upsampled to pixel space with a VAE decoder. For longer sequences (*e.g.*, 23 frames), the same trend holds: all methods scale roughly linearly in runtime, while peak memory remains similar.

Table 6: **Inference efficiency** with different frame lengths.

Frames	Method	Time (s)↓	Memory (GB)↓
12	DELTA [56]	14.64	29.97
12	DELTAv2 [57]	5.00	35.46
12	**TrackCraft3R**	**3.91**	**7.63**
23	DELTA [56]	28.92	30.78
23	DELTAv2 [57]	9.70	35.90
23	**TrackCraft3R**	**7.84**	**7.63**

6 Conclusion

We presented TrackCraft3R, the first method to repurpose a video diffusion transformer as a single-pass dense 3D tracker. By introducing a dual-latent representation that couples per-frame geometry latents with first-frame-anchored track latents, together with a temporal RoPE alignment that specifies the target timestamp of each track latent, TrackCraft3R converts the per-frame generative paradigm of video DiTs into a reference-anchored dense tracking paradigm with LoRA fine-tuning. TrackCraft3R achieves state-of-the-art performance on standard 3D sparse and dense tracking benchmarks while running $1.3\times$ faster and using $4.6\times$ less peak memory than the strongest iterative 3D tracker, and further demonstrates robustness to large motions and long videos.

Acknowledgments

This research was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, RS-2024-00509279, RS-2025-II212068, RS2023-00227592, RS-2025-02214479, RS-2024-00457882, RS-2025-25441838, RS2025-02217259, RS-2026-25519202, RS-2024-00436680) and the Culture, Sports, and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism (RS-2024-00345025, RS-2024-00333068), the InnoCORE program of the Ministry of Science and ICT (N10250156) and National Research Foundation of Korea (RS-2024-00346597).

This research was supported by the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT).

References

[1] Manuel López Antequera, Pau Gargallo, Markus Hofinger, Samuel Rota Bulo, Yubin Kuang, and Peter Kontschieder. Mapillary planet-scale depth dataset. In *European Conference on Computer Vision*, pages 589–604. Springer, 2020.

[2] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024.

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[4] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv preprint arXiv:2001.10773*, 2020.

[5] Hila Chefer, Uriel Singer, Amit Zohar, Yuval Kirstain, Adam Polyak, Yaniv Taigman, Lior Wolf, and Shelly Sheynin. VideoJam: Joint appearance-motion representations for enhanced motion generation in video models. *arXiv preprint arXiv:2502.02492*, 2025.

[6] Seokju Cho, Jiahui Huang, Jisu Nam, Honggyu An, Seungryong Kim, and Joon-Young Lee. Local all-pair correspondence for point tracking. In *European conference on computer vision*, pages 306–325. Springer, 2024.

[7] Seokju Cho, Jiahui Huang, Seungryong Kim, and Joon-Young Lee. Seurat: From moving points to depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7211–7221, 2025.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

[10] Ilona Demler, Saumya Chauhan, and Georgia Gkioxari. Is this tracker on? a benchmark protocol for dynamic tracking. *arXiv preprint arXiv:2510.19819*, 2025.

[11] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-Vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

[12] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023.

[13] Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4RTrack: Simultaneous 4D reconstruction and tracking in the world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8503–8513, 2025.

[14] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. GeoWizard: Unleashing the diffusion priors for 3D geometry estimation from a single image. In *European Conference on Computer Vision*, pages 241–258. Springer, 2024.

[15] Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Yusuf Aytar, Michael Rubinstein, Chen Sun, et al. Motion prompting: Controlling video generation with motion trajectories. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1–12, 2025.

[16] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3749–3761, 2022.

[17] Jisang Han, Honggyu An, Jaewoo Jung, Takuya Narihira, Junyoung Seo, Kazumi Fukuda, Chaehyun Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. D^2USt3R: Enhancing 3D reconstruction for dynamic scenes. *arXiv preprint arXiv:2504.06264*, 2025.

[18] Jisang Han, Sunghwan Hong, Jaewoo Jung, Wooseok Jang, Honggyu An, Qianqian Wang, Seungryong Kim, and Chen Feng. Emergent outlier view rejection in visual geometry grounded transformers. *arXiv preprint arXiv:2512.04012*, 2025.

[19] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *European Conference on Computer Vision*, pages 59–75. Springer, 2022.

[20] Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Suya You, et al. AllTracker: Efficient dense point tracking at high resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5253–5262, 2025.

[21] Jing He, Haodong Li, Wei Yin, Yixun Liang, Leheng Li, Kaiqiang Zhou, Hongbo Zhang, Bingbing Liu, and Ying-Cong Chen. Lotus: Diffusion-based visual foundation model for high-quality dense prediction. *arXiv preprint arXiv:2409.18124*, 2024.

[22] Eric Hedlin, Gopal Sharma, Shweta Mahajan, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. Unsupervised semantic correspondence using stable diffusion. *Advances in Neural Information Processing Systems*, 36:8266–8279, 2023.

[23] Edward J. Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[24] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. DepthCrafter: Generating consistent long depth sequences for open-world videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2005–2015, 2025.

[25] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. ViPE: Video pose engine for 3D geometric perception. *arXiv preprint arXiv:2508.10934*, 2025.

[26] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2821–2830, 2018.

[27] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. PointWorld: Scaling 3D world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[28] Wonbong Jang, Shikun Liu, Soubhik Sanyal, Juan Camilo Perez, Kam Woh Ng, Sanskar Agrawal, Juan-Manuel Perez-Rua, Yiannis Douratsos, and Tao Xiang. Rays as pixels: Learning a joint distribution of videos and camera trajectories. *arXiv preprint arXiv:2604.09429*, 2026.

[29] Zeren Jiang, Chuanxia Zheng, Iro Laina, Diane Larlus, and Andrea Vedaldi. Geo4D: Leveraging video generators for geometric 4D scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20658–20671, 2025.

[30] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4D: Learning how things move in 3D from internet stereo videos. *arXiv preprint arXiv:2412.09621*, 2024.

[31] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE international conference on computer vision*, pages 3334–3342, 2015.

[32] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13229–13239, 2023.

[33] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *European conference on computer vision*, pages 18–35. Springer, 2024.

[34] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025.

[35] Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4D: Unified feed-forward metric 4D reconstruction. *arXiv preprint arXiv:2512.10935*, 2025.

[36] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9492–9502, 2024.

[37] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. MapAnything: Universal feed-forward metric 3D reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

[38] Jisoo Kim, Jungbin Cho, Sanghyeok Chu, Ananya Bal, Jinhyung Kim, Gunhee Lee, Sihaeng Lee, Seung Hwan Kim, Bohyung Han, Hyunmin Lee, et al. Pri4R: Learning world dynamics for vision-language-action models with privileged 4D representation. *arXiv preprint arXiv:2603.01549*, 2026.

[39] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[40] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[41] Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, Joao Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. TapVid-3D: A benchmark for tracking any point in 3D. *Advances in Neural Information Processing Systems*, 37:82149–82165, 2024.

[42] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3D with MASt3R. In *European conference on computer vision*, pages 71–91. Springer, 2024.

[43] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018.

[44] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. MegaSaM: Accurate, fast and robust structure and motion from casual dynamic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10486–10496, 2025.

[45] Yiqing Liang, Abhishek Badki, Hang Su, James Tompkin, and Orazio Gallo. Zero-shot monocular scene flow estimation in the wild. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21031–21044, 2025.

[46] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[47] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, 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*, pages 22160–22169, 2024.

[48] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[49] Xinhang Liu, Yuxi Xiao, Donny Y Chen, Jiashi Feng, Yu-Wing Tai, Chi-Keung Tang, and Bingyi Kang. Trace Anything: Representing any video in 4D via trajectory fields. *arXiv preprint arXiv:2510.13802*, 2025.

[50] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[51] Jiahao Lu, Jiayi Xu, Wenbo Hu, Ruijie Zhu, Chengfeng Zhao, Sai-Kit Yeung, Ying Shan, and Yuan Liu. Track4world: Feedforward world-centric dense 3d tracking of all pixels. In *European Conference on Computer Vision*, pages 184–204. Springer, 2026.

[52] Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4rc: 4d reconstruction via conditional querying anytime and anywhere. *arXiv preprint arXiv:2602.10094*, 2026.

[53] Jinjie Mai, Wenxuan Zhu, Haozhe Liu, Bing Li, Cheng Zheng, Jürgen Schmidhuber, and Bernard Ghanem. Can video diffusion model reconstruct 4D geometry? *arXiv preprint arXiv:2503.21082*, 2025.

[54] Jisu Nam, Gyuseong Lee, Sunwoo Kim, Hyeonsu Kim, Hyoungwon Cho, Seyeon Kim, and Seungryong Kim. Diffusion model for dense matching. *arXiv preprint arXiv:2305.19094*, 2023.

[55] Jisu Nam, Soowon Son, Dahyun Chung, Jiyoung Kim, Siyoon Jin, Junhwa Hur, and Seungryong Kim. Emergent temporal correspondences from video diffusion transformers. *arXiv preprint arXiv:2506.17220*, 2025.

[56] Tuan Duc Ngo, Peiye Zhuang, Chuang Gan, Evangelos Kalogerakis, Sergey Tulyakov, Hsin-Ying Lee, and Chaoyang Wang. DELTA: Dense efficient long-range 3D tracking for any video. *arXiv preprint arXiv:2410.24211*, 2024.

[57] Tuan Duc Ngo, Ashkan Mirzaei, Guocheng Qian, Hanwen Liang, Chuang Gan, Evangelos Kalogerakis, Peter Wonka, and Chaoyang Wang. DELTAv2: Accelerating dense 3D tracking. *arXiv preprint arXiv:2508.01170*, 2025.

[58] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng Carl Ren. Aria Digital Twin: A new benchmark dataset for egocentric 3D machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20133–20143, 2023.

[59] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

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

[61] RHOS Team. OpenD4RT: An unofficial PyTorch/GPU implementation of D4RT for 4D reconstruction and tracking. `https://github.com/Lijiaxin0111/Open-d4rt`, 2026. GitHub repository.

[62] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021.

[63] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International journal of computer vision*, 80(1):72–91, 2008.

[64] Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J. Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. *Advances in Neural Information Processing Systems*, 36:39443–39469, 2023.

[65] Jiahao Shao, Yuanbo Yang, Hongyu Zhou, Youmin Zhang, Yujun Shen, Vitor Guizilini, Yue Wang, Matteo Poggi, and Yiyi Liao. Learning temporally consistent video depth from video diffusion priors. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22841–22852, 2025.

[66] Soowon Son, Honggyu An, Chaehyun Kim, Hyunah Ko, Jisu Nam, Dahyun Chung, Siyoon Jin, Jung Yi, Jaewon Min, Junhwa Hur, et al. Repurposing video diffusion transformers for robust point tracking. *arXiv preprint arXiv:2512.20606*, 2025.

[67] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[68] Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-DPM: 4D video reconstruction with dynamic point maps. *arXiv preprint arXiv:2601.09499*, 2026.

[69] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[70] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

[71] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in neural information processing systems*, 36:1363–1389, 2023.

[72] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[73] Bo Wang, Jian Li, Yang Yu, Li Liu, Zhenping Sun, and Dewen Hu. SceneTracker: Long-term scene flow estimation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[74] 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.

[75] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[76] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20697–20709, 2024.

[77] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A dataset to push the limits of visual SLAM. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020.

[78] Hongchi Xia, Yang Fu, Sifei Liu, and Xiaolong Wang. RGBD objects in the wild: Scaling real-world 3D object learning from RGB-D videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22378–22389, 2024.

[79] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: Tracking any 2D pixels in 3D space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20406–20417, 2024.

[80] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. SpatialTrackerV2: 3D point tracking made easy. *arXiv preprint arXiv:2507.12462*, 2025.

[81] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. DynamiCrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision*, pages 399–417. Springer, 2024.

[82] Tian-Xing Xu, Xiangjun Gao, Wenbo Hu, Xiaoyu Li, Song-Hai Zhang, and Ying Shan. GeometryCrafter: Consistent geometry estimation for open-world videos with diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6632–6644, 2025.

[83] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3R: Towards 3D reconstruction of 1000+ images in one forward pass. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21924–21935, 2025.

[84] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

[85] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[86] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1790–1799, 2020.

[87] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[88] Bowei Zhang, Lei Ke, Adam W. Harley, and Katerina Fragkiadaki. TAPIP3D: Tracking any point in persistent 3D geometry. *arXiv preprint arXiv:2504.14717*, 2025.

[89] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K Barral, Raia Hadsell, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7382–7392, 2026.

[90] Hongfei Zhang, Harold Haodong Chen, Chenfei Liao, Jing He, Zixin Zhang, Haodong Li, Yihao Liang, Kanghao Chen, Bin Ren, Xu Zheng, et al. DVD: Deterministic video depth estimation with generative priors. *arXiv preprint arXiv:2603.12250*, 2026.

[91] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19855–19865, 2023.

[92] Ruijie Zhu, Jiahao Lu, Wenbo Hu, Xiaoguang Han, Jianfei Cai, Ying Shan, and Chuanxia Zheng. MotionCrafter: Dense geometry and motion reconstruction with a 4D VAE. *arXiv preprint arXiv:2602.08961*, 2026.

This appendix complements the main paper with the following:

- Sec. A: Additional details on the training dataset.
- Sec. B: Comparison with a dense 2D tracker [20].
- Sec. C: Comparison with sparse 3D trackers [80, 88].
- Sec. D: Comparison with V-DPM [68].
- Sec. E: Attention visualizations.
- Sec. F: Additional quantitative results.
- Sec. G: Additional qualitative results.
- Sec. H: Limitations and future work.

A Training Datasets

As summarized in Tab. 7, we train TrackCraft3R on four synthetic datasets: Kubric [16], DynamicReplica [32], PointOdyssey [91], and TartanAir [77]. Kubric, DynamicReplica, and PointOdyssey provide RGB, depth, camera parameters, and 3D trajectories. For Kubric [16], following [13], we render 6K sequences (480×832, 81 frames) and extract dense trajectories from the first frame. DynamicReplica and PointOdyssey provide sparse 3D trajectories from mesh vertices. TartanAir contains static scenes with large camera motion and provides RGB, depth, and camera poses. During training, we randomly sample a temporal stride from the strides listed in Tab. 7 for each dataset to cover diverse motion patterns.

Table 7: **Training data.** Number of videos and temporal strides.

Dataset	#Videos	Strides
Kubric [16]	6,042	[3, 4, 5, 6, 7]
DynamicReplica [32]	483	[5, 6, 7, 8, 9]
PointOdyssey [91]	45	[2, 3, 4, 5, 6]
TartanAir [77]	163	[1, 2, 3]

B Comparison with Lifted Dense 2D Tracker

Table 8: **3D tracking comparison with lifted AllTracker [20].** We report AJ, APD_{3D}, and OA after Sim(3) alignment. The estimated dense 2D tracks from AllTracker are lifted to 3D using ViPE [25] depth and camera poses. The best and second-best results are highlighted in dark and light blue, respectively.

	ADT [58]			PStudio [31]			DR [32]			PO [91]			Kubric [16]			Average		
Method	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
AllTracker [20] + ViPE [25]	0.6177	0.7234	0.9421	0.6463	0.8214	0.8301	0.4719	0.5979	0.9123	0.5410	0.7098	0.8601	0.2903	0.3879	0.9318	0.5134	0.6481	0.8953
TrackCraft3R + ViPE [25]	0.6683	0.7688	0.9405	0.6803	0.8163	0.8937	0.5842	0.7034	0.9408	0.5836	0.7262	0.8943	0.3032	0.3940	0.9597	0.5639	0.6817	0.9258

In Tab. 8, we further compare TrackCraft3R with AllTracker [20], a recent dense 2D tracker, on sparse and dense 3D tracking benchmarks. We use depth and camera pose from ViPE [25] to unproject the estimated 2D tracks into 3D world coordinates. TrackCraft3R consistently outperforms AllTracker, achieving higher overall AJ, APD_{3D}, and OA across all benchmarks.

C Comparison with Sparse 3D Trackers

Table 9: **3D tracking comparison with SpatialTrackerV2 [80] and TAPIP3D [88].** We report AJ, APD_{3D}, and OA after Sim(3) alignment. The best and second-best results are highlighted in dark and light blue, respectively.

	ADT [58]			PStudio [31]			DR [32]			PO [91]			Average		
Method	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
SpatialTrackerV2 [80] + ViPE [25]	0.6533	0.7818	0.9193	0.6152	0.7961	0.8142	0.4499	0.5762	0.9172	0.5023	0.7060	0.8042	0.5552	0.7150	0.8637
TAPIP3D [88] + ViPE [25]	0.6616	0.7602	0.9419	0.6426	0.8224	0.8248	0.4131	0.5925	0.7586	0.5487	0.7055	0.8698	0.5665	0.7202	0.8488
TrackCraft3R + ViPE [25]	0.6683	0.7688	0.9405	0.6803	0.8163	0.8937	0.5842	0.7034	0.9408	0.5836	0.7262	0.8943	0.6291	0.7537	0.9173

In Tab. 9, we compare TrackCraft3R with the recent sparse 3D trackers SpatialTrackerV2 [80] and TAPIP3D [88] on the sparse 3D tracking benchmarks. Note that SpatialTrackerV2 and TAPIP3D also take camera poses and depth from off-the-shelf models as input. For fair comparison, we use ViPE [25] for all methods. TrackCraft3R outperforms both SpatialTrackerV2 and TAPIP3D, achieving the best average AJ, APD_{3D}, and OA.

Table 10: **3D tracking comparison with V-DPM [68]** under the same protocol as Tab. 1 (identical frame length, datasets, and metrics). We report AJ, APD_{3D}, and OA after Sim(3) alignment. **TrackCraft3R** + V-DPM uses V-DPM's predicted pointmaps as input geometry. Since V-DPM does not predict visibility, its OA is computed with the geometric proxy of Sec. 5.2. The best and second-best results are highlighted in dark and light blue, respectively.

	ADT [58]			PStudio [31]			DR [32]			PO [91]			Kubric [16]			Average		
Method	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
V-DPM [68]	0.7461	0.8962	0.8778	0.7306	0.9135	0.8322	0.5339	0.8374	0.6852	0.6740	0.9562	0.7688	0.6688	0.8214	0.8730	0.6707	0.8849	0.8074
TrackCraft3R + DA3 [46]	0.8557	0.9457	0.9445	0.7286	0.8712	0.8891	0.6515	0.7703	0.9388	0.7060	0.8421	0.8938	0.4208	0.5047	0.9587	0.6725	0.7868	0.9250
TrackCraft3R + V-DPM [68]	0.8187	0.9374	0.9149	0.6492	0.7846	0.8923	0.7014	0.8160	0.9403	0.7867	0.9268	0.8953	0.5921	0.7007	0.9488	0.7096	0.8331	0.9183

D Comparison with V-DPM

While V-DPM [68] is a concurrent work trained on different dataset scales, we provide additional comparisons for completeness. V-DPM is also complementary to TrackCraft3R, as its predicted frame-anchored pointmaps can directly serve as our input geometry (TrackCraft3R + V-DPM). Tab. 10 reports results under the same protocol as Tab. 1. Both TrackCraft3R + DA3 [46] and TrackCraft3R + V-DPM outperform V-DPM in average AJ and OA, with TrackCraft3R + V-DPM achieving the best average AJ, while V-DPM achieves higher average APD_{3D}. Notably, the tracking stage of TrackCraft3R runs **6.6× faster** with **2.3× less memory** than V-DPM (Tab. 12), and TrackCraft3R + V-DPM remains **4.5× faster** even including V-DPM's reconstruction (Tab. 13).

Occlusion Accuracy and AJ. V-DPM does not predict visibility and relies on the geometric proxy of Sec. 5.2 for OA, whereas TrackCraft3R predicts visibility with a dedicated head. Since AJ jointly measures 3D accuracy and occlusion, this is a fundamental limitation of V-DPM as a tracker.

Dataset Scale. V-DPM relies heavily on 3D/4D supervision. Its VGGT [74] backbone is pre-trained on 17 3D-annotated datasets: Co3Dv2 [60], BlendedMVS [86], DL3DV [47], MegaDepth [43], Kubric [16], WildRGB [78], ScanNet [8], HyperSim [62], Mapillary [1], Habitat [70], Replica [32], MVS-Synth [26], PointOdyssey [91], Virtual KITTI [4], Aria Synthetic Environments [58], Aria Digital Twin [58], and an Objaverse [9]-like synthetic asset set. V-DPM then fine-tunes the backbone and heads on 6 additional datasets (ScanNet++ [87], BlendedMVS [86], Kubric-F/G [16], PointOdyssey [91], and Waymo [69]), benefiting from 23 datasets in total. In contrast, TrackCraft3R is initialized from Wan2.1-T2V [72], pre-trained on generic web images and videos with *no 3D annotations of any kind*, and fine-tuned on only 4 synthetic datasets (Kubric [16], PointOdyssey [91], Dynamic Replica [32], and TartanAir [77]). **The 3D/4D supervision seen by TrackCraft3R is thus a small fraction of that seen by V-DPM (23 *vs.* 4 datasets).**

Despite this gap, TrackCraft3R + V-DPM achieves competitive APD_{3D} while exceeding V-DPM in AJ and OA, showing that spatio-temporal priors learned from generic video data effectively compensate for the absence of dense 3D supervision. We attribute the remaining APD_{3D} gap to dataset scale: TrackCraft3R + V-DPM accesses only V-DPM's output pointmaps, whereas V-DPM's tracking head directly exploits backbone features trained on 23 datasets. This coupling, however, ties V-DPM's tracking to a fixed backbone, whereas TrackCraft3R can leverage any off-the-shelf reconstruction model and will benefit from stronger 3D foundation models in the future.

Input Pointmap Quality. To understand why TrackCraft3R + DA3 outperforms TrackCraft3R + V-DPM on some benchmarks, we measure the error of the input pointmaps against the ground truth after the same Sim(3) alignment as the tracking metric, reporting the median distance normalized by scene size (Tab. 11). On every benchmark, more accurate input geometry yields better AJ and APD_{3D}: DA3 is more accurate on ADT and PStudio, and V-DPM on DR, PO, and Kubric. Thus, TrackCraft3R faithfully reflects its input geometry and will improve as reconstruction models advance.

Table 11: **Input geometry error** of the reconstruction pointmaps from DA3 [46] and V-DPM [68].

Input geometry	ADT [58]	PStudio [31]	DR [32]	PO [91]	Kubric [16]
DA3 [46]	**4.99%**	**17.37%**	7.21%	6.01%	14.56%
V-DPM [68]	11.14%	32.86%	**5.97%**	**4.36%**	**5.27%**

Inference Efficiency. TrackCraft3R is substantially more efficient than V-DPM. As shown in Tab. 12, evaluated at 448×448 resolution on 12- and 23-frame clips using a single A6000 GPU, the efficiency

gap widens as the clip length L grows. For a 12-frame clip, TrackCraft3R runs **3.2× faster** and uses **1.7× less memory**. For a 23-frame clip, the gap widens to **6.6× faster** and **2.3× less memory**.

Table 12: **Inference efficiency** with different frame lengths.

Frames	Method	Time (s)↓	Memory (GB)↓
12	V-DPM [68]	12.32	12.60
12	**TrackCraft3R**	**3.91**	**7.63**
23	V-DPM [68]	51.49	17.69
23	**TrackCraft3R**	**7.84**	**7.63**

V-DPM predicts track pointmaps $\{P_0(t_j)\}_{j=0}^{L-1}$ via an attention-based, time-conditioned decoder invoked once per timestamp t_j. Running the decoder L times, with each call performing self-attention over all L frames, incurs $\mathcal{O}(L^2)$ time and $\mathcal{O}(L)$ memory. In contrast, TrackCraft3R predicts all trajectories in a *single feed-forward* pass within the *compressed latent space* of a video DiT. For longer clips, TrackCraft3R uses interleaved inference with a fixed clip length, yielding $\mathcal{O}(L)$ runtime and $\mathcal{O}(1)$ peak memory. This efficiency gap is decisive for long-video applications.

End-to-End Efficiency. Since V-DPM performs reconstruction and tracking jointly, while Tab. 12 reports only our tracking stage, we also measure the end-to-end cost of TrackCraft3R + V-DPM, including V-DPM's reconstruction, under the same setting. As shown in Tab. 13, TrackCraft3R remains **2.2×** and **4.5× faster** at 12 and 23 frames, respectively, at essentially identical peak memory. V-DPM's runtime is dominated by its tracking decoder (about 85%), which is invoked once per target timestamp and attends over all L frames each time, scaling as $\mathcal{O}(L^2)$. In contrast, TrackCraft3R predicts all trajectories in a single pass, so the gap widens with clip length. The peak memory is set by V-DPM's reconstruction front-end, while our decoder stays at 7.63 GB regardless of L, so the runtime gain comes at no additional memory cost.

Table 13: **End-to-end inference efficiency** including V-DPM's geometry reconstruction.

Frames	Method	Time (s)↓	Memory (GB)↓
12	V-DPM [68]	12.45	12.63
12	**TrackCraft3R** + V-DPM [68]	**5.63**	**12.61**
23	V-DPM [68]	53.08	17.75
23	**TrackCraft3R** + V-DPM [68]	**11.69**	**17.73**

Summary. TrackCraft3R trades a small amount of point accuracy for (i) data efficiency, requiring only 4 synthetic 3D/4D-annotated datasets for fine-tuning compared to V-DPM's 23 3D/4D-annotated datasets; (ii) compatibility with any 3D geometry estimator, naturally benefiting from future advances in 3D foundation models, including V-DPM itself; and (iii) substantial efficiency gains, particularly for long videos.

E Additional Attention Visualization

Temporal alignment between track and geometry latents. Fig. 6 and Fig. 7 visualize the query–key attention from a track latent $\mathbf{r}_5$ to geometry latents $\{\mathbf{g}_k\}_{k=0}^{F}$ across transformer layers. The red box marks the temporally aligned geometry latent $\mathbf{g}_5$. We observe that each track latent $\mathbf{r}_i$ assigns the highest attention to its corresponding geometry latent $\mathbf{g}_i$. We quantify this by averaging attention mass over all transformer layers: the temporally aligned geometry latent receives the highest attention (29.0% in Fig. 6 and 30.1% in Fig. 7). This verifies that temporal RoPE alignment provides a reliable signal for identifying the correct timestamp.

Correspondence within aligned latents. Fig. 8 and Fig. 9 visualize attention between $\mathbf{r}_5$ and $\mathbf{g}_5$ across transformer layers. Full 3D attention establishes reliable spatial correspondences between track and geometry latents under motion (*e.g.*, the moving baseball in Fig. 8). As discussed in prior works [55, 66], we observe layer-wise behaviors: several layers focus on RoPE-initialized positions, while a subset of layers (highlighted in red) finds correspondences between the same physical points. The same layers exhibit this behavior across different samples (Fig. 9), indicating that these layer-wise functions are consistent across inputs.

F Additional Quantitative Results

Robustness to Input Geometry from Other Reconstruction Systems. We additionally evaluate TrackCraft3R with pointmaps from VGGT-Omega [75]. As shown in Tab. 14, TrackCraft3R outperforms DELTAv2 [57] by a large margin on all three average metrics, improving average AJ from 0.527 to 0.685, APD_{3D} from 0.718 to 0.800, and OA from 0.810 to 0.921, while performing comparably on Kubric.

Table 14: **3D tracking with input geometry from VGGT-Omega [75].** Both methods take the same VGGT-Omega pointmaps as input. We report AJ, APD_{3D}, and OA after Sim(3) alignment. The better result is highlighted.

	ADT [58]			PStudio [31]			DR [32]			PO [91]			Kubric [16]			Average		
Method	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
DELTAv2 [57] + VGGT-Omega	0.6097	0.8117	0.8091	0.5526	0.8508	0.7072	0.5146	0.6995	0.7881	0.5432	0.7330	0.8046	0.4144	0.4944	0.9429	0.5269	0.7179	0.8104
TrackCraft3R + VGGT-Omega	0.8480	0.9376	0.9352	0.7258	0.8694	0.8881	0.7073	0.8225	0.9412	0.7428	0.8810	0.8940	0.4017	0.4881	0.9475	0.6851	0.7997	0.9212

Table 15: **3D tracking comparison with recent methods** under the same protocol as Tab. 1. We report AJ, APD_{3D}, and OA after Sim(3) alignment. The best and second-best results are highlighted in dark and light blue, respectively.

	ADT [58]			PStudio [31]			DR [32]			PO [91]			Kubric [16]			Average		
Method	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑	AJ↑	APD_{3D}↑	OA↑
Track4World [51]	0.7796	0.8539	0.9733	0.6460	0.8282	0.8418	0.6791	0.8043	0.8418	0.6671	0.8108	0.9009	0.4630	0.5574	0.9649	0.6470	0.7709	0.9045
4RC [52]	0.7229	0.8786	0.8728	0.7058	0.8874	0.8307	0.6184	0.8545	0.7614	0.6414	0.9049	0.7726	0.4489	0.5597	0.8730	0.6275	0.8170	0.8221
Open-D4RT [89]	0.6577	0.7824	0.9115	0.7237	0.8765	0.8749	0.5424	0.6926	0.8921	0.6636	0.8018	0.9041	0.2405	0.3108	0.9380	0.5656	0.6928	0.9041
TrackCraft3R + DA3 [46]	0.8626	0.9510	0.9445	0.7287	0.8713	0.8891	0.6518	0.7706	0.9388	0.7288	0.8678	0.8938	0.4208	0.5047	0.9587	0.6785	0.7931	0.9250

Comparison with Recent Methods. We further compare TrackCraft3R with recent 3D tracking methods, Track4World [51], 4RC [52], and D4RT [89], under our evaluation protocol. Track4World and 4RC are concurrent with our work. Since the official code of D4RT is not publicly available, we use the community implementation Open-D4RT [61]. As shown in Tab. 15, TrackCraft3R achieves the best average AJ and OA, and competitive APD_{3D}. Note that 4RC initializes its encoder and geometry decoder from DA3 [46], inheriting the large-scale geometric prior learned from 3D supervision, whereas TrackCraft3R only takes DA3 pointmaps as input.

G Additional Qualitative Results

We present additional qualitative results and comparisons with DELTAv2 [57] on ITTO [10] and DAVIS [59] videos in Figs. 10 and 11.

H Limitations and Future Work

Following the common convention in world-coordinate 3D point tracking [56, 57, 80, 88], TrackCraft3R relies on per-frame depth and camera pose from external 3D foundation models [25, 44, 46, 74]. While this design choice aligns with prior work, it also means that the accuracy of TrackCraft3R is bounded by the quality of the input geometry, as shown in Tab. 4 in the main paper. At the same time, this design allows TrackCraft3R to benefit from future advances in 3D foundation models, as improved geometry estimators can be incorporated without retraining.

A further direction is to jointly generate video and 3D tracks, unifying generation and dense 4D perception within a single video DiT. Such a unified model could serve as a strong foundation for robotic manipulation, where recent work uses generated videos and tracks as intermediate representations for action prediction [2, 27, 38, 39].

Our method may have implications when applied to real-world videos involving people, such as tracking individuals. We encourage responsible use.

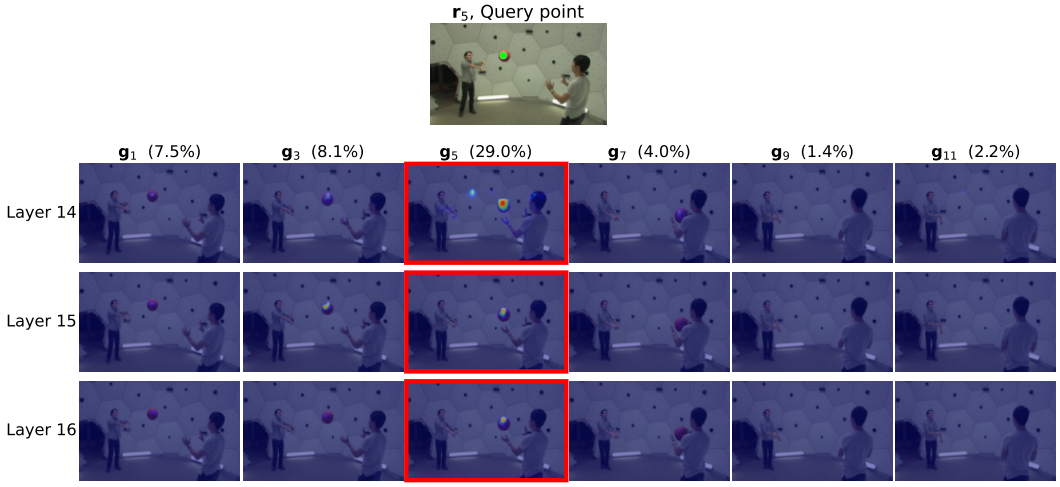


Figure 6: **Query–key attention visualization on PStudio [31].** The query point is marked with a green circle on the baseball. We visualize attention from the track latent $\mathbf{r}_5$ at timestamp t_5 to geometry latents $\{\mathbf{g}_j\}_{j=0}^{F}$ across transformer layers. Attention is concentrated on the temporally aligned geometry latent $\mathbf{g}_5$ (29.0%), showing that RoPE correctly assigns a target timestamp to each track latent.

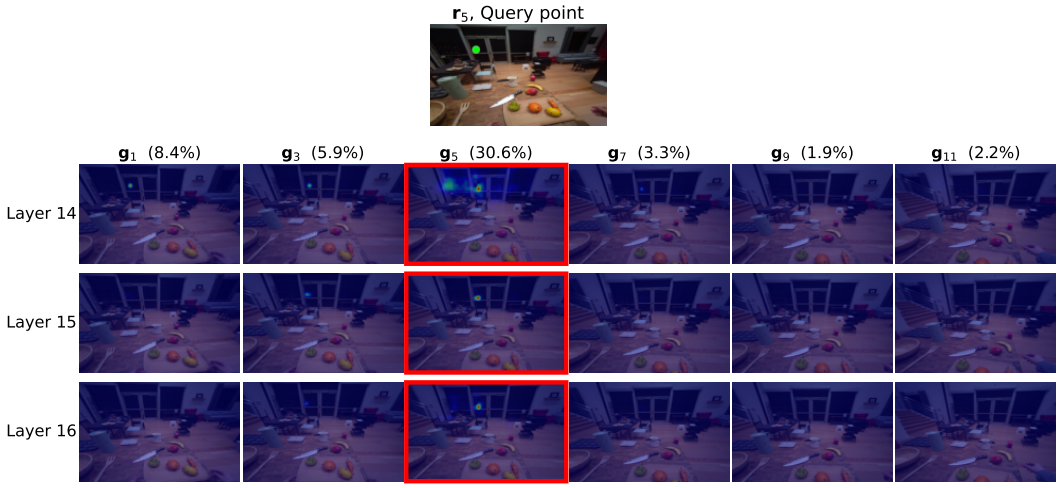


Figure 7: **Query–key attention visualization on ADT [58].** The query point is marked with a green circle on the left door. We visualize attention from the track latent $\mathbf{r}_5$ at timestamp t_5 to geometry latents $\{\mathbf{g}_j\}_{j=0}^{F}$ across transformer layers. Attention is concentrated on the temporally aligned geometry latent $\mathbf{g}_5$ (30.6%), showing that RoPE correctly assigns a target timestamp to each track latent.

s

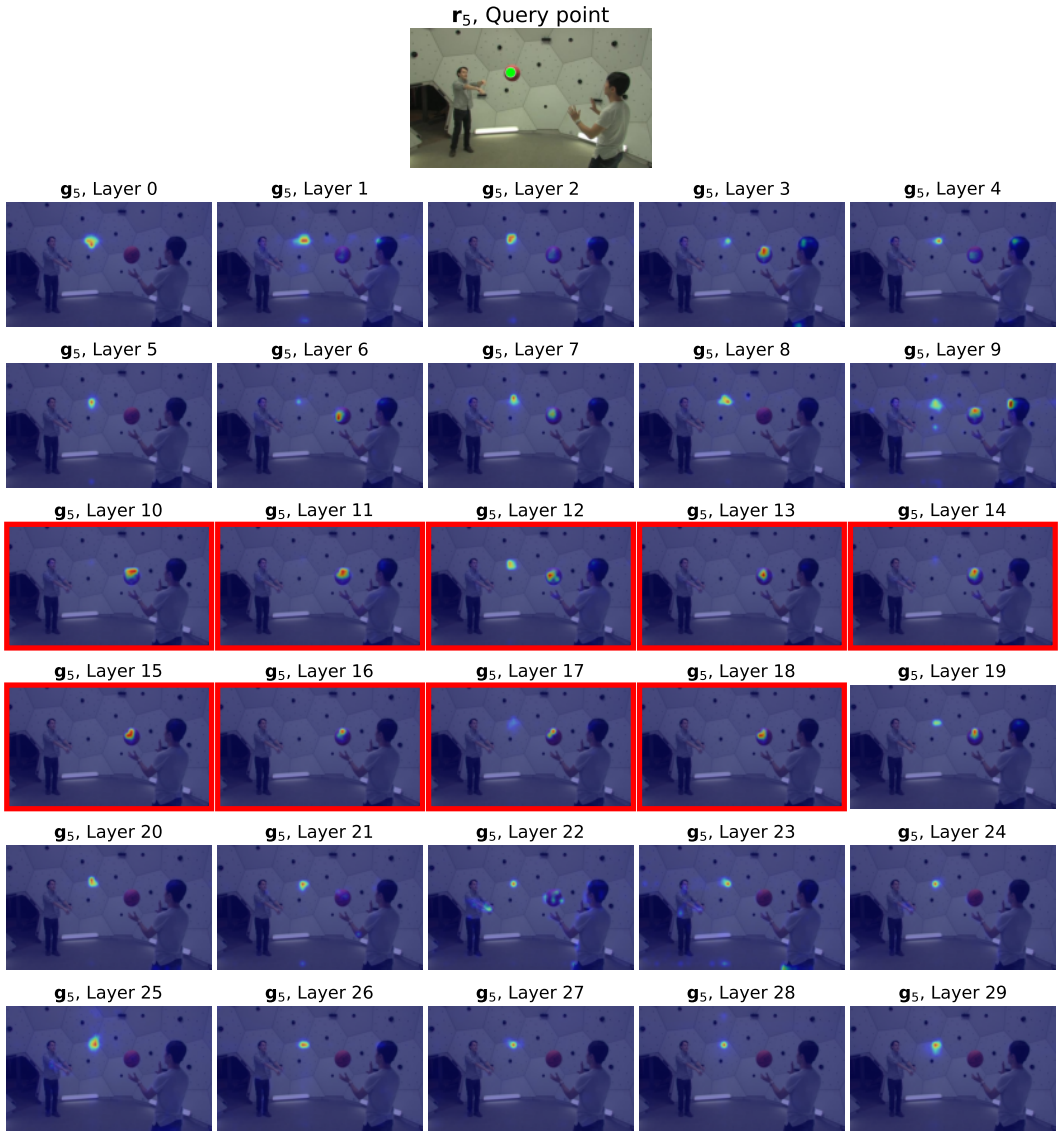


Figure 8: **Query–key attention visualization on PStudio [31].** The query point is marked with a green circle. Attention between the temporally aligned track and geometry latents identifies accurate correspondences of the same physical points in specific layers (highlighted in red boxes).

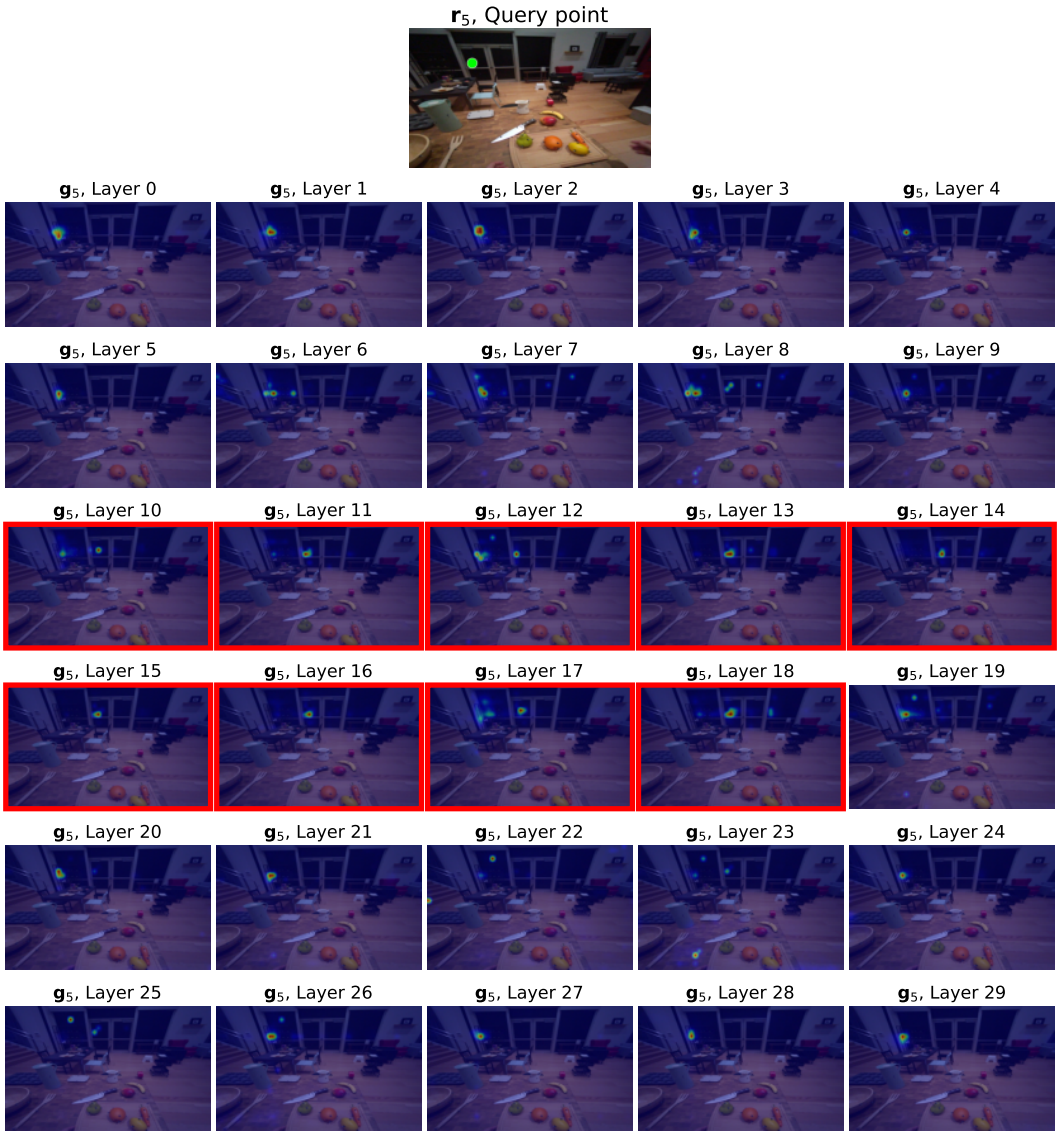


Figure 9: **Query–key attention visualization on ADT [58].** The query point is marked with a green circle. Attention between the temporally aligned track and geometry latents identifies accurate correspondences of the same physical points in specific layers (highlighted in red boxes).



Figure 10: **Qualitative results on ITTO [10] videos.** TrackCraft3R accurately estimates dense 3D trajectories on real-world videos under large camera motion, object dynamics and occlusion.

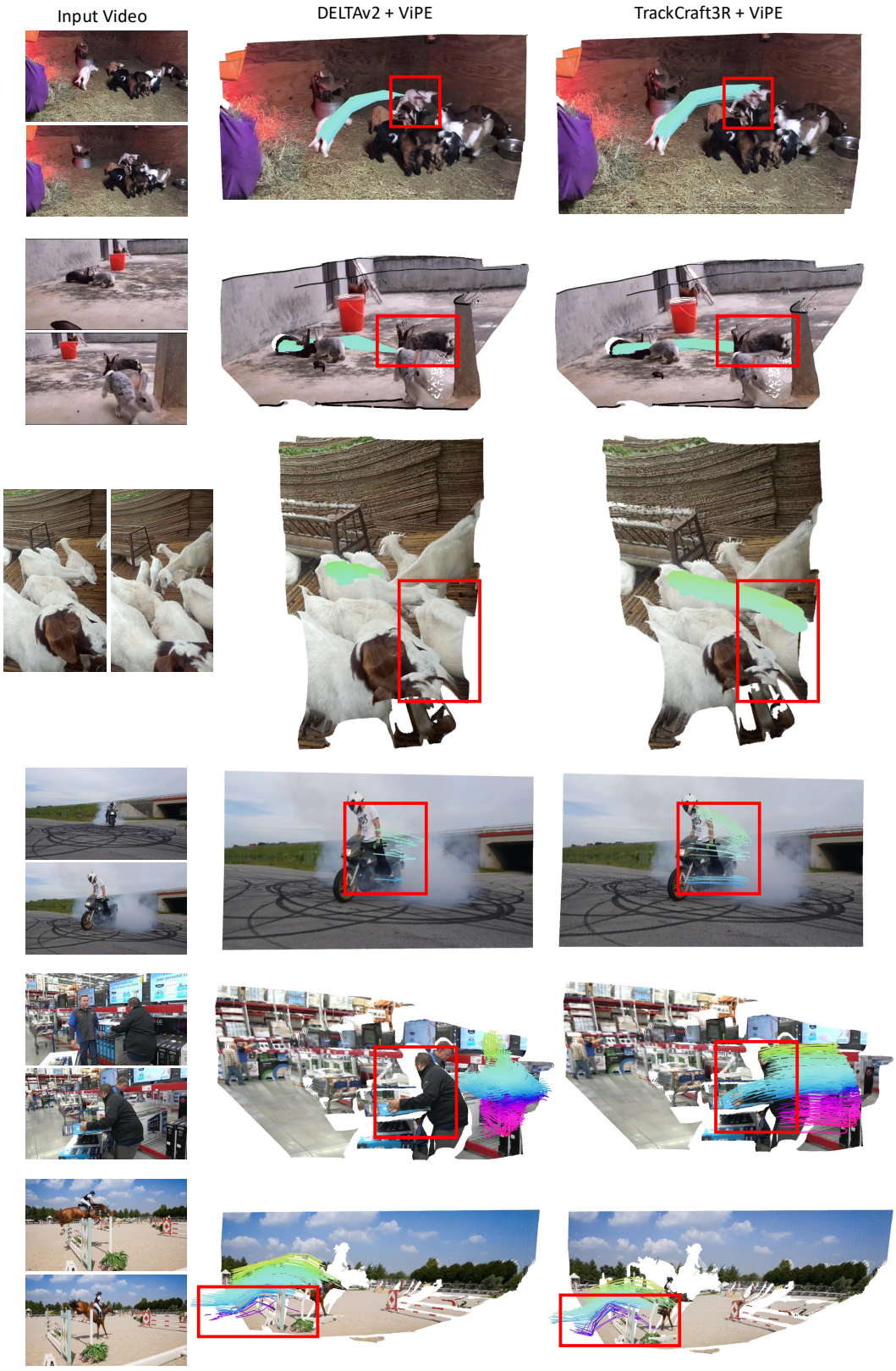


Figure 11: **Qualitative comparison on ITTO [10] and DAVIS [59] videos.** TrackCraft3R accurately estimates dense 3D trajectories on real-world videos under large camera motion, object motion, and occlusion. Note that the same query points are shared across methods.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our three contributions: (1) the first method to repurpose a video diffusion transformer for feed-forward dense 3D tracking, (2) a dual-latent representation and temporal RoPE alignment that convert frame-anchored generation into reference-anchored dense 3D tracking, and (3) state-of-the-art performance on standard 3D tracking benchmarks with robustness to large temporal strides and long videos. These claims are supported by Sec. 3 (method) and Sec. 4 (experiments, including Tab. 1 for SOTA performance, Tab. 6 for efficiency, and Fig. 4 for robustness).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations and future directions in Appendix H. Specifically, our method relies on per-frame depth and camera pose from external 3D foundation models, and its accuracy is therefore upper-bounded by the quality of the input geometry, as quantified in Tab. 4. We also discuss potential societal implications when applying our method to real-world videos involving people.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results. Our contributions are methodological (repurposing video DiTs for dense 3D tracking via dual-latent representation and temporal RoPE alignment) and empirical (state-of-the-art performance on standard benchmarks).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the information necessary to reproduce our main results. Sec. 4.2 fully describes the model architecture (dual-latent representation, temporal RoPE alignment, trajectory and visibility prediction heads). Sec. 5.1 specifies training details, including the backbone (Wan 2.1-T2V), LoRA fine-tuning, two-stage training schedule, optimizer, learning rates, batch sizes, training duration, and resolution. Sec. 5.2 describes evaluation datasets, metrics, and baselines. Appendix A further details the training datasets, including resolution, number of frames, and temporal stride sampling. We will publicly release our code and weights.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: As stated in the abstract, our code and model weights will be publicly released. All training datasets used in our work (Kubric, PointOdyssey, Dynamic Replica, TartanAir) are publicly available, and we describe their preprocessing in Appendix A. Evaluation benchmarks (TAPVid-3D ADT/PStudio, Point Odyssey, Dynamic Replica, Kubric) are also publicly available, with evaluation protocols described in Sec. 5.2.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and evaluation details are provided. Sec. 5.1 specifies the architecture, two-stage training schedule, optimizer, learning rates, batch sizes, training duration, resolution, clip length, and training objective. Evaluation datasets, splits, metrics, baselines, and alignment protocol are described in Sec. 5.2. Appendix A provides additional dataset details including temporal stride sampling.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following common practice in the dense 3D tracking literature [13, 35, 57], we report single-run results without error bars due to the high computational cost of training (5 days on 8 H200 GPUs per run). Our evaluation is performed across multiple benchmarks (ADT, PStudio, PO, DR, Kubric) covering both real-world and synthetic data, providing diverse evidence for our claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 5.1 reports the training compute: 8 NVIDIA H200 GPUs over two stages (3 days for Stage 1 and 2 days for Stage 2). Inference efficiency, including peak GPU memory and runtime on a single NVIDIA A6000 GPU at 448×448 resolution for 12- and 23-frame clips, is reported in Tab. 6 and Tab. 12.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and our research conforms with it in every respect. We use only publicly available datasets and pre-trained models, properly cite all assets, and discuss potential societal implications of tracking individuals in real-world videos in Appendix H.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential societal impacts in Appendix H. On the positive side, dense 3D tracking enables applications such as robotic manipulation, dynamic scene reconstruction, and controllable video generation, as discussed in Sec. 1. On the negative side, our method may have implications when applied to real-world videos involving people, such as tracking individuals, and we encourage responsible use.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our model is a dense 3D tracker that predicts geometric correspondences from input videos and does not generate new content (e.g., images, text, or videos). It therefore poses no high risk of misuse comparable to generative models or scraped datasets. We do not release any new dataset, and our model weights are built upon the publicly available Wan 2.1-T2V backbone.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets used in our work, including the pre-trained backbone, training datasets, evaluation benchmarks, and baseline methods, are properly cited throughout the paper. They are used in accordance with their respective licenses for academic research.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We introduce a new model (TrackCraft3R), which is documented in detail throughout the paper: architecture in Sec. 4.2, training details in Sec. 5.1, and evaluation in Sec. 5.2. Our code and model weights will be publicly released with appropriate documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing or research with human subjects. All training and evaluation data come from publicly available datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as any important, original, or non-standard component of our core methodology. Our method repurposes a video diffusion transformer (Wan 2.1-T2V) for 3D tracking, which is not a language model.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.