TRACK4D: Representing Dense 3D Tracking for Video Diffusion Models

Yushi Lan[1] **Zeren Jiang**[1] **Kelvin Li**[1]
Xingang Pan[2] **Chuanxia Zheng**[2] **Andrea Vedaldi**[1]

[1]Visual Geometry Group, University of Oxford [2]Nanyang Technological University

{yushi,zeren,kelvin,vedaldi}@robots.ox.ac.uk {xingang.pan,chuanxia.zheng}@ntu.edu.sg

https://nirvanalan.github.io/projects/track4d

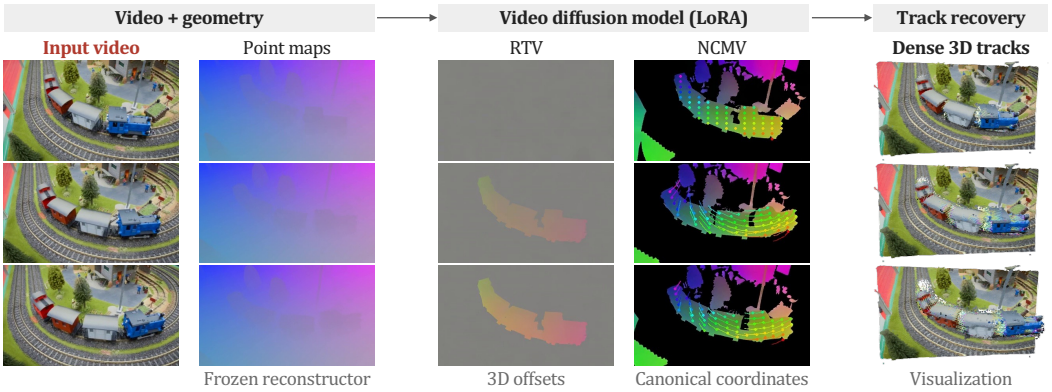


Figure 1: **TRACK4D formulates dense 3D tracking as conditional video generation.** From a monocular video and point maps of a frozen 3D reconstructor, a LoRA-adapted video diffusion model generates a Residual 3D Tracking Video (metric 3D offsets) and a Normalized Coordinate Map Video (canonical coordinates), from which dense 3D tracks are recovered without per-scene optimization.

Abstract

We study how to repurpose video diffusion models (VDMs), pretrained on millions of videos, for dense 3D point tracking, where training data is scarce. Doing so is non-trivial, as it requires representing 3D tracks as video-like signals that may lie outside the natural-video distribution on which the VDM was trained. We thus systematically assess the effectiveness of two such video representations. The first, Residual 3D Tracking Video (RTV), encodes per-pixel 3D offsets relative to a query frame and allows recovery of 3D trajectories directly from the generated video. The second, Normalized Coordinate Map Video (NCMV), labels each pixel with a canonical coordinate invariant through time. This is closer to a natural video and easier for the VDM to learn, but requires additional post-processing with an off-the-shelf 2D tracker to recover 3D tracks. RTV tracks more accurately, while NCMV shows that a VDM can generate dense correspondence as a video. We instantiate both representations with LoRA adaptation of a frozen Wan backbone, conditioned on point maps from an off-the-shelf 3D reconstructor as a geometric scaffold. Trained with a single dynamic 3D tracking dataset, TRACK4D achieves competitive zero-shot tracking on four real and synthetic benchmarks. A regression readout that bypasses the video VAE and uses additional training data brings further improvement, demonstrating the potential of this direction for dense 3D tracking. We hope our analysis will inspire future work that leverages VDM priors for dense 3D tracking and 4D reconstruction. More details can be found on our project page.

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

1 Introduction

We consider the problem of *dense monocular 4D reconstruction*, i.e., recovering the 3D shape and motion of every visible surface point in a video. 4D reconstruction is propaedeutic to understanding the physical world and supports applications such as video editing [1], spatial reasoning [2] and robotics [3, 4]. Traditional 4D reconstruction relied on per-scene optimization [5–8], which is costly and brittle as dynamic reconstruction needs strong *statistical priors* [5]. Feed-forward networks now excel at *static* 3D reconstruction [9–15]; for *dynamics*, sparse 3D point trackers [16–18] scale poorly to dense outputs, and dense 4D methods [19–24] are bottlenecked by the scarcity of video with *dense 3D tracking annotations*.

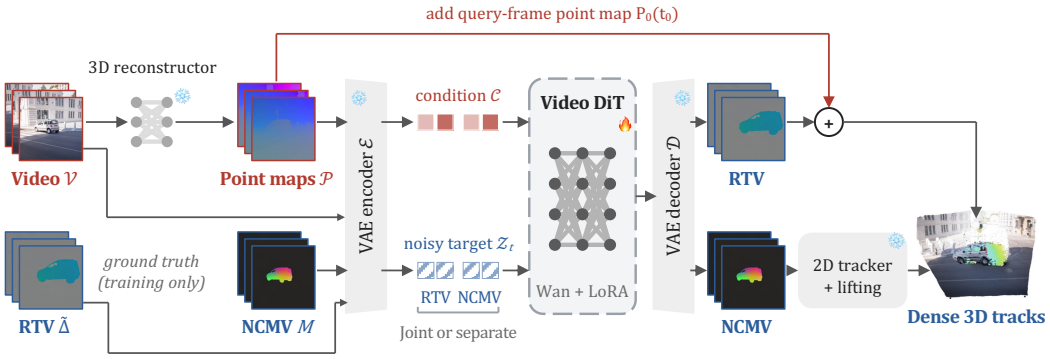
Here we propose to address the scarcity of training data by leveraging Video Diffusion Models (VDMs) [25, 26] pretrained on millions of unlabelled videos. VDMs have already been shown to transfer well to depth prediction [27], dynamic 3D reconstruction [28], and short-term correspondences [29]. How to extend VDMs to 3D tracking is, however, non-trivial, as one must represent the 3D tracks as a video-like signal, representing at every pixel the location of the corresponding 3D point in world space. While this signal may be in the *form* of a video, i.e., a spatio-temporal map, it *looks nothing like a natural video*. How to adapt the latent space of the VDM to represent such a signal is the key engineering problem. In this paper, we explore this design choice systematically.

We note that there are two fairly different ways to represent 3D tracks as a video (Figure 1). The first is a *Residual 3D Tracking Video* (RTV), where each pixel in a given query frame is mapped over time to different 3D displacement vectors (offsets). By looking at how the value associated to a pixel varies over time, one recovers directly the 3D trajectory of the corresponding physical point, tracking its motion forward. However, as shown in Figure 1, the resulting video is unnatural as motion is encoded as an intensity change rather than a spatial shift. The other approach is to use instead a *Normalized Coordinate Map Video* (NCMV), which associates each pixel to the 3D coordinate of the corresponding canonical physical point at a fixed time and spatial reference frame. These canonical coordinates assign to each physical point a unique 'color', consistent over time. The resulting video is more natural, as motion is encoded as a spatial shift of these colors, and is thus easier for the VDM to generate. However, recovering the forward 3D trajectories requires further 2D tracking.

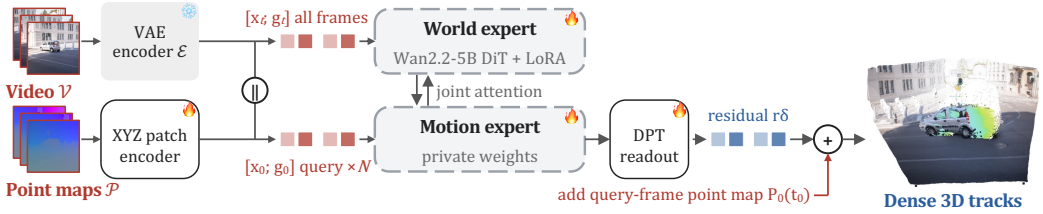
We explore both representations, using LoRA to adapt a frozen Wan [26] VDM backbone for 3D tracking (Figure 4). We condition the generator on the output of an off-the-shelf 3D reconstructor (*e.g.* VGGT [11], STream3R [13], DAv3 [14]), providing a geometric scaffold to the model, and learn to decode 3D tracks by fine-tuning the VDM. Since only LoRA adapters are trained, TRACK4D directly benefits from advances in video generation, and even the 14B backbone can be fine-tuned with academic resources.

Encoding tracks as a video requires a target at every pixel, which restricts training to a single dense dynamic 3D tracking dataset (Kubric MOVi-F [30]). Nevertheless, TRACK4D achieves competitive zero-shot performance on four tracking benchmarks and, on the held-out dense Kubric split, outperforms St4RTrack [19] and TraceAnything [31] by a clear margin in dense APD/EPE. As for the best representation, we find that RTV is simpler and results in better forward tracking performance, despite arguably being harder to learn. Still, NCMV shows that a VDM can generate dense correspondence directly as a video output, whereas prior work only reads sparse 2D tracks from *frozen* VDM features [32, 33]. This suggests that video generators carry enough structure to produce dense correspondence, even if NCMV is not the most accurate way to read out tracks.

To summarise, we make three contributions. First, we formulate dense 3D tracking as conditional video generation in a pretrained VDM, distinct from prior work [32, 33] that reads sparse 2D correspondences from *frozen* VDM features. This allows us to *sample* trajectories from the VDM's generative prior instead of reading them out deterministically; although VDMs denoise iteratively, few-step denoising retains similar accuracy at a lower cost (Table 4). Second, we systematically study how to represent 3D tracks inside a VDM, comparing RTV (explicit 3D offsets) and NCMV (implicit canonical-coordinate correspondence) within a single model: RTV is the recommended output, while NCMV serves as a probe of VDM correspondence priors and as an auxiliary training objective, since the joint checkpoint slightly improves RTV (Appendix G.4). Third, we study a VAE-free regression readout that admits sparse supervision and improves further with more data. Since it learns a residual on the input point map, its accuracy is limited by that geometry, which suggests training geometry and motion jointly [22, 34], left to future work (Section 4.5).



(a) Generative readout (RTV and NCMV)



(b) Regression readout (Track4D-Reg)

Figure 2: **TRACK4D overview. (a) Generative readout.** A frozen reconstructor supplies point maps $\mathcal{P}$; the video VAE encodes them with video $\mathcal{V}$ into condition $\mathcal{C}$. The LoRA-adapted video DiT denoises the RTV and NCMV latents, jointly by default, and the VAE decodes the videos. Hatched targets are noised in training and start from Gaussian noise at inference. Adding inverse-normalized RTV offsets to the query-frame point map gives 3D tracks; NCMV requires 2D tracking and lifting. **(b) Regression readout.** Point maps enter through a patch encoder, a private motion expert attends jointly with the video DiT, and a DPT head reads residuals without the output VAE (Section 3.4 and Figure 8). Flames denote trained modules, snowflakes frozen modules.

2 Related work

Feed-forward 3D and 4D reconstruction. Predicting dense 3D geometry in a single forward pass has progressed from pairwise point map regression [9] replacing classical pipelines [6, 35–37], to scalable static multi-view models [11, 14, 38, 15] and dynamic sequential methods [10, 13, 39, 40]. St4RTrack [19] and DPM [21] pioneered dense 4D reconstruction by augmenting DUSt3R-style architectures with motion prediction; subsequent work [31, 22–24, 41–43, 34] further advances this direction. Because these models build on 3D reconstructors trained on static data, extending them to 4D requires tracking supervision, which remains scarce [30, 44, 45]. These deterministic regressors optimize geometry and motion directly and can also supply our geometric scaffold (Appendix G.10). TRACK4D takes a complementary route by adapting priors already encoded in a pretrained VDM.

Dense 3D point tracking. Temporal correspondence has been studied through optical flow [46–48] and long-range 2D point tracking [49–53]. Lifting 2D tracks to 3D usually requires a separate depth model: SpatialTracker [16, 17] and TAPIP3D [18] follow this two-stage strategy but inherit errors from each stage. DELTA [54, 55] trains end-to-end dense 3D trackers with sparse-to-dense upsampling but require curated tracking supervision. The generative TRACK4D differs in two ways: it produces dense 3D tracks directly as VDM output rather than from a supervised 3D-tracking head, and it combines pretrained motion priors with limited tracking supervision. The introduced representation RTV is closely related to neural scene-flow representations [56, 7], but is dense, in metric world coordinates, and generated through iterative denoising without per-pair optimization.

Repurposing video diffusion models for geometric tasks. Pretrained video generators [57–59, 26, 25, 60] encode powerful spatio-temporal priors. On the image side, Marigold [27] and follow-ups [61] fine-tune diffusion models for depth. For dynamic scenes, Geo4D [28] predicts per-frame point maps, 4DNex [62] adopts width-concatenated multi-modal latents, and VIST3A [63] and Gen3R [64] couple a VGGT backbone with a VDM for joint RGB and 3D generation. However, these produce *time-variant* geometry [22] without dense correspondences across time. MotionCrafter [29] extends

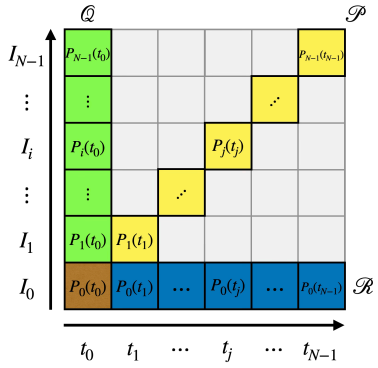


Figure 3: **Prediction formulations** (adapted from V-DPM [22]). Each entry $P_i(t_j)$ (short for $P_i(t_j, \pi_0)$, with the reference frame π_0 fixed) gives the positions at time t_j of the surface points observed in frame I_i. $\mathcal{P}$ (diagonal): geometry at each frame's capture time. $\mathcal{Q}$ (column): time-invariant prediction [22], referring every frame I_i back to the reference time t_0. $\mathcal{R}$ (row): dense 3D tracking, propagating the query-frame point map P_0 across all timestamps. TRACK4D encodes $\mathcal{R}$ explicitly as 3D offsets (RTV) and $\mathcal{Q}$ as normalized canonical colors (NCMV), from which $\mathcal{R}$ is recovered by 2D tracking.

Geo4D to predict scene flow but only between neighboring frames, and MoMaps [65] generates per-pixel 3D trajectories (our $\mathcal{R}$) from a single image with an image diffusion model. The concurrent TrackCraft3R [66] adapts a video DiT with LoRA to predict reference-anchored 3D track residuals; it retains the output VAE, whereas our regression readout removes it (Sections 3.4 and 4.5).

A second line of work observes that VDMs already *contain* 2D tracking signal in their internal features: DIFT [32] extracts semantic correspondence from image diffusion features; DiffTrack [33] extends this to video DiTs; Track4Gen [67] uses these features as an auxiliary signal to improve generated-video consistency; and subsequent work [68, 69] achieves competitive sparse 2D tracking by reading from frozen VDM features. TRACK4D differs by *generating* dense 3D tracking videos as model output, rather than reading sparse 2D tracks from frozen features — giving, to our knowledge, an early systematic study of how to represent dense 3D tracking inside a pretrained VDM.

3 Method

Given a monocular video $\mathcal{V} \in \mathbb{R}^{N\times H\times W\times 3}$, TRACK4D recovers dense 3D point trajectories in a fixed reference frame. We assume that camera parameters and depth are available from an off-the-shelf reconstructor and focus on the tracking problem.

3.1 Video-like representations of dynamics

Let $\mathcal{V} = \{I_i\}_{i=0}^{N-1}$ be a sequence of N RGB frames captured at times t_i from cameras with extrinsics $\pi_i \in \mathbb{SE}(3)$, and u a pixel location. A *Dynamic Point Map* [22] $P_i(t, \pi)(u)$ assigns to each pixel u in I_i its 3D location at time t expressed in reference frame π. We fix $\pi = \pi_0$ and abbreviate $P_i(t)$ for $P_i(t, \pi_0)$, so 3D coordinates are viewpoint-invariant.

As i, j vary from 0 to $N-1$, the point maps $P_i(t_j)$ form a 2D grid (Figure 3). The diagonal

$$\mathcal{P} = \big(P_0(t_0)\,,\ P_1(t_1)\,,\ \ldots,\ P_{N-1}(t_{N-1})\big) \tag{1}$$

records the 3D position of pixels at their capture time and is recoverable from depth and camera parameters alone. Tracking information lives in the off-diagonal entries:

$$\mathcal{Q} = \big(P_0(t_0)\,,\ P_1(t_0)\,,\ \ldots,\ P_{N-1}(t_0)\big), \quad \mathcal{R} = \big(P_0(t_0)\,,\ P_0(t_1)\,,\ \ldots,\ P_0(t_{N-1})\big). \tag{2}$$

$\mathcal{Q}$ is a collection of *time-invariant* point maps: the 3D coordinates of each pixel referred back to the fixed reference time t_0, undoing scene motion (the V-DPM representation [22, 21]). As a video, $\mathcal{Q}$ is spatially aligned to $\mathcal{V}$ frame-by-frame. $\mathcal{R}$ is dual: a collection of *time-dependent* point maps that forward pixels of the reference frame I_0 to their 3D location at later times t_j — as a video, this is spatially aligned with I_0 but color-varying [52, 23, 53]; MoMaps [65] calls this image of per-pixel 3D trajectories a *motion map* and generates it from a single image.

3.2 TRACK4D

Given $\mathcal{V}$ and the diagonal point maps $\mathcal{P}$ from an off-the-shelf 3D reconstructor (geometric scaffold), we sample row $\mathcal{R}$ and column $\mathcal{Q}$ from the conditional distribution

$$\mathbb{P}\Big(\mathcal{R}, \mathcal{Q} \,\Big|\, \mathcal{V}, \mathcal{P}\Big), \tag{3}$$



Figure 4: **Two video representations of dense 3D tracking on DAVIS [70].** Rows show RGB, RTV offsets, NCMV canonical coordinates, and recovered XYZ tracks. Adding inverse-normalized RTV offsets to the query-frame point map recovers the tracks. All predictions are TRACK4D outputs.

using a pretrained video diffusion model (VDM). $\mathcal{R}$ encodes full 3D tracks directly; $\mathcal{Q}$ does so indirectly but is spatially aligned with $\mathcal{V}$ and resembles natural video.

Point-map values lie outside the $[-1, 1]$ range expected by the video VAE. A naïve global normalization to $[-1, 1]$ [28, 62] works for $\mathcal{Q}$ but wastes dynamic range for $\mathcal{R}$, since dynamic regions occupy only a small portion of the scene. Following [29] we instead represent $\mathcal{R}$ as a 3D offset:

$$P_i(t_j) = P_i(t_0) + \mathbf{\Delta}_i^j, \tag{4}$$

where $\mathbf{\Delta}_i^j$ denotes the deformation vector $(\Delta x, \Delta y, \Delta z)$ of the point map $P_i(t_j)$ relative to the query frame $P_i(t_0)$ (the resulting tracks are in Figure 4, row 4). We define this offset across arbitrary timestamps, not just neighboring ones as in [29]. To further improve sensitivity to small displacements, we predict offsets in log space [14, 61]:

$$\tilde{\mathbf{\Delta}}_i^j = \operatorname{sign}(\mathbf{\Delta}_i^j) \cdot \log\big(|\mathbf{\Delta}_i^j| + 1\big) \cdot K. \tag{5}$$

With this transform, static regions with zero deformation remain exactly zero. We choose a scalar K so that $\tilde{\mathbf{\Delta}}_i^j \in [-1, 1]$, where K is computed from the training set statistics. We thus encode $\tilde{\mathbf{\Delta}}$, which we call Residual 3D Tracking Video (RTV) (Figure 4, row 2), with our autoencoder in lieu of $\mathcal{R}$.

As for $\mathcal{Q}$, we map it to the range $[-1, 1]$ linearly, and call the output a Normalized Coordinate Map Video (NCMV) [71, 72] (Figure 4, row 3). Specifically, we define a normalization function $\phi : \mathbb{R}^3 \to [-1, 1]^3$ that maps the 3D coordinates of the foreground objects to the unit cube:

$$\phi(\mathbf{p}) = 2\,\bar{\mathbf{p}} - \mathbf{1}, \quad \text{where} \quad \bar{\mathbf{p}} = \frac{\mathbf{p} - \mathbf{p}_{\min}}{\mathbf{p}_{\max} - \mathbf{p}_{\min}}. \tag{6}$$

Here $\mathbf{p} \in P_0(t_0)$ is a 3D point extracted from the point map $P_0(t_0)$ for the query frame at $t = 0$, and $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ are the computed minimum and maximum values from the point map, respectively. Using ϕ, we define the NCMV to be $M = \Phi(\mathcal{Q})$, where Φ is applied component-wise. Intuitively, this amounts to 'colorizing' each pixel of the reference image I_0 with a normalized version of the corresponding physical point coordinates

$$M_0(u) = \phi\big(P_0(t_0)\,(u)\big). \tag{7}$$

The value $M_0(u)$ can also be thought of as a unique identifier assigned to that pixel. Subsequent frames $j > 0$ are colorized in a similar manner. The end result is that each pixel u' at time t_j receives the same 3D point/color/identifier assigned to its counterpart u at time t_0, where the correspondence (u, u') is given by the 3D tracking:

$$M_j(u') = M_0(u), \quad \text{where } u' \text{ is the projection of } P_0(t_j)\,(u) \text{ into camera } \pi_j. \tag{8}$$

By construction, pixels corresponding to the same 3D surface point share a consistent and unique color throughout the video sequence. We expect this representation to be easier for the VDM to learn, consistent with recent findings that generative models implicitly capture dense correspondences [73, 74, 32].

Recovering 3D tracking from NCMV. In principle, we can decode NCMV colors into tracks by matching points by their unique canonical 'color' across frames, but this performs poorly due to the finite distinctiveness and accuracy of the signal (Section 4.4). Instead, we run CoTracker3 [75] on M to obtain dense 2D correspondences and lift them to 3D using VGGT [11] on the original video $\mathcal{V}$.

Video diffusion model. We adopt Wan [26] as our backbone: its 3D VAE encodes videos into spatio-temporal latents and the DiT [76] applies full attention over all latent tokens, in contrast to factorized spatial-temporal attention [77]. Text conditioning uses umT5 [78].

Both RTV and NCMV are 3-channel videos and use the unmodified Wan VAE; we add no new output heads. Adaptation uses LoRA [79] on the DiT attention/MLP linears (rank 64), with VAE and patch embedder frozen — see Appendix D for full layer-level details.

3.3 Generating multimodal 3D tracking

TRACK4D jointly generates RTV and NCMV in one iterative sampling process, which lets us compare the representations head-to-head within a shared architecture. RTV is the output we recommend for deployment; NCMV is retained as an auxiliary representation and training objective.

We extend an I2V VDM to jointly generate two 3-channel videos, $\{\tilde{\mathbf{\Delta}}_0^j\}_{j=0}^{N-1}$ (RTV) and $\{M_j\}_{j=0}^{N-1}$ (NCMV), conditioned on $\mathcal{V}$ and $\mathcal{P}$. All modalities are first encoded into the latent space via the VAE encoder $\mathcal{E}(\cdot)$, and concatenated along the width dimension to form unified latent representations:

$$\mathcal{Z}_1 = [x_{\tilde{\mathbf{\Delta}}};\ x_M], \quad \mathcal{C} = [x_{\mathcal{V}};\ x_{\mathcal{P}}], \quad \mathcal{Z}_1, \mathcal{C} \in \mathbb{R}^{n \times h \times 2w \times c}, \tag{9}$$

where $x_* = \mathcal{E}(*)$, $\mathcal{Z}_1$ denotes the clean generation target, $\mathcal{C}$ denotes the conditioning latents (kept clean throughout training), $[\cdot\,;\,\cdot]$ indicates concatenation along the width dimension [62, 64], and n, h, w, c denote the spatio-temporal dimensions after the video VAE $\mathcal{E}$. We adopt width-wise concatenation to allow the model to flexibly predict non-pixel-aligned outputs (Figure 4, row 2), whereas channel-wise concatenation is better suited to input-output aligned predictions [80–82].

Training follows Rectified Flow [83, 84]: with $\mathcal{Z}_0 \sim \mathcal{N}(0, I)$, $t \sim \mathcal{U}(0, 1)$, and $\mathcal{Z}_t = t\mathcal{Z}_1 + (1-t)\mathcal{Z}_0$, we regress the velocity $v_t = \mathcal{Z}_1 - \mathcal{Z}_0$:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\mathcal{Z}_0, \mathcal{Z}_1, \mathcal{C}, t} \left\| u_{\boldsymbol{\theta}}([\mathcal{Z}_t; \mathcal{C}], t) - v_t \right\|_2^2, \quad [\mathcal{Z}_t; \mathcal{C}] \in \mathbb{R}^{n \times h \times 4w \times c}. \tag{10}$$

Four learnable domain embeddings differentiate the modalities, following [62].

3.4 Regression readout without the output VAE

The generative readout decodes tracking through the video VAE, which blurs depth discontinuities (Appendix G.8) and needs a target at every pixel. Our third readout keeps the VDM prior and the residual target of Equation (4) but *regresses* the residual from the VDM features: without a VAE on the output path, the loss applies only at annotated pixels, so sparse tracking datasets can be used. Following TrackCraft3R [66], the model is anchored at the query frame: for every pixel u of I_0 and time t_j it predicts a normalized residual $\boldsymbol{\delta}_j(u)$ and composes

$$\hat{P}_0(t_j)(u) = P_0(t_0)\,(u) + r\,\boldsymbol{\delta}_j(u), \tag{11}$$

with r the per-clip normalization radius of $\mathcal{P}$. In Figure 2b, the point maps enter through a small patch encoder rather than the VAE, so every frame is the pair $[x_{\mathcal{V}};\ g_{\mathcal{P}}]$ of its RGB latent and geometry features. The LoRA-adapted Wan2.2-TI2V-5B [26] *world expert* [85] and a private *motion expert* of equal depth process all-frame and repeated query-frame tokens, respectively, with joint attention at every layer [86]. A DPT head [87, 15] on four motion-expert layers outputs the residual and a visibility logit in a single forward pass, without noise or denoising. TrackCraft3R instead retains the output VAE. Architecture and training details are in Appendix G.6 and Figure 8.

4 Experiments

4.1 Settings

Datasets. In its generative form, TRACK4D requires video sequences with *dense* 3D tracking annotations, which are scarce in the real world, so we train only on synthetic data; the regression

readout (Section 3.4, Appendix G.5) relaxes this to sparse supervision. We re-render Kubric [30] into $5{,}600$ clips of 49 frames with dense 3D tracks computed following Delta3D [54]. We additionally include TartanAir [88], a high-quality synthetic dataset with negligible object motion, with tracking ground truth set to the static query-frame point map. We evaluate on five benchmarks: PointOdyssey (PO) [44] (synthetic, complex motion); Aria Digital Twin (ADT) [89] (real-world egocentric with ground-truth 3D); Panoptic Studio (PStudio) [90] (real-world multi-person scenes); Dynamic Replica (DR) [45] (synthetic indoor with dynamic objects); and a held-out Kubric test split disjoint from training. We use 50 sequences per benchmark and their first 49 frames (49 ADT sequences in the regression replays, Appendix G.6). The first four have only *sparse* annotations. Our generative models are zero-shot on all four, unlike prior models [19, 31, 23, 22] trained on one or more, and TRACK4D-Reg, trained on PointOdyssey and Dynamic Replica (Section 4.5).

Metrics. Following St4RTrack [19] and TAPVid-3D [91], we report *Average Percent of Points within Delta* ($\text{APD}_{3\text{D}}$) at $\delta_{3\text{D}} \in \{0.1, 0.3, 0.5, 1.0\}$ m over 49 frames, and *End-Point Error* (EPE) after a global $\mathrm{Sim}(3)$ alignment of the predicted tracks to the ground truth [92], as in 4RC [34]; all results in Table 1 are $\mathrm{Sim}(3)$ aligned. The generative readouts do not predict visibility; we report APD/EPE for all methods, including TRACK4D-Reg.

Baselines and comparison protocol. We compare TRACK4D against four groups of methods that produce 3D trajectories: **(i)** 2D-tracker pipelines that lift 2D tracks to 3D via a depth backbone — SpatialTrackerV2 [17] and CoTracker3* [75] (lifted by VGGT [11]); **(ii)** dense 4D reconstruction methods — St4RTrack [19] and TraceAnything [31]; **(iii)** concurrent methods built on stronger 3D foundation backbones — Any4D [23] and V-DPM [22]; and **(iv)** TrackCraft3R [66], replayed with the same STream3R geometry. DELTA [54, 55] and TAPIP3D [18] use incompatible conditioning interfaces (no point-map input) and are excluded from this comparison. Each baseline runs in its native protocol with its own depth/camera estimator as released; TRACK4D uses STream3R [13] or DAv3 [14] as the scaffold, all evaluated under the D4RT [24] protocol; this is a *protocol* comparison rather than an isolation of the conditioning factor.

Implementation and inference cost. The joint model of Table 1 trains a rank-64 LoRA [79] on the official Wan [26] codebase, continuing from an RTV-only adapter for $5{,}000$ iterations on 32 GH200 GPUs (about one day; Appendix D).

Inference uses 50 denoising steps with random latent noise (0–400 steps) on the input video latent [93] to reduce the synthetic-to-real gap. In the 5B sampling ablation, 50, 10 and 5 steps take 14, 9.6 and 8.6 s per clip on one A6000 with similar EPE (Appendix F).

4.2 Qualitative evaluation

Figure 5 visualizes zero-shot 3D reconstruction and tracking on DAVIS [70]; conditioning point maps $\mathcal{P}$ come from STream3R [13] or DAv3 [14]. TRACK4D produces temporally coherent dense trajectories that follow camera-induced parallax and dynamic object motion. Selected cases highlight dynamic-object failures; broader DAVIS results appear in Appendix H, and Figure 6 compares all three readouts with TrackCraft3R.

4.3 Quantitative evaluation

Results overview. Table 1 reports 3D tracking on the four benchmarks with sparse annotations for **(a)** RTV (TRACK4D-RTV) and **(b)** NCMV (TRACK4D-NCMV) from the joint model, and **(c)** the regression readout TRACK4D-Reg (Section 4.5) with the same scaffold.

Dense Kubric. On the held-out dense Kubric split (Table 5) — the only benchmark as dense as our training signal — TRACK4D-RTV (0.336 APD / 1.079 EPE) outperforms St4RTrack (0.047 / 5.333) and TraceAnything (0.288 / 1.550) by a clear margin; on dynamic points V-DPM is best.

Comparison with feed-forward 3D tracking. SpatialTrackerV2 [17], St4RTrack [19], Any4D [23] and V-DPM [22] are trained on splits of one or more evaluation datasets (*e.g.*, PointOdyssey, Dynamic Replica), whereas the generative TRACK4D has dynamic tracking labels only from Kubric and is zero-shot on the four external benchmarks. TRACK4D-RTV is nevertheless competitive: on ADT it ranks first in APD on both All (0.892) and Dyn (0.954), and on PStudio and DR it is comparable to or

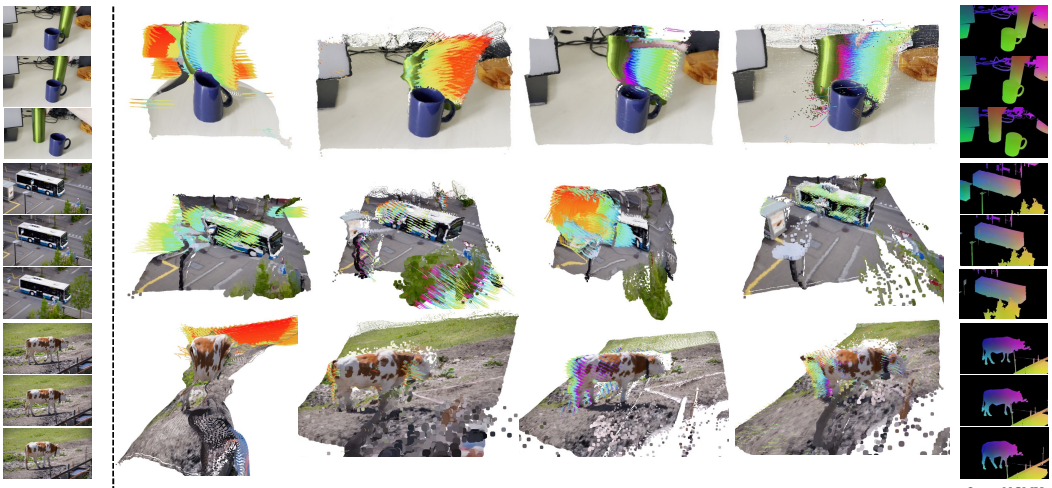


Figure 5: **Qualitative comparison of dynamic tracking on DAVIS [70].** Dense 3D tracking is visualized as colored point clouds, with color encoding query-frame identity of each tracked point. Left: input frames. Right: reconstructions from St4RTrack [19], TraceAnything [31], Any4D [23], and TRACK4D (ours). TRACK4D produces more temporally coherent trajectories with fewer outliers, particularly on articulated and deformable motion (*e.g.* moving mug, walking cows). All baselines are run from their official released checkpoints and inference scripts at the resolution recommended by the authors; sequences shown were selected to expose dynamic-object failure modes for the comparators. More DAVIS results appear in Figure 14.

Table 1: **Sparse 3D point tracking ($\mathrm{Sim}(3)$ aligned).** APD↑ and EPE↓ after a global $\mathrm{Sim}(3)$ alignment to the ground truth, as in 4RC [34], on all points (All) and dynamic points (Dyn). Bold and underline mark **best** and second-best results. CoTracker*: CoTracker lifted by VGGT. TRACK4D-NCMV and TRACK4D-RTV are the two readouts of the joint Wan2.1-I2V-14B model; TRACK4D-NCMV predicts dynamic points only. ‡Official checkpoint replayed with STream3R geometry under the same scorer (dynamic ADT not reported). †TRACK4D-Reg (Section 3.4): Wan2.2-5B, STream3R scaffold, eight training sources including PointOdyssey and Dynamic Replica.

	PointOdyssey [44]				**ADT** [89]				**PStudio** [90]		**Dynamic Replica** [45]			
	All		Dyn		All		Dyn		Dyn		All		Dyn	
Method	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓
CoTracker* [75]	0.6165	0.4721	0.4722	0.5264	0.8303	0.1664	0.9157	**0.0397**	0.7579	0.2031	0.6319	0.3842	0.6456	0.3014
SpaTrackerV2 [17]	0.2655	1.1016	0.3512	0.6921	0.5594	0.5158	0.8509	0.0621	0.5325	0.4211	0.6264	0.4052	0.5380	0.3910
St4RTrack [19]	0.6751	0.3126	0.7380	0.2146	0.7848	0.1990	0.8864	0.0526	0.8002	0.1709	0.7353	0.2651	0.7944	0.1681
TraceAnything [31]	0.2629	1.1003	0.4323	0.5798	0.8156	0.1773	0.8708	0.0559	0.6469	0.3085	0.3766	0.7415	0.5418	0.3972
Any4D [23]	0.5332	0.6213	0.4012	0.6243	0.7955	0.1787	0.8633	0.0592	0.7702	0.1971	0.7484	0.2647	0.6668	0.2723
V-DPM [22]	**0.8350**	**0.1726**	**0.8169**	**0.1688**	0.8880	**0.1109**	0.8590	0.0593	**0.9180**	**0.0916**	**0.8065**	**0.2032**	0.7761	0.1900
TrackCraft3R‡ [66]	0.7904	0.2227	0.7734	0.2494	0.8874	0.1163	–	–	0.8418	0.1400	0.7839	0.2289	0.7571	0.2207
TRACK4D-NCMV	–	–	0.4798	0.5162	–	–	0.8855	0.0513	0.7495	0.2098	–	–	0.6587	0.2873
TRACK4D-RTV	0.7786	0.2289	0.6896	0.2887	**0.8920**	0.1130	**0.9541**	0.0610	0.8101	0.1596	0.7751	0.2374	0.7847	0.1859
TRACK4D-Reg†	0.8050	0.2078	0.7155	0.2691	0.8854	0.1168	0.9430	0.0675	0.8572	0.1311	0.7871	0.2284	**0.8141**	**0.1674**

better than St4RTrack and Any4D. V-DPM is strongest on PO, PStudio and DR All; on PO it is trained on the same benchmark and builds on a VGGT backbone trained on broader data (Appendix G.1).

RTV vs. NCMV: representation comparison. On dynamic points (where NCMV produces predictions), TRACK4D-RTV consistently outperforms TRACK4D-NCMV, *e.g.*, 0.690 vs 0.480 APD on PO Dyn: RTV reads 3D offsets directly (Equation (4)), while NCMV adds a 2D-tracker decoding stage that compounds errors. Yet TRACK4D-NCMV *still* outperforms CoTracker3* [75] (same VGGT lifting) on PO Dyn (0.480 vs 0.472) and DR Dyn (0.659 vs 0.646), and is comparable on PStudio and ADT: feeding CoTracker the VDM-generated NCMV instead of RGB improves APD on PO and DR, evidence that a VDM can produce a structured dense correspondence signal as output, not only as an internal feature [32, 33, 67]. We therefore recommend RTV for deployment and treat NCMV as a probe of VDM correspondence priors and an auxiliary training target (Appendix G.4).

4.4 Ablations and analysis

Table 2: **Ablations of design choices.** Baseline is TRACK4D (Wan 2.2 5B, STream3R). Top three rows modify one component each; bottom rows scale the VDM backbone (1.3B / 14B) and swap conditioning (DAv3 / STream3R). Green/red show gains/losses over baseline; intensity indicates magnitude. All models are trained and evaluated at 256×256 following [24]. Reported on ADT [89], PStudio [90], and DR [45]; PO [44] is excluded due to mismatched aspect ratios.

Variant	**ADT** [89]				**PStudio** [90]		**DR** [45]			
	All		Dyn		Dyn		All		Dyn	
	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓	APD↑	EPE↓
TRACK4D (Wan 2.2 5B)	0.8831	0.1196	0.9370	0.0722	0.7281	0.2231	0.5245	0.4468	0.7009	0.2536
w/o Residual Pred	0.6625	0.3235	0.9218	0.0859	0.6100	0.3358	0.2959	0.8439	0.4837	0.4756
w/o Log Norm	0.8793	0.1217	0.9200	0.0837	0.7109	0.2384	0.5109	0.4668	0.6857	0.2690
w/o XYZ Cond	0.8722	0.1289	0.8974	0.1046	0.6396	0.3106	0.4386	0.5718	0.5908	0.3608
TRACK4D (Wan 2.1 1.3B)	0.8588	0.1391	0.8601	0.1295	0.6928	0.2574	0.5283	0.4425	0.6999	0.2549
TRACK4D (Wan 2.1 14B, DAv3)	0.9182	0.0854	0.9293	0.0777	0.8050	0.1652	0.7309	0.2769	0.6415	0.3093
TRACK4D (Wan 2.1 14B, STream3R)	0.8867	0.1164	0.9390	0.0715	0.7855	0.1762	0.5300	0.4391	0.7183	0.2379

Table 2 ablates key design choices at a training and evaluation resolution of 256×256.

Importance of residual 3D tracking representation. Replacing RTV with absolute 3D coordinate prediction (*w/o Residual Pred*) causes a substantial drop across all benchmarks (*e.g.*, APD on DR All $0.525 \rightarrow 0.296$). Predicting absolute coordinates forces the model to memorize static scene geometry; the residual formulation factors out the static component and lets the VDM focus on dynamic deformation (Appendix Figure 12).

Normalization and conditioning. Replacing log-space normalization with min-max (*w/o Log Norm*) degrades performance, particularly on dynamic points (*e.g.*, ADT Dyn $0.937 \rightarrow 0.920$): min-max compresses small displacements into a narrow range, whereas log-space expands them and maps zero displacement to zero. Removing the point-map conditioning (*w/o XYZ Cond*) degrades most on PStudio ($0.728 \rightarrow 0.640$) and DR Dyn ($0.701 \rightarrow 0.591$), supporting an explicit 3D scaffold.

Robustness to scaffold quality. DAv3 [14] yields stronger all-point tracking, while STream3R [13] performs better on dynamic points, likely because DAv3 provides more accurate per-frame depth, benefiting static regions, while STream3R's causal architecture better preserves temporal consistency for moving objects. The trade-off is largest on DR (dynamic-point APD $0.642 \rightarrow 0.718$ with STream3R, our default scaffold).

Scaling of video diffusion models. A key advantage of our framework is that it directly inherits advances in video generation. We compare 1.3B, 5B and 14B Wan backbones with identical hyperparameters. Larger models generally improve tracking, suggesting that stronger VDM priors translate into stronger tracking; with LoRA, even the 14B model trains on academic resources.

4.5 Scaling the regression readout

The generative readout is limited by the video VAE (Appendix G.8) and by dense supervision; we ask whether the regression readout of Section 3.4 removes both limits once scaled, training it on eight dense and sparse sources that, as for V-DPM and 4RC, include PointOdyssey and Dynamic Replica (Appendix G.6). We compare with TrackCraft3R [66], which keeps the VAE output path, replaying its official checkpoint and TRACK4D-Reg under the scorer of Table 1 with the same VGGT-Ω [15], DAv3-L [14] or STream3R point maps.

Table 9 shows that TRACK4D-Reg improves both metrics over TrackCraft3R in nine of twelve benchmark/geometry cells; EPE differs by less than 0.002 in the other three. Both systems, however, vary substantially with the source geometry, and no scaffold is best everywhere (Appendix G.7); as the two models differ in backbone, raster, training data, loss and readout, the table compares complete systems. TRACK4D-Reg ranks second behind V-DPM on PointOdyssey and Dynamic Replica (all points) and first on Dynamic Replica's dynamic points (Table 1). The official V-DPM [22] and 4RC [34] checkpoints, replayed under the same scorer with their own geometry, remain stronger

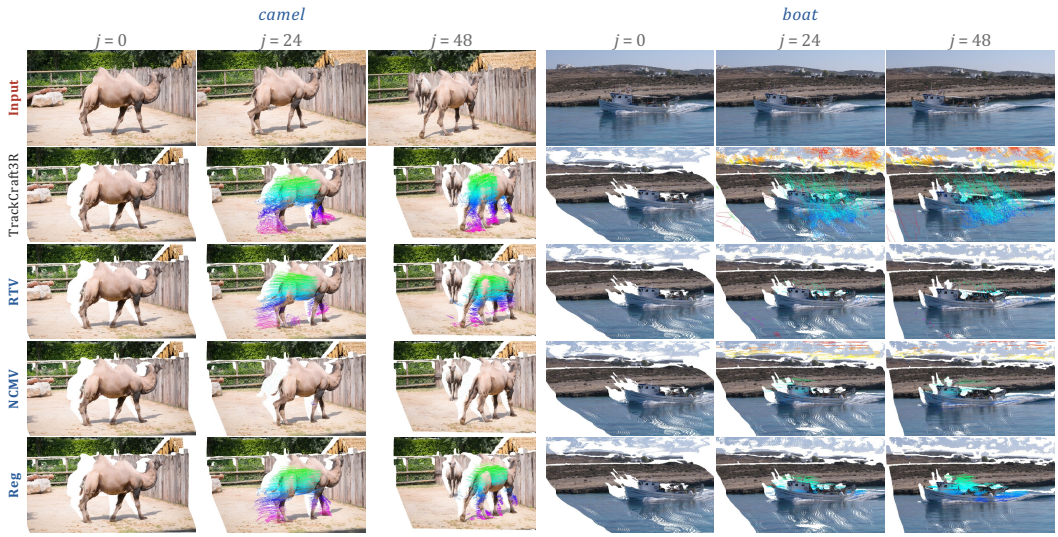


Figure 6: **TrackCraft3R *vs*. the three TRACK4D readouts on DAVIS [70].** Each panel shows the point cloud of frame j and the last 12 frames of the predicted 3D tracks, from one fixed view (the frame-0 input camera rotated by $4°$). All methods receive the same VGGT-Ω point maps and draw the same 767 queries, the most-moving points of a dense grid, colored by image row. RTV and NCMV come from the joint Wan2.1-I2V-14B model (50 steps, no input-latent noise); NCMV tracks are read with CoTracker3, lifted with the same point maps, and frozen where NCMV support is lost (on *camel*, after the first frames). No ground truth or $\mathrm{Sim}(3)$ is used. More clips in Figures 9 and 10.

on PointOdyssey and Dynamic Replica even against our best scaffold (0.14–0.19 against 0.21–0.23 EPE, Table 9) and close on PStudio and ADT; both predict geometry jointly with motion, exactly where TRACK4D-Reg is weakest. In Figure 6, its tracks stay on the boat, whereas TrackCraft3R's spread over the water and sky; over all 91 DAVIS clips, the in-frame median second difference of the projected tracks is 1.8 *vs*. 6.5 px, smaller for TRACK4D-Reg on 90.

Geometry or motion? Substituting either factor of the output, the supplied query point or the predicted relative motion (Equation (11)), with ground truth *after* prediction (Table 11; a diagnostic only) shows that query geometry dominates the error on PointOdyssey and Dynamic Replica: EPE drops by an order of magnitude ($0.413 \to 0.053$ and $0.226 \to 0.027$ with VGGT-Ω), versus at most 0.01 for ground-truth motion. As a residual on the input point map, the readout is limited by that geometry, which suggests training geometry and motion jointly.

5 Conclusions and Future Work

We presented TRACK4D, an early systematic study of how to represent dense 3D point tracking inside a pretrained video diffusion model. We compared two natural representations within a single VDM: a Residual 3D Tracking Video (RTV) from which metric tracks are read directly, and a Normalized Coordinate Map Video (NCMV) that gives each 3D point a canonical color consistent over time. Both are produced by LoRA adaptation of a frozen Wan backbone conditioned on point maps from an off-the-shelf 3D reconstructor. Trained with a single dynamic 3D tracking dataset, TRACK4D is competitive across four zero-shot benchmarks and outperforms St4RTrack and TraceAnything by a clear margin on the held-out dense Kubric split. Our central finding is that VDM priors transfer to dense 3D tracking when the right representation is chosen, and RTV is the cleaner deployment path. Without the output VAE, residual regression admits sparse supervision and, once scaled, outperforms TrackCraft3R on most benchmark/geometry pairs, showing the potential of this direction. We view TRACK4D as an initial step toward repurposing VDM priors for 4D reconstruction.

Limitations and future work. (1) RTV needs a target at every pixel, which restricts training to a single dense dynamic 3D tracking dataset; (2) NCMV yields less accurate tracks; (3) TRACK4D-Reg is limited by its input geometry. A natural next step is a unified model that predicts geometry and motion jointly, trained on large-scale 3D/4D data; see Appendix A for further directions.

Acknowledgments and Disclosure of Funding

This research was supported by the Sony Research Award Programme 2024 and ERC CoG UNION (101001212). We thank Isambard-AI for support, and Yihang Luo and Gabrijel Boduljak for their discussion. The authors declare no competing interests.

References

[1] Yao-Chih Lee, Zhoutong Zhang, Jiahui Huang, Jui-Hsien Wang, Joon-Young Lee, Jia-Bin Huang, Eli Shechtman, and Zhengqi Li. Generative video motion editing with 3D point tracks. In *CVPR*, 2026.

[2] Zeqi Xiao, Yiwei Zhao, Lingxiao Li, Yushi Lan, Ning Yu, Rahul Garg, Roshni Cooper, Mohammad H. Taghavi, and Xingang Pan. Video4Spatial: Towards visuospatial intelligence with context-guided video generation. In *CVPR Findings*, 2026.

[3] Seungjae Lee, Yoonkyo Jung, Inkook Chun, Yao-Chih Lee, Zikui Cai, Hongjia Huang, Aayush Talreja, Tan Dat Dao, Yongyuan Liang, Jia-Bin Huang, and Furong Huang. TraceGen: World modeling in 3D trace space enables learning from cross-embodiment videos. In *CVPR*, 2026.

[4] 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. In *CVPR*, 2026.

[5] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *TPAMI*, 30(5), 2008.

[6] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[7] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. In *ICCV*, 2025.

[8] Frano Rajič, Haofei Xu, Marko Mihajlovic, Siyuan Li, Irem Demir, Emircan Gündoğdu, Lei Ke, Sergey Prokudin, Marc Pollefeys, and Siyu Tang. Multi-view 3D point tracking. In *ICCV*, 2025.

[9] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024.

[10] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *CVPR*, 2025.

[11] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025.

[12] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *ICLR*, 2026.

[13] Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. STream3R: Scalable sequential 3D reconstruction with causal transformer. In *ICLR*, 2026.

[14] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

[15] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *CVPR*, 2026.

[16] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: Tracking any 2D pixels in 3D space. In *CVPR*, 2024.

[17] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Iurii Makarov, Bingyi Kang, Xin Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. SpatialTrackerV2: 3D point tracking made easy. In *ICCV*, 2025.

[18] Bowei Zhang, Lei Ke, Adam W Harley, and Katerina Fragkiadaki. TAPIP3D: Tracking any point in persistent 3D geometry. In *NeurIPS*, 2025.

[19] 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 *ICCV*, 2025.

[20] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4D: Learning how things move in 3D from internet stereo videos. In *CVPR*, 2025.

[21] Edgar Sucar, Zihang Lai, Eldar Insafutdinov, and Andrea Vedaldi. Dynamic Point Maps: A versatile representation for dynamic 3D reconstruction. In *ICCV*, 2025.

[22] Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-DPM: 4D video reconstruction with dynamic point maps. In *CVPR*, 2026.

[23] Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4D: Unified feed-forward metric 4D reconstruction. In *CVPR Workshops*, 2026.

[24] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one D4RT at a time. In *CVPR*, 2026.

[25] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI technical report, 2024.

[26] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[27] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, 2024.

[28] Zeren Jiang, Chuanxia Zheng, Iro Laina, Diane Larlus, and Andrea Vedaldi. Geo4D: Leveraging video generators for geometric 4D scene reconstruction. In *ICCV*, 2025.

[29] 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. In *CVPR*, 2026.

[30] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *CVPR*, 2022.

[31] 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. In *ICLR*, 2026.

[32] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *NeurIPS*, 2023.

[33] Jisu Nam, Soowon Son, Dahyun Chung, Jiyoung Kim, Siyoon Jin, Junhwa Hur, and Seungryong Kim. Emergent temporal correspondences from video diffusion transformers. In *NeurIPS*, 2025.

[34] Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4RC: 4D reconstruction via conditional querying anytime and anywhere. In *ICML*, 2026.

[35] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.

[36] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[37] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent MVSNet for high-resolution multi-view stereo depth inference. In *CVPR*, 2019.

[38] 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 *CVPR*, 2025.

[39] Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4D Gaussian transformer using real-world monocular videos. In *NeurIPS*, 2025.

[40] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. In *NeurIPS*, 2025.

[41] Songyan Zhang, Yongtao Ge, Jinyuan Tian, Guangkai Xu, Hao Chen, Chen Lv, and Chunhua Shen. POMATO: Marrying pointmap matching with temporal motions for dynamic 3D reconstruction. In *ICCV*, 2025.

[42] Yiqing Liang, Abhishek Badki, Hang Su, James Tompkin, and Orazio Gallo. Zero-shot monocular scene flow estimation in the wild. In *CVPR*, 2025.

[43] 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 *ECCV*, 2026.

[44] 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 *ICCV*, 2023.

[45] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *CVPR*, 2023.

[46] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[47] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[48] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[49] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022.

[50] 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. In *NeurIPS*, 2022.

[51] 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 *ICCV*, 2023.

[52] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *ECCV*, 2024.

[53] Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas J. Guibas. AllTracker: Efficient dense point tracking at high resolution. In *ICCV*, 2025.

[54] Tuan Duc Ngo, Peiye Zhuang, Evangelos Kalogerakis, Chuang Gan, Sergey Tulyakov, Hsin-Ying Lee, and Chaoyang Wang. DELTA: Dense efficient long-range 3D tracking for any video. In *ICLR*, 2025.

[55] 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.

[56] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. In *NeurIPS*, 2021.

[57] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022.

[58] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023.

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

[60] 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.

[61] Gangwei Xu, Haotong Lin, Hongcheng Luo, Xianqi Wang, Jingfeng Yao, Lianghui Zhu, Yuechuan Pu, Cheng Chi, Haiyang Sun, Bing Wang, Guang Chen, Hangjun Ye, Sida Peng, and Xin Yang. Pixel-perfect depth with semantics-prompted diffusion transformers. In *NeurIPS*, 2025.

[62] Zhaoxi Chen, Tianqi Liu, Long Zhuo, Jiawei Ren, Zeng Tao, He Zhu, Fangzhou Hong, Liang Pan, and Ziwei Liu. 4DNeX: Feed-forward 4D generative modeling made easy. *arXiv preprint arXiv:2508.13154*, 2025.

[63] Hyojun Go, Dominik Narnhofer, Goutam Bhat, Prune Truong, Federico Tombari, and Konrad Schindler. Text-to-3D by stitching a multi-view reconstruction network to a video generator. In *ICLR*, 2026.

[64] Jiaxin Huang, Yuanbo Yang, Bangbang Yang, Lin Ma, Yuewen Ma, and Yiyi Liao. Gen3R: 3D scene generation meets feed-forward reconstruction. In *CVPR*, 2026.

[65] Jiahui Lei, Kyle Genova, George Kopanas, Noah Snavely, and Leonidas Guibas. MoMaps: Semantics-aware scene motion generation with motion maps. In *ICCV*, 2025.

[66] Jisu Nam, Jahyeok Koo, Soowon Son, Jaewoo Jung, Honggyu An, Junhwa Hur, and Seungryong Kim. TrackCraft3R: Repurposing video diffusion transformers for dense 3D tracking. In *NeurIPS*, 2026.

[67] Hyeonho Jeong, Chun-Hao P. Huang, Jong Chul Ye, Niloy J. Mitra, and Duygu Ceylan. Track4Gen: Teaching video diffusion models to track points improves video generation. In *CVPR*, 2025.

[68] Ayush Shrivastava, Sanyam Mehta, Daniel Geng, and Andrew Owens. Point prompting: Counterfactual tracking with video diffusion models. In *ICLR*, 2026.

[69] Soowon Son, Honggyu An, Jisu Nam, Hyunah Ko, Chaehyun Kim, Dahyun Chung, Jung Yi, Siyoon Jin, Junhwa Hur, and Seungryong Kim. Probing and leveraging video diffusion transformer features for robust point tracking. *arXiv preprint arXiv:2512.20606*, 2025.

[70] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[71] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *CVPR*, 2019.

[72] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. CRM: Single image to 3D textured mesh with convolutional reconstruction model. In *ECCV*, 2024.

[73] Yushi Lan, Chen Change Loy, and Bo Dai. DDF: Correspondence distillation from NeRF-based GAN. *IJCV*, 132(3):611–631, 2024.

[74] Jiteng Mu, Shalini De Mello, Zhiding Yu, Nuno Vasconcelos, Xiaolong Wang, Jan Kautz, and Sifei Liu. CoordGAN: Self-supervised dense correspondences emerge from GANs. In *CVPR*, 2022.

[75] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *ICCV*, 2025.

[76] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[77] 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 *ECCV*, 2024.

[78] Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. UniMax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *ICLR*, 2023.

[79] 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 *ICLR*, 2022.

[80] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-one video creation and editing. In *ICCV*, 2025.

[81] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[82] Runjia Li, Moayed Haji-Ali, Ashkan Mirzaei, Chaoyang Wang, Arpit Sahni, Ivan Skorokhodov, Aliaksandr Siarohin, Tomas Jakab, Junlin Han, Sergey Tulyakov, Philip Torr, and Willi Menapace. EgoEdit: Dataset, real-time streaming model, and benchmark for egocentric video editing. In *CVPR*, 2026.

[83] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[84] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[85] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

[86] Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *TMLR*, 2025.

[87] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[88] 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 *IROS*, 2020.

[89] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria Digital Twin: A new benchmark dataset for egocentric 3D machine perception. In *ICCV*, 2023.

[90] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic Studio: A massively multiview system for social interaction capture. *TPAMI*, 41(1), 2019.

[91] Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, João Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. TAPVid-3D: A benchmark for tracking any point in 3D. In *NeurIPS*, 2024.

[92] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.

[93] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, and Di Zhang. ReCamMaster: camera-controlled generative rendering from a single video. In *ICCV*, 2025.

[94] Homanga Bharadhwaj and Yash Jangir. MotionForesight: Re-purposing video models for future 3D scene-flow prediction. *arXiv preprint arXiv:2607.16192*, 2026.

[95] Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3D with language instruction. *arXiv preprint arXiv:2606.18558*, 2026.

[96] Stanislaw Szymanowicz, Jason Y. Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T. Barron, and Philipp Henzler. Bolt3D: Generating 3D scenes in seconds. In *ICCV*, 2025.

[97] Michael J. Black, Priyanka Patel, Joachim Tesch, and Jinlong Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *CVPR*, 2023.

[98] David Yifan Yao, Albert J. Zhai, and Shenlong Wang. Uni4D: Unifying visual foundation models for 4D modeling from a single video. In *CVPR*, 2025.

[99] Zixuan Huang, Xiang Li, Zhaoyang Lv, and James M. Rehg. How much 3D do video foundation models encode? In *CVPR*, 2026.

[100] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[101] 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 *CVPR*, 2024.

[102] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017.

[103] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018.

[104] Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, Andrei Zanfir, and Cristian Sminchisescu. Video generation models are general-purpose vision learners. In *ECCV*, 2026.

[105] NVIDIA et al. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

A Limitations and future work

Our study exposes a trade-off between learnability and accuracy: (1) NCMV is expected to be easier for the VDM to learn but yields less accurate tracks; (2) decoding RTV through the video VAE introduces artifacts near object boundaries and needs a target at every pixel, which restricts training to a single dense dynamic 3D tracking dataset; (3) a regression head that infers motion from VDM features, akin to inverse dynamics, admits sparse supervision and benefits from additional training data (Section 4.5). We look forward to a unified model that predicts geometry and motion directly, trained on large-scale 3D/4D data, which requires more compute.

TRACK4D is trained on 49-frame clips (TRACK4D-Reg on 11-frame windows) and currently assumes the first frame as the query frame; extending it to longer videos and arbitrary query frames remains future work. The generative readouts do not predict per-point visibility. Faster sampling, *e.g.* by consistency-style distillation, is another direction. Forecasting 3D tracks beyond the observed frames has been explored for scene flow [94] and point trajectories [95]; conversely, Track4Gen [67] shows that learning to track points improves video generation.

The generative readouts remain noisy at boundaries: the scaffold dominates global error, while the video VAE adds boundary artifacts (Appendix G.8). Transformer-based VAEs may reduce these artifacts but require retraining [96]. Camera-following scenes can yield near-static RTV (Figure 11); the normalization audit preserves motion targets but does not establish the cause (Appendix G.9). Broader synthetic data [97] or real-video pseudo-labels [98] could test the training-coverage hypothesis. We do not evaluate accuracy in occluded regions and make no claim of superiority there. TRACK4D-Reg predicts visibility, but we do not evaluate its accuracy. Direct feature readout lowers overall error and admits sparse supervision (Appendix G.5 and Section 4.5), yet query geometry dominates the error of TRACK4D-Reg on PointOdyssey and Dynamic Replica (Appendix G.7).

B Changes from the submitted version

The task, title, and central question are unchanged: TRACK4D represents dense 3D tracking inside a pretrained video diffusion model, and compares a residual readout (RTV) with a coordinate-map readout (NCMV). The generative model, its checkpoint, and every number in the submitted tables (main results, ablations, sampling steps) are unchanged. This version adds a third readout that regresses the same residual without the video VAE, adds TrackCraft3R as a baseline, and expands the analysis. Table 3 summarizes the differences and Figure 7 reproduces the submitted overview.

Table 3: **Submitted version versus this version.** The submitted column summarizes the described design; the training run that produced the submitted numbers is distinguished below.

	Submitted version	This version
Readouts	RTV and NCMV, generated jointly by a LoRA-adapted Wan2.1-I2V-14B	Unchanged; adds TRACK4D-Reg, a regression readout of the RTV residual without the output VAE (Section 3.4)
Generative model	Wan2.1-I2V-14B, LoRA rank 64, point-map scaffold, Kubric and TartanAir	Unchanged checkpoint
Regression model	–	Wan2.2-TI2V-5B with a motion expert and a DPT head, eight training sources (Appendix G.6)
Main table	Four sparse benchmarks and dense Kubric	Four sparse benchmarks; adds TRACK4D-Reg and TrackCraft3R; dense Kubric in Table 5
Analyses	Ablations, sampling steps	Adds runtime, role of NCMV, VAE-free readout, artifact sources, camera-following audit, geometry/motion oracle (Appendix G)
Figures	Overview of the generative model (Figure 7)	Overview redrawn with the regression readout (Figure 2); DAVIS comparison with TrackCraft3R (Figures 6 and 9)

Training details. The submitted text stated 10,000 iterations on 4×A6000 GPUs with batch size 16 and learning rate 2×10^{-5}. The joint checkpoint behind the submitted and current numbers was

continued from an RTV-only adapter for 5,000 joint-training iterations on 32 GH200 GPUs for about one day, with learning rate 4×10^{-5}; Appendix D now states this configuration.

Evaluation. Dense Kubric moved from the main table to Table 5, since it is not part of the official WorldTrack evaluation. TrackCraft3R [66] and TRACK4D-Reg are scored with the scorer of the main table; V-DPM and 4RC are replayed under the same scorer as references (Table 9). Several of the analyses in Appendix G were first reported in the author response.

Presentation. The abstract, introduction, related work, and conclusion each gain sentences on the regression readout and on concurrent work; Sections 3.4 and 4.5 are new. To keep the main text within ten pages, the residual-prediction ablation, additional DAVIS comparison, ablation table, and implementation details appear in the appendix. The main text retains both representation figures.

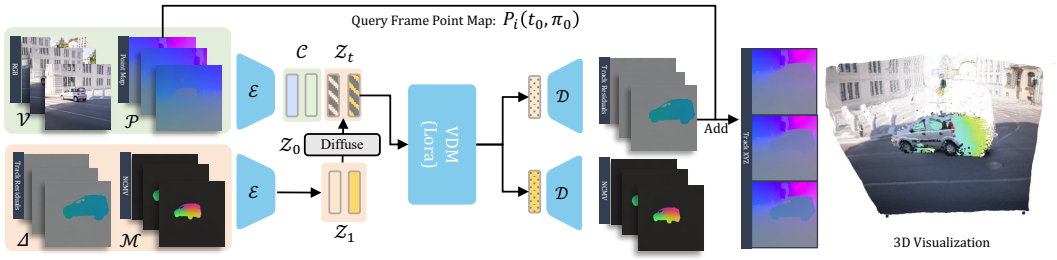


Figure 7: **Submitted overview** (reproduced from the submitted version). A monocular video $\mathcal{V}$ and point maps $\mathcal{P}$ from an off-the-shelf reconstructor are encoded by the video VAE into the condition $\mathcal{C}$; the VDM jointly generates the RTV $\boldsymbol{\Delta}$ and the NCMV $\mathcal{M}$, and the decoded residuals are added to the query-frame point map. Figure 2 shows the current overview.

C Dataset details

The generative readouts require video data with **dense 3D tracking annotations** for training. Existing 3D tracking datasets such as PointOdyssey [44] and Dynamic Replica [45] are not suitable: they provide only sparse annotations. DELTA [54] provides a pre-processed Kubric set, but lacks flexibility in selecting the query frame (only predefined frames are queryable) and does not include the NCMV representation; its sequences contain only 21 frames, whereas our generative model uses 49.

We re-render Kubric with the official Kubric-F [30] pipeline, redesigning the camera trajectories to introduce larger viewpoint variation. The training set is 5,600 sequences of 49 frames each, with RGB, depth, camera, and dense 3D tracking annotations. NCMV targets are obtained by projecting query-frame 3D tracks onto the target camera trajectory. We reserve a held-out evaluation split disjoint from training. NCMV videos are computed from the rendered tracks. We additionally include TartanAir [88] for static-scene diversity; for these sequences, the tracking ground truth is set to zero motion and NCMV supervision is masked out. We will release rendering scripts and the rendered training/evaluation splits.

D Architecture and LoRA details

The generative readouts introduce no new output heads. Both RTV and NCMV are 3-channel videos and are encoded by the original Wan VAE without modification — the latent channel count is unchanged from the released VDM. Multi-modality is handled at the token-sequence level: the clean target $\mathcal{Z}_1 = [x_{\boldsymbol{\Delta}}; x_M]$ and the conditioning $\mathcal{C} = [x_{\mathcal{V}}; x_{\mathcal{P}}]$ are each width-concatenated, then concatenated again to form the $4w$-wide DiT input (Section 3.3), following [62, 64].

LoRA [79] is applied with rank 64 to the attention and MLPs of the DiT; the VAE encoder/decoder, the patch embedder, and the umT5 [78] text encoder are kept frozen. We add four learnable domain-specific tokens (one per modality: RGB, point map, RTV, NCMV) to differentiate input streams, following [62]. Trainable parameters account for well under 1% of the DiT, allowing the 14B Wan backbone to be fine-tuned with plain data parallelism on 32 GH200 GPUs. We use a per-GPU batch

Table 4: **Sampling-step ablation.** We ablate inference-time sampling steps using a Wan2.2-5B TRACK4D model trained at 256×256 on 512 Kubric scenes without point-map conditioning. Performance remains largely stable when reducing the number of sampling steps from 50 to 10, with 5 steps still competitive while further reducing runtime.

Steps	**EPE**↓					**Time / Seq.** ↓
	Kubric [30]	PO [44]	ADT [89]	PStudio [90]	DR [45]	
50	1.0463	0.3953	0.1149	0.2646	0.2549	14.0 ± 1.3 s
10	**1.0394**	0.3821	0.1066	**0.2642**	**0.2548**	9.6 ± 1.1 s
5	1.0862	**0.3773**	**0.1039**	0.2668	0.2610	$\mathbf{8.6 \pm 1.1}$ s

of 2 and learning rate 4×10^{-5}, with all other hyperparameters identical to the official Wan release; inputs are resized so the short side is up to 256 pixels, with 49 frames per clip.

The Wan implementation is adapted from DiffSynth-Studio with the video data-loading pipeline from STream3R [13]; the multi-view augmentations originally designed for STream3R are removed. For all loaded videos we follow Geo4D [28]: we normalize the camera trajectory to the first-frame coordinate system and scale the point map by `common_scale` (the mean absolute scale of $\mathcal{P}$). Tracking offsets use the same geometric scale; the signed-log transform is inverted before adding predicted offsets to the query-frame point map. Undoing the geometric scaling then recovers metric trajectories.

E Comparison protocol details

Table 1 compares baselines under their native protocols, each using its own depth/camera estimator. Following D4RT [24], each method runs at its native resolution: baselines resize internally, while the generative TRACK4D uses a 256-pixel short side (TRACK4D-Reg: Appendix G.6). We summarize the conditioning interfaces below for transparency:

- St4RTrack [19]: jointly predicts tracking and depth from RGB; no external scaffold.
- TraceAnything [31]: end-to-end from RGB.
- Any4D [23]: built on a VGGT-style 3D backbone trained on broad data.
- V-DPM [22]: built on a VGGT-style 3D backbone trained on broad data and fine-tuned on multiple tracking benchmarks.
- CoTracker3* [75]: 2D tracker, lifted to 3D via VGGT [11].
- SpatialTrackerV2 [17]: joint depth + tracking head.
- TRACK4D (ours): conditioned on point maps from STream3R [13] or DAv3 [14].

Note on SpatialTrackerV2. SpatialTrackerV2 also relies on scene-geometry conditioning; ours uses a stronger 3D backbone (STream3R / DAv3) than its default depth pipeline. Replacing SpatialTrackerV2's depth with our scaffold requires retraining its fusion head, which we do not pursue.

F Sampling steps

VDMs typically require many denoising steps to generate high-quality videos. Although TRACK4D is built on a VDM backbone, it remains effective with substantially fewer sampling steps. Table 4 shows that reducing the number of steps preserves, and in some cases improves, tracking performance while lowering inference time, consistent with observations in Geo4D [28]. Understanding and exploiting this sampling robustness for faster inference is an interesting direction for future work. Unless otherwise stated, generative-model experiments use 50 sampling steps; TRACK4D-Reg uses no denoising. The inference time is evaluated on a single NVIDIA A6000 GPU; peak GPU memory is 31.9 GB and is essentially independent of the sampling step count.

G Additional experiments and analyses

This section reports additional experiments and diagnostics motivated by questions about task-specific data efficiency, the role of NCMV, computational cost, and the source of geometric artifacts. Unless noted, metrics use the main paper's closed-form $\mathrm{Sim}(3)$ alignment.

G.1 Task-specific post-training data

Both TRACK4D and V-DPM [22] adapt pretrained visual backbones rather than training them from scratch: TRACK4D builds on Wan, while V-DPM builds on VGGT. We therefore distinguish generic foundation-model pretraining from task-specific post-training. The dynamic component of generative TRACK4D uses $5{,}600 \times 49 \approx 274$K densely annotated Kubric frames. V-DPM instead uses four dynamic sources (Kubric-F, Kubric-G, PointOdyssey, and Waymo), alongside two static sources (ScanNet++ and BlendedMVS). Its reported per-epoch sampling counts do not establish a directly comparable total of distinct annotated frames, so we do not infer a frame-count ratio. This comparison does not claim lower total pretraining cost, since the generic pretraining data and compute of Wan and VGGT are not directly comparable.

The distinction also affects benchmark interpretation. PointOdyssey is included in the training data of V-DPM, SpatialTrackerV2, St4RTrack, and Any4D, but is fully held out for the submitted TRACK4D model. The PointOdyssey results of the generative readouts in Table 1 therefore measure zero-shot transfer, whereas this benchmark is in-domain for these baselines and TRACK4D-Reg.

G.2 Dense Kubric evaluation

Table 5 reports the held-out dense Kubric split. It matches the training distribution of TRACK4D-RTV and is reported separately from the WorldTrack benchmarks in Table 1.

Table 5: **Dense 3D tracking on the held-out Kubric split.** APD↑ / EPE↓ after $\mathrm{Sim}(3)$ alignment on 50 held-out sequences of 49 frames, dense annotations; "All" and "Dyn" as in Table 1. Kubric matches the training distribution of TRACK4D-RTV and is not part of the benchmark suite of Table 1.

Method	All		Dyn	
	APD↑	EPE↓	APD↑	EPE↓
St4RTrack [19]	0.0468	5.3331	0.0546	2.8485
TraceAnything [31]	<u>0.2879</u>	<u>1.5501</u>	0.1379	1.9263
Any4D [23]	0.2242	1.9223	0.1370	1.8107
V-DPM [22]	0.1729	4.7054	**0.4822**	**0.6845**
TRACK4D-RTV	**0.3359**	**1.0790**	<u>0.1890</u>	<u>1.4042</u>

G.3 Inference time and memory

For the baseline timings, we measure inference cost on ten fixed 49-frame clips using each method's official preprocessing and native inference resolution. After warm-up, we record the mean and standard deviation of per-sequence runtime and the peak reserved GPU memory. Table 6 uses the 5-step TRACK4D configuration, which retains similar EPE to the default 50-step sampler (Table 4).

Table 6: **Inference time and peak memory for 49-frame sequences on one GPU.** Native inference resolutions are shown in parentheses. The TRACK4D row measures the 5B, 5-step configuration, which is not the configuration used for the accuracy results in Table 1. V-DPM does not fit on the 48 GB A6000; its H100 measurement is included only as a hardware-unmatched reference and should not be read as a direct runtime comparison. †Measured on an A100 over 90 warm DAVIS clips (five passes, BF16), excluding the production of the input point maps; also hardware-unmatched.

Method	**Hardware**	**Runtime (s/seq.)** ↓	**Peak memory (GB, reserved)** ↓
St4RTrack (512^2)	A6000 48 GB	6.07 ± 0.03	**9.56**
TraceAnything (512^2)	A6000 48 GB	6.76 ± 0.05	37.17
CoTracker + VGGT (476^2)	A6000 48 GB	7.53 ± 0.15	23.04
4RC (504×378)	A6000 48 GB	8.76 ± 0.09	20.60
Any4D (518^2)	A6000 48 GB	10.07 ± 0.14	43.31
SpatialTrackerV2 (518^2)	A6000 48 GB	28.00 ± 2.71	26.29
V-DPM (518^2)	A6000 48 GB	OOM	–
V-DPM (518^2)	H100 80 GB	139.39	53.49
TRACK4D (5B, 5 steps)	A6000 48 GB	8.60 ± 1.10	31.9
TRACK4D-Reg (288×512)†	A100 80 GB	6.51 ± 0.51	26.0

On the shared A6000 hardware, TRACK4D is slower and more memory-intensive than the lightest feed-forward baselines, but has similar runtime to 4RC and Any4D in this suite. Its 31.9 GB peak reserved memory is lower than TraceAnything and Any4D, but higher than St4RTrack, CoTracker+VGGT, 4RC, and SpatialTrackerV2. V-DPM exceeds the A6000 memory budget.

G.4 Role and supervision of NCMV

Foreground supervision. Kubric provides ground-truth instance masks. During training, we supervise NCMV only on foreground objects; for the static TartanAir clips, NCMV supervision is masked out. At inference, the model receives no external foreground mask. It predicts NCMV directly and must implicitly identify the relevant foreground regions. This separation is often successful, but it can fail when a moving object is interpreted as static.

RTV-only versus joint training. We compare the Wan-14B RTV-only checkpoint with its joint RTV+NCMV continuation in Table 7. The joint checkpoint improves EPE and APD on every benchmark, reducing mean all-point EPE by 2.34% and mean dynamic EPE by 2.93% (computed from unrounded means). Training history and sampling settings are not fully matched, so this is a checkpoint comparison rather than an isolated NCMV-loss ablation.

Table 7: **RTV-only and joint RTV+NCMV checkpoints.** Per-benchmark rows report dynamic EPE ↓ / APD ↑ on 49-frame sequences; the final row reports all-point means. Both are Wan-14B LoRA checkpoints with different training histories, and they are not sampled identically. The joint checkpoint is the model of Table 1: here it is sampled with 1 denoising step and CFG 1 at a 256-pixel short side, whereas Table 1 reports its submitted evaluation with 50 denoising steps and random input-latent noise (Section 4.1). The two tables therefore give different values for the same checkpoint (*e.g.* PStudio dynamic EPE 0.1511 here *vs*. 0.1596 in Table 1). Bold denotes the better result.

Benchmark	RTV-only	Joint RTV+NCMV
PointOdyssey	0.2891 / 0.6908	**0.2856 / 0.6931**
ADT	0.0715 / 0.9390	**0.0666 / 0.9466**
Panoptic Studio	0.1610 / 0.8093	**0.1511 / 0.8240**
Dynamic Replica	0.1886 / 0.7813	**0.1861 / 0.7850**
Mean (dynamic)	0.1776 / 0.8051	**0.1724 / 0.8122**
Mean (all-point)	0.1848 / 0.8134	**0.1804 / 0.8195**

The joint recipe improves RTV despite generating an additional representation. The table reports one checkpoint pair, not a multi-seed study, and does not isolate the auxiliary-task contribution. We retain RTV as the primary practical output and regard NCMV as an auxiliary representation analysis and training objective. Because NCMV assigns persistent canonical coordinates rather than anchoring every output to a single query frame, it can in principle represent points that first become visible later. Our targets, however, are built from query-frame tracks, so exploiting this would require canonical-coordinate supervision for such points; it has not been validated empirically here.

A dedicated 512×512 NCMV model. For visualization we also train a separate NCMV-only model. It adapts Wan2.1-T2V-14B with rank-64 LoRA, is trained only on Kubric NCMV targets ($5{,}636$ clips, 512×512) with a two-phase curriculum, $4{,}000$ steps on 29-frame windows and then $4{,}000$ steps on full 49-frame clips (effective batch 16, about 270 H200 GPU-hours in total), and samples NCMV in 50 denoising steps with source-video classifier-free guidance 3. Unlike the joint model, it is conditioned on the RGB video alone, without point maps. We use it only for qualitative results on the project page; every NCMV number in this paper comes from the joint model.

G.5 VAE-free direct feature readout

The original formulation encodes the dense tracking target as a three-channel video and decodes it through the RGB-pretrained video VAE. This requires a tracking value at every pixel and therefore prevents direct use of datasets that provide only sparse trajectories. Motivated by shallow readouts for measuring the 3D information in video-foundation-model features [99], we attach a shallow ViT/DPT-style regression head to intermediate Wan2.2-5B features and jointly train the head with LoRA adaptation of the backbone. The regression loss is applied only at annotated locations, making sparse tracking datasets compatible with the framework.

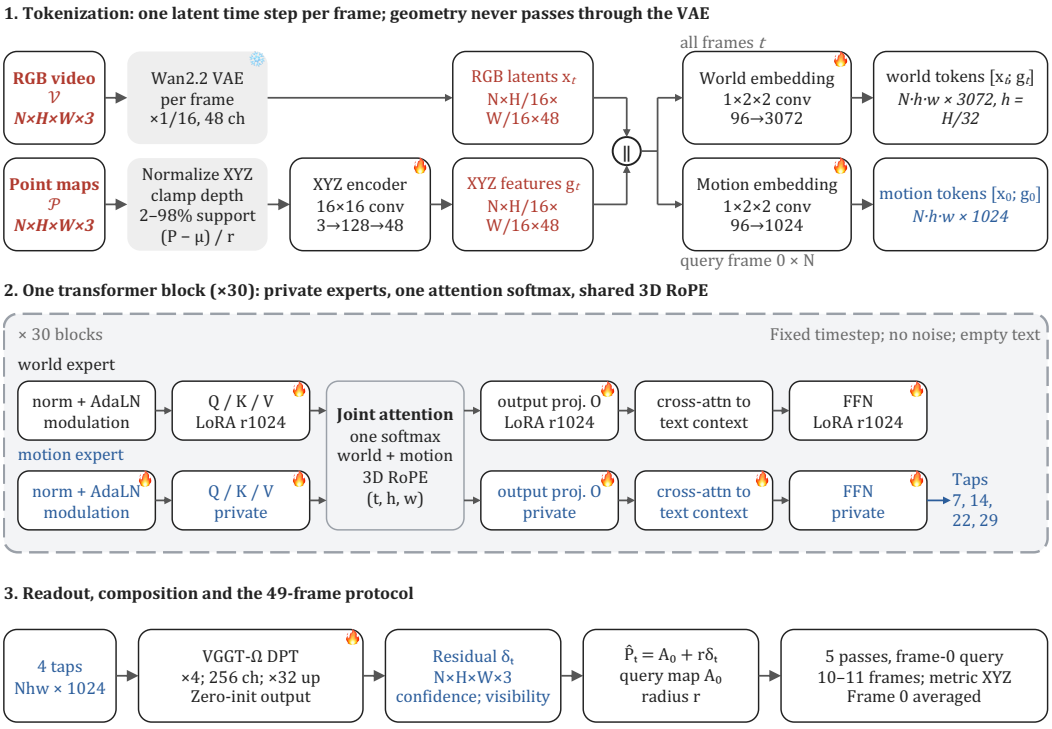


Figure 8: **Tensor flow of the regression readout (TRACK4D-Reg). Top:** frame-wise RGB latents and point-map features form world tokens; repeating the query-frame features forms motion tokens. **Middle:** the LoRA-adapted Wan expert and private motion expert share attention and 3D rotary positions in each block. **Bottom:** motion features from four blocks feed a DPT head for residuals and visibility. Residuals are added to the supplied query point map; five frame-0-anchored passes cover 49 frames. Flames: trained modules; VAE and Wan base weights: frozen.

Table 8 separates the effects of the output path, auxiliary reconstruction supervision, and additional tracking data. Replacing VAE decoding with direct readout under the same Kubric tracking data lowers mean dynamic EPE from 0.232 to 0.225. Adding auxiliary depth and camera regression lowers it to 0.212, and adding PointOdyssey tracks lowers it further to 0.186. The complete recipe improves mean dynamic EPE by approximately 20% and raises mean APD from 0.740 to 0.793.

Table 8: **VAE-free direct readout with a Wan2.2-5B backbone.** Dynamic EPE ↓ / APD ↑; the mean averages the four benchmarks. The first two rows compare the output paths under the same Kubric tracking data. The final row additionally uses PointOdyssey for training, so PointOdyssey is in-domain for that variant.

Model	PointOdyssey	ADT	PStudio	DynamicReplica	Mean
Submitted 5B (VAE decode)	0.379 / 0.594	**0.072 / 0.937**	0.223 / 0.728	0.254 / 0.701	0.232 / 0.740
Direct readout, Kubric tracking	0.321 / 0.655	0.117 / 0.871	0.251 / 0.695	0.210 / 0.753	0.225 / 0.743
+ depth and camera supervision	0.317 / 0.657	0.107 / 0.883	0.216 / 0.734	0.206 / 0.758	0.212 / 0.758
+ PointOdyssey tracking	**0.290 / 0.689**	0.090 / 0.913	**0.176 / 0.786**	**0.187 / 0.785**	**0.186 / 0.793**

The 0.232 → 0.225 comparison is the data-matched comparison of the output paths. The full 0.232 → 0.186 gain is cumulative and should not be attributed to direct readout alone. In particular, adding PointOdyssey to the model with auxiliary reconstruction supervision provides a further 12.3% EPE reduction. These experiments are currently limited to the 5B backbone.

G.6 Regression readout: architecture, training and protocol

This section documents the scaled regression model of Section 3.4 (TRACK4D-Reg), whose results are reported in Section 4.5. It is a different model family from the generative RTV/NCMV models

of Table 1 and from the 5B direct-readout probes of Appendix G.5; the three must not be read as one ablation ladder, and the STream3R rows in Tables 1 and 9 report the same evaluation.

Architecture. Figure 8 shows the tensor flow. The RGB video is encoded frame by frame by the frozen Wan2.2 VAE ($\times 1/16$, 48 channels), so that every frame keeps its own latent time step; the normalized point maps are encoded per frame by a 16×16 patch encoder ($3 \to 128 \to 48$ channels) and never pass through the VAE. The two 48-channel maps are concatenated and enter the Wan2.2-TI2V-5B DiT (30 blocks, width 3,072) through its patch embedding, widened from 48 to 96 input channels with the geometry half initialized from the RGB weights; this is the *world expert*, adapted with LoRA of rank 1,024 on the Q/K/V/O and FFN projections. The *motion expert* is a private 30-block transformer (width 1,024, FFN width 4,096) with its own patch embedding, norms, modulation, Q/K/V/O and FFN; its input is the query-frame pair $[x_{\mathcal{V}}; g_{\mathcal{P}}]_0$ repeated at every target time [66]. In each block the queries, keys and values of both groups are projected into the same head space, carry the same 3D rotary position (time, height, width), and are concatenated under one attention softmax [86]; each group then applies its own output projection, cross-attention to the (empty) text context and FFN. Motion tokens after blocks 7, 14, 22 and 29 feed the VGGT-Ω DPT head [87, 15] (256 fusion features, $\times32$ spatial upsampling, zero-initialized output projection so that training starts from the zero-motion predictor), which outputs the residual (identity activation), a confidence and a separate visibility logit. The query-frame residual is supervised like any other frame and is not forced to zero at inference. Inference uses the last training timestep as a fixed regression timestep, with no noise, classifier-free guidance or iterative denoising; the world blocks after the deepest tap cannot reach the readout and their LoRA tensors are frozen. The checkpoint holds 1,475 trained tensors (588 LoRA and patch-embedding, 822 motion expert, 62 DPT, 3 patch encoder); the base Wan weights are loaded unchanged.

Normalization and composition. Each clip's point maps are expressed in the first camera, depth is clamped to $[10^{-6}, 80]$ m, the dense support between the 2nd and 98th depth percentiles is centred on its mean μ and divided by its maximum radius r. With $a_0 = (P_0(t_0) - \mu)/r$ the normalized query point and $\boldsymbol{\delta}_j$ the predicted residual, the metric output is $\hat{P}_0(t_j) = \mu + r(a_0 + \boldsymbol{\delta}_j) = P_0(t_0) + r\,\boldsymbol{\delta}_j$ (Equation (11)). During training the loss composes the residual with the *ground-truth* query point, $G_0 + r\,\boldsymbol{\delta}_j$, whereas deployment composes it with the *estimated* point map; the model is therefore not trained to correct an imperfect anchor, which the oracle analysis of Appendix G.7 identifies as the dominant error source on PointOdyssey and Dynamic Replica.

Training. Windows of 11 frames are sampled in sequential order with per-source maximum frame intervals between 4 and 9 and a reversal probability of 0.25, at three canvases (512×288, 384×384, 288×512). The mixture contains our Kubric split and Kubric-F [30], PointOdyssey [44], Dynamic Replica [45], Waymo [100], DL3DV [101], ScanNet [102] and MVS-Synth [103]; the loss is applied only at annotated pixels (masked MSE on the composed points with a $10\times$ weight on dynamic pixels and a visibility BCE), so sparse sources are used as is. The reported checkpoint is the end of a three-stage, weights-only warm-start lineage (12,750 $\to$ 500 $\to$ 1,000 steps, optimizer and RNG state reset at each boundary) and is not a model trained from scratch for 1,000 steps; all three stages ran on two NVIDIA H200 GPUs, about seven days in total; the final stage uses two processes with a per-process batch of 32 and gradient accumulation 2 (effective batch 128), AdamW with learning rate 5×10^{-5} (minimum 10^{-5}), betas $0.9/0.95$, visibility loss weight 1, and bf16 checkpoints; RGB latent noise and the image/geometry augmentations are disabled in this stage.

Evaluation protocol. Tables 9 and 11 use 50 sequences per benchmark (49 for ADT, whose one sequence without a valid frame-0 query is excluded by the membership rule), the first 49 frames, queries at ground-truth points visible in frame 0 mapped to the model raster by the width/height ratio and sampled at the nearest pixel, and one closed-form $\mathrm{Sim}(3)$ per sequence over all selected tracks and timestamps; APD uses thresholds $\{0.1, 0.3, 0.5, 1.0\}$ m. TRACK4D-Reg runs at 288×512 (384×384 for ADT) and covers the 49 frames with five deterministic frame-0-anchored passes of 10 or 11 frames (frame IDs $\{0, 5, 10, \dots\}$, $\{0, 1, 6, 11, \dots\}$, $\{0, 2, 7, 12, \dots\}$, $\{0, 3, 8, 13, \dots\}$, $\{0, 4, 9, 14, \dots\}$); frame 0 is averaged over the passes and every other frame is predicted once. Each pass is normalized and mapped back to metric coordinates independently before merging; there is no temporal smoothing, per-pass alignment or interpolation. The TrackCraft3R rows replay the official checkpoint and preprocessing at its native 480×832 raster with the same point-map caches and scorer; the comparison therefore matches data membership and scorer but not model input

Table 9: **Regression readout *vs*. TrackCraft3R.** All-point EPE↓ / APD↑ on 50 sequences per benchmark (49 for ADT), using the protocol of Table 1. TrackCraft3R and TRACK4D-Reg share point-map caches; bold marks the better result. V-DPM and 4RC use their own geometry. PointOdyssey and Dynamic Replica are training datasets for TRACK4D-Reg, V-DPM and 4RC. °Oracle diagnostic without retraining: the ground-truth query point replaces the estimated one after prediction, scored under the fixed $\mathrm{Sim}(3)$ of the row above (Table 11), following TrackCraft3R's input-geometry ablation.

Geometry	Method	PStudio	PointOdyssey	Dynamic Replica	ADT
own	V-DPM [22]	0.0883 / 0.9224	0.1403 / 0.8642	0.1914 / 0.8184	0.1045 / 0.9024
	4RC [34]	0.1067 / 0.8936	0.1939 / 0.8347	0.1604 / 0.8664	0.1144 / 0.8894
VGGT-Ω	TrackCraft3R [66]	0.1017 / 0.9031	0.4370 / 0.6524	**0.2239 / 0.7783**	0.0935 / **0.9245**
	TRACK4D-Reg (ours)	**0.0947 / 0.9155**	**0.4127 / 0.6668**	0.2257 / 0.7777	**0.0933** / 0.9230
	TRACK4D-Reg + GT query geometry°	0.0900 / 0.9169	0.0531 / 0.9450	0.0275 / 0.9743	0.0374 / 0.9582
DAv3-L	TrackCraft3R [66]	0.1347 / 0.8491	0.5224 / 0.5064	0.2859 / 0.7134	0.4002 / 0.6187
	TRACK4D-Reg (ours)	**0.1270 / 0.8625**	**0.3536 / 0.6616**	**0.2836 / 0.7182**	**0.1400 / 0.8498**
	TRACK4D-Reg + GT query geometry°	0.1146 / 0.8761	0.0970 / 0.8944	0.0337 / 0.9666	0.0555 / 0.9401
STream3R	TrackCraft3R [66]	0.1400 / 0.8418	0.2227 / 0.7904	0.2289 / 0.7839	**0.1163 / 0.8874**
	TRACK4D-Reg (ours)	**0.1311 / 0.8572**	**0.2078 / 0.8050**	**0.2284 / 0.7871**	0.1168 / 0.8854

raster. The V-DPM and 4RC rows replay the official checkpoints with their own geometry pipelines under the same membership and scorer (49 frames, query frame 0), so they are references for the absolute level of the benchmark rather than a controlled comparison. The geometry caches come from VGGT-Ω [15], DAv3-L (Depth Anything 3, Large) [14] and STream3R [13].

Comparison with the generative models. Table 10 scores TRACK4D-Reg with STream3R geometry, the scaffold of Table 1, against the two generative models of the paper under the same scorer: the Wan2.2-5B ablation baseline of Table 2 and the joint Wan2.1-I2V-14B model of Table 1 (1-step dynamic values of Table 7; with the 50-step values of Table 1 the ranking below is unchanged). TRACK4D-Reg is better than both on PStudio, PointOdyssey and Dynamic Replica and is close to the 14B model on ADT, with a 5B backbone and one forward pass per window. The comparison is not controlled: the generative models run at a 256-pixel short side and see only Kubric and TartanAir, whereas TRACK4D-Reg runs at 288×512 and trains on eight sources including PointOdyssey and Dynamic Replica, so PStudio and ADT are the only zero-shot cells for all three.

Table 10: **TRACK4D-Reg against the generative models with the same STream3R scaffold.** Dynamic EPE↓ / APD↑ under the WorldTrack scorer (50 sequences, 49 frames; ADT 49); the last column reports all-point values of TRACK4D-Reg. Bold marks the best dynamic value. See the text for the differences in raster and training data.

Benchmark	Wan2.2-5B (RTV)	Wan2.1-14B joint (RTV)	TRACK4D-Reg	TRACK4D-Reg, all points
PStudio	0.2231 / 0.7281	0.1511 / 0.8240	**0.1311 / 0.8572**	0.1311 / 0.8572
PointOdyssey	0.3792 / 0.5939	0.2856 / 0.6931	**0.2691 / 0.7155**	0.2078 / 0.8050
Dynamic Replica	0.2536 / 0.7009	0.1861 / 0.7850	**0.1674 / 0.8141**	0.2284 / 0.7871
ADT	0.0722 / 0.9370	**0.0666 / 0.9466**	0.0675 / 0.9430	0.1168 / 0.8854

G.7 Geometry-versus-motion oracle

Table 11 reports the substitutions discussed in Section 4.5 on all primary benchmarks and both geometry backends, together with the secondary ADT benchmark and with refitted scores. After applying the original $\mathrm{Sim}(3)$ S to the prediction, write $p = S(\hat{P})$, $a = p_0 - G_0$ for the anchor error at the query frame and $m = (p_j - p_0) - (G_j - G_0)$ for the relative-motion error.

For every retained observation,

$$\|p_j - G_j\|^2 = \|a + m\|^2 = \|a\|^2 + \|m\|^2 + 2\, a^\top m, \tag{12}$$

which Table 12 reports as sequence-macro means. On PointOdyssey and Dynamic Replica the anchor term dominates the budget for both geometry backends; on PStudio the two terms are comparable and the negative cross term cancels a substantial part of either, so a motion oracle can even raise the fixed-alignment EPE (DAv3-L). The geometry oracle remains backend-dependent: PointOdyssey EPE is 0.053 with VGGT-Ω and 0.097 with DAv3-L.

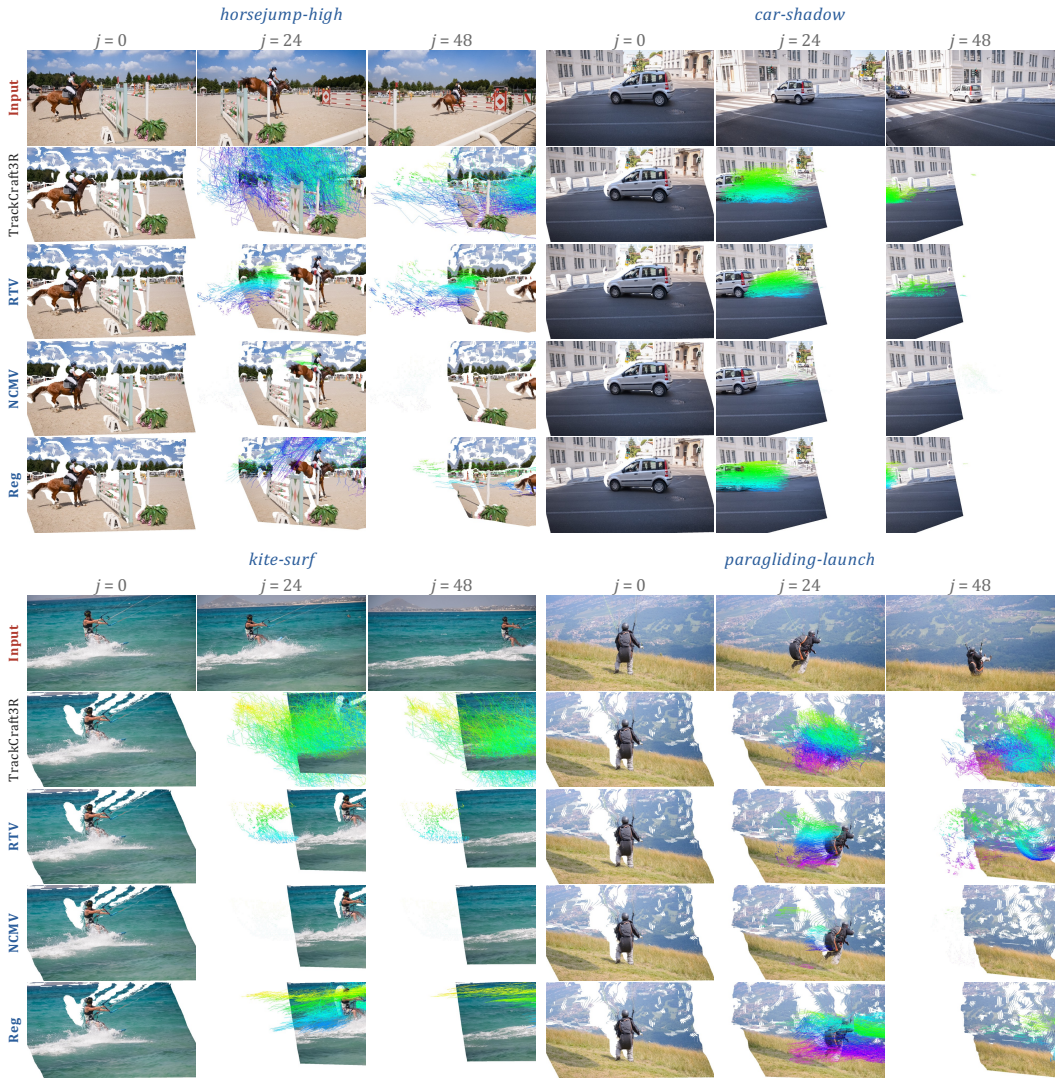


Figure 9: **More DAVIS clips, TrackCraft3R *vs*. the three TRACK4D readouts.** Same setup as Figure 6. Because the view is fixed at frame 0, subjects that the camera follows leave the view in later frames (*car-shadow*, *kite-surf*, *parkour*). Over all 91 DAVIS clips, the median second difference of the projected tracks over consecutive frames, counting points inside the frame, is 6.5 px for TrackCraft3R and 1.8 px for TRACK4D-Reg (672-px-wide frames), and it is smaller for TRACK4D-Reg on 90 clips. This measures temporal smoothness, not accuracy.

Swapping the geometry backend. The same conclusion follows from replaying the scorecard checkpoint with STream3R [13] point maps computed from 256-pixel-short-side frames, the scaffold of Table 1, under the unchanged protocol (Table 9). PointOdyssey EPE halves ($0.413 \rightarrow 0.208$, APD $0.667 \rightarrow 0.805$) with no change to the tracker, Dynamic Replica changes little ($0.226 \rightarrow 0.228$), and PStudio and ADT get worse ($0.095 \rightarrow 0.131$, $0.093 \rightarrow 0.117$); the official TrackCraft3R checkpoint shows the same pattern on PointOdyssey ($0.437 \rightarrow 0.223$). No single backend is best on every benchmark, and the pattern is not specific to the reported weights: Table 13 repeats the swap for two earlier checkpoints of the lineage with the same outcome. These are backend-conditioned comparisons of complete systems: they show that performance depends on the supplied geometry, not that the anchor alone explains every difference between backends. Since the closed-form $\mathrm{Sim}(3)$ minimizes squared distances rather than EPE or APD, a refit can score a higher EPE than the fixed alignment, as on PointOdyssey and Dynamic Replica; neither is a bound, and we report both.

Selected ADT failures of TrackCraft3R. The large DAv3-L/ADT gap in Table 9 was examined on three sequences, two without dynamic queries and one mixed. Independently aligned query errors of

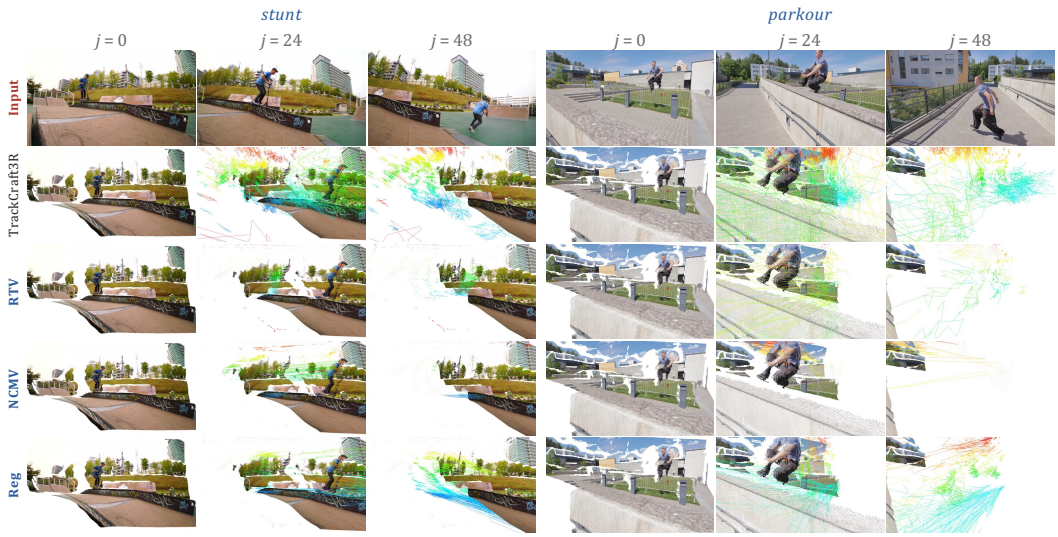


Figure 10: **More DAVIS clips (continued).** Same setup as Figures 6 and 9.

Table 11: **Full geometry/motion oracle for TRACK4D-Reg.** EPE↓ / APD↑ on all 50 sequences of each primary benchmark (ADT, 49 sequences, is secondary). *Fixed* columns score the substituted output under the original all-track $\mathrm{Sim}(3)$ of the current prediction; *refit* columns fit a new closed-form $\mathrm{Sim}(3)$ to each substituted output. Closed-form $\mathrm{Sim}(3)$ minimizes squared distances rather than EPE or APD, so refitting can raise EPE; neither column gives a lower bound.

			Fixed $\mathrm{Sim}(3)$		**Refit** $\mathrm{Sim}(3)$	
Geometry	**Benchmark**	**Current**	GT query geometry	GT motion	GT query geometry	GT motion
VGGT-Ω	PStudio	0.0947 / 0.9155	0.0900 / 0.9169	0.0900 / 0.9239	0.0763 / 0.9415	0.0672 / 0.9492
	PointOdyssey	0.4127 / 0.6668	0.0531 / 0.9450	0.4035 / 0.6787	0.0567 / 0.9518	0.4017 / 0.6810
	Dynamic Replica	0.2257 / 0.7777	0.0275 / 0.9743	0.2207 / 0.7850	0.0344 / 0.9764	0.2202 / 0.7851
	ADT (secondary)	0.0933 / 0.9230	0.0374 / 0.9582	0.1056 / 0.9105	0.0261 / 0.9745	0.0789 / 0.9459
DAv3-L	PStudio	0.1270 / 0.8625	0.1146 / 0.8761	0.1289 / 0.8614	0.0933 / 0.9147	0.1011 / 0.9050
	PointOdyssey	0.3536 / 0.6616	0.0970 / 0.8944	0.3418 / 0.6719	0.0994 / 0.8979	0.3207 / 0.6980
	Dynamic Replica	0.2836 / 0.7182	0.0337 / 0.9666	0.2806 / 0.7230	0.0426 / 0.9682	0.2799 / 0.7252
	ADT (secondary)	0.1400 / 0.8498	0.0555 / 0.9401	0.1508 / 0.8388	0.0444 / 0.9615	0.1151 / 0.8834

Table 12: **Exact squared-error budget of TRACK4D-Reg.** Sequence-macro mean squared error in m^2 under the original $\mathrm{Sim}(3)$, decomposed as in Equation (12); the identity holds to floating-point precision. The cross term is large and negative on PStudio, so anchor and motion errors must not be read as additive EPE contributions.

Geometry	**Benchmark**	**Total**	**Anchor**	**Motion**	**Cross**
VGGT-Ω	PStudio	0.0219	0.0197	0.0200	−0.0178
	PointOdyssey	0.6349	0.6299	0.0127	−0.0076
	Dynamic Replica	0.2103	0.2074	0.0060	−0.0031
DAv3-L	PStudio	0.0307	0.0335	0.0280	−0.0308
	PointOdyssey	0.3784	0.3658	0.0422	−0.0297
	Dynamic Replica	0.2560	0.2558	0.0090	−0.0088

the two systems are comparable, but the TrackCraft3R replay produces much larger within-window displacement for static queries; removing its per-window query correction does not change its fixed-alignment EPE materially, so the spurious motion lives in the temporal output rather than in the anchor. This locates the behavior; it does not identify the VAE, the DiT or the decoder as its cause, since no component-level ablation was run. A residual numerical difference between the historical predictions and an A100 replay (largest selected-case EPE change $0.00035\,\mathrm{m}$) remains unresolved and is small relative to the performance gap above.

Table 13: **Backend swap across checkpoints of the TRACK4D-Reg lineage.** EPE↓ / APD↑ on all points using VGGT-Ω or STream3R point maps, with fixed weights within each checkpoint. The reported checkpoint follows Table 9; inference settings for the earlier checkpoints have not been fully verified, so cross-checkpoint differences are descriptive. Across checkpoints, STream3R roughly halves PointOdyssey EPE; Dynamic Replica changes little, while PStudio and ADT prefer VGGT-Ω.

Checkpoint	Geometry	PStudio	PointOdyssey	Dynamic Replica	ADT
parent run, step 13,750	VGGT-Ω	0.2317 / 0.7197	0.4337 / 0.6426	0.2436 / 0.7550	0.1030 / 0.9106
	STream3R	0.2420 / 0.7082	0.2350 / 0.7693	0.2476 / 0.7611	0.1234 / 0.8764
recovery stage, step 7,500	VGGT-Ω	0.0988 / 0.9080	0.4146 / 0.6648	0.2272 / 0.7758	0.0995 / 0.9157
	STream3R	0.1338 / 0.8527	0.2103 / 0.8016	0.2300 / 0.7848	0.1183 / 0.8841
recovery stage, step 1,000 (reported)	VGGT-Ω	0.0947 / 0.9155	0.4127 / 0.6668	0.2257 / 0.7777	0.0933 / 0.9230
	STream3R	0.1311 / 0.8572	0.2078 / 0.8050	0.2284 / 0.7871	0.1168 / 0.8854

G.8 Source and spatial distribution of geometric artifacts

We investigate three possible sources of the flying-point artifacts: the point-map scaffold, RTV residual prediction, and the video VAE codec. All diagnostics use the same fixed, scene-disjoint subset of Kubric val10 and evaluate outputs in metric 3D space without $\mathrm{Sim}(3)$ alignment, so their absolute values are not comparable with Table 1. For the spatial analysis, edge pixels are the top 10% of ground-truth local 3D discontinuities, dilated by two pixels.

Controlled stage-wise diagnostics. We replace the other stages with ground-truth inputs to estimate the error associated with each component (Table 14). These diagnostics do not form an additive decomposition: stages interact, and predicted residuals can partly correct scaffold errors.

Table 14: **Controlled stage-wise diagnostics of geometric error.** Metric-space EPE on a fixed Kubric val10 subset. The arms isolate different sources and are not additive.

Diagnostic arm	Isolates	All EPE ↓
Predicted point-map scaffold	scaffold error	≈ 2.34
Oracle scaffold + predicted RTV residual	residual-prediction error	≈ 0.84
Ground-truth target → VAE round-trip	VAE codec floor	≈ 0.79
Full pipeline	all stages combined	≈ 2.12

The scaffold has the largest isolated error. With an oracle scaffold, the predicted residual contributes approximately 0.84 EPE, while a VAE round-trip of the ground-truth target produces an approximately 0.79 EPE floor. The full pipeline slightly improves over the predicted scaffold alone, consistent with the residual correcting some scaffold error.

Boundary localization and metric impact. Passing ground-truth targets through the VAE without diffusion yields approximately $2.5\times$ higher reconstruction error near ground-truth 3D discontinuities than in interior regions. We further compare the VAE-decoded model with a data-matched direct-readout model trained on the same Kubric tracking data, without auxiliary depth/camera supervision or PointOdyssey (Table 15). Both rows of Table 15 are these data-matched training runs, so their absolute EPE differs from the full-pipeline arm of Table 14.

Table 15: **Metric impact of removing the VAE output path.** Metric-space EPE on the fixed Kubric val10 subset.

Output path	All EPE ↓	Edge EPE ↓	Interior EPE ↓
VAE-decoded	1.340	1.479	1.300
VAE-free direct readout	**0.859**	**1.331**	**0.728**
Relative reduction	35.9%	10.0%	44.0%

In this data-matched comparison, replacing the VAE-decoded output path with direct readout reduces overall EPE by 35.9%; with the round-trip floor above, this indicates that the codec meaningfully affects the reported metric. However, the gain is much larger in interior regions (44.0%) than at boundaries (10.0%), and the direct-readout model still has approximately $1.8\times$ higher edge EPE than interior EPE. The evidence therefore identifies the scaffold as the main source of global geometric

error and the VAE as a secondary error floor that accentuates boundary-localized artifacts; neither the VAE nor residual prediction alone explains all flying points.

G.9 Camera-following-object failure analysis

The failure case in Figure 11 raises the possibility that the log-space transform in Equation (5), including the scale factor K, compresses genuine deformation toward zero. We audit this hypothesis on 50 Kubric clips in which the camera follows a moving object, comparing the true world-space displacement with the normalized target. Substantial motion remains clearly separated from zero after normalization; for example, a clip with approximately $9\,\mathrm{m}$ of object displacement retains a strongly non-zero normalized target. The audit therefore does not support the specific hypothesis that log-space normalization universally erases the deformation signal.

This result does not by itself prove dataset bias. Because the generative model receives dynamic supervision only from our Kubric subset, while other 3D trackers combine Kubric with datasets such as Dynamic Replica and PointOdyssey, insufficient coverage of camera-following-object trajectories is a plausible explanation. This hypothesis remains unverified. Identifying the precise stage at which the model suppresses the preserved motion signal requires tracing the normalized target, model prediction, and decoded output on the same clips.

G.10 Positioning relative to recent feed-forward methods

Several recent methods approach dense dynamic reconstruction through direct feed-forward geometric regression. POMATO [41] combines point-map matching with temporal motion in a DUSt3R-style framework. Zero-MSF [42] jointly predicts geometry and scene flow and scales training to approximately one million annotated synthetic samples. Track4World [43] uses a VGGT-style video encoder and 3D correlation to regress pairwise dense 2D/3D flows that are composed into world-centric tracks. The concurrent D4RT [24] encodes a video into a global representation and decodes flexible spatio-temporal point queries, while the concurrent 4RC [34] encodes a video once and conditionally queries dense geometry and motion at arbitrary times.

These deterministic regressors are complementary to TRACK4D. They optimize geometric outputs directly, whereas our primary question is how a pretrained generative video prior can represent and produce dense 3D tracking signals. The direct-readout experiment in Appendix G.5 begins to connect the two categories by retaining the generative video backbone while replacing the RGB-oriented output codec with a geometric regression head. Concurrent work on general-purpose perception from video-generative pretraining [104] and omnimodal world models that combine understanding, generation, simulation, and action [105] further motivates studying tracking within shared generative backbones. A similar distinction exists in depth estimation, where generative approaches [27, 61] are studied as a complementary category alongside mature discriminative regressors. The performance gap to mature specialist regressors remains an important limitation.

H More results and discussion

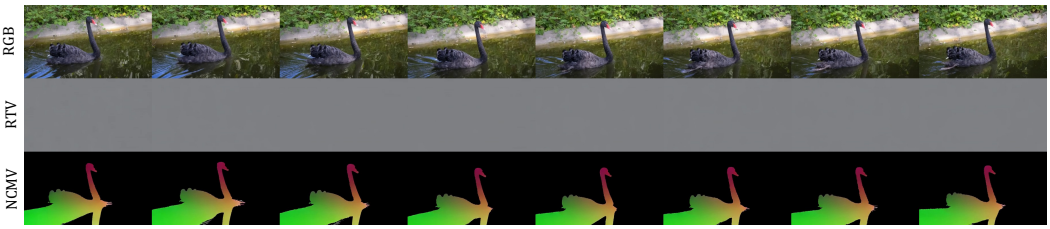


Figure 11: **Failure case.** TRACK4D predicts near-zero deformation in this video, where the camera moves with the object. Limited coverage of such camera-following-object trajectories in the training data is a plausible, but not conclusively established, cause (Appendix G.9).

Decoding tracks from NCMV: CoTracker vs. nearest-neighbor. Replacing CoTracker with nearest-neighbor color matching and a Kalman filter reduces APD from 0.4798 to 0.3174 on

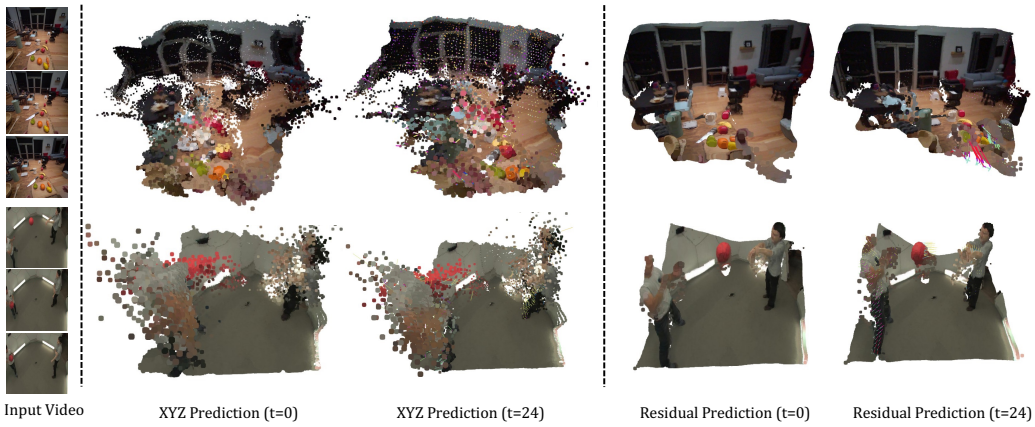


Figure 12: **Ablation: absolute XYZ prediction vs. residual prediction.** We visualize 3D tracking results at $t = 0$ and $t = 24$ on ADT [89] and PStudio [90] sequences. Left: predicting absolute 3D coordinates directly (*w/o Residual Pred*). Right: our residual formulation (Equation (4)). Absolute prediction produces noisy geometry with scattered outliers, whereas the residual formulation yields cleaner reconstructions with more coherent motion.

PointOdyssey and from 0.7495 to 0.5052 on PStudio. Thus, finite-resolution NCMV colors alone are insufficient for reliable matching; the 2D tracker materially improves decoding.

Additional visualizations. Figure 14 shows additional zero-shot DAVIS results; Figure 13 compares Wan 5B and 14B on the same examples, where 14B produces sharper motion.

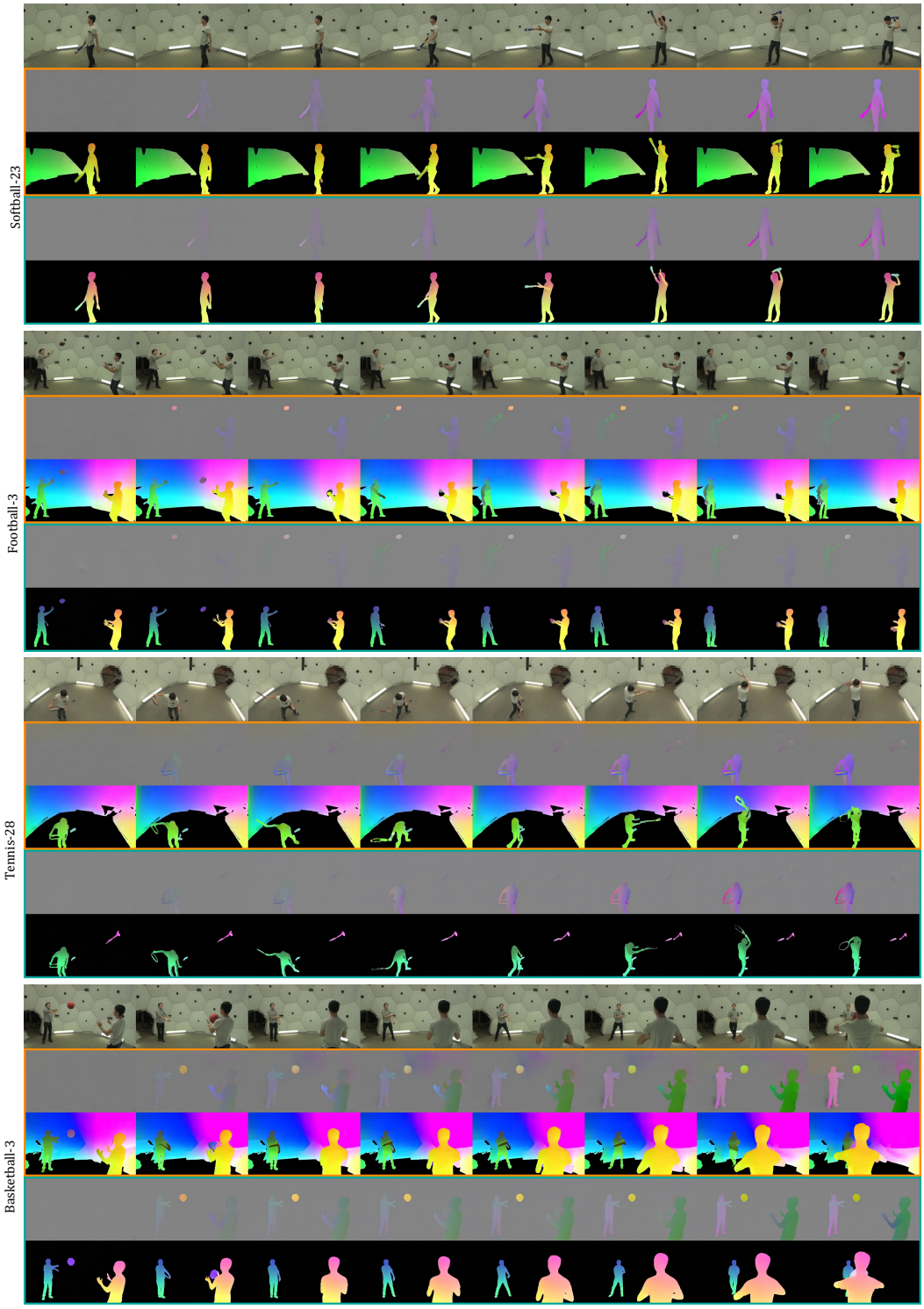


Figure 13: **Effect of backbone size.** Specifically, we show our model's performance on four samples from PStudio [90], with sequence names on the left. For each sequence, we compare the results from TRACK4D-5B (in orange) and TRACK4D-14B (in green). Both RTV and NCMV are visualized. The Wan-14B-based TRACK4D shows better foreground-background factorization and sharper motion, suggesting benefits from stronger video diffusion models.

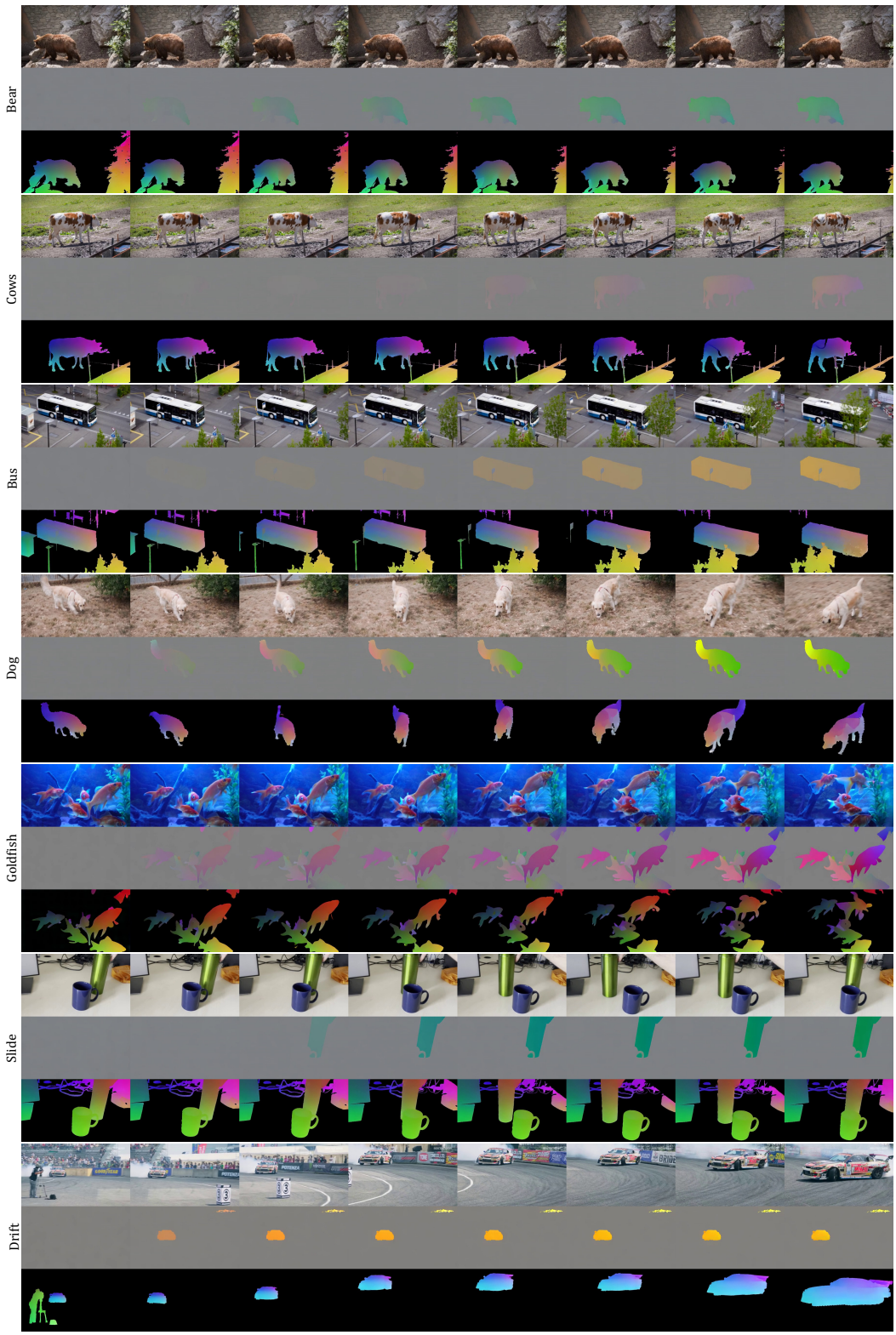


Figure 14: **More qualitative results.** We show seven zero-shot inference results from DAVIS [70]. The first row shows RGB input, with eight frames sampled uniformly from a 49-frame clip, and the second and third rows show RTV and NCMV predictions. The query frame is the first frame shown in the first row. RTV encodes 3D offsets on the query-frame grid; NCMV encodes persistent coordinates on each video frame's grid.

I Broader impacts

Dense 3D tracking from monocular video can support robotics, 4D reconstruction, video editing, and scientific motion analysis. Like other motion-tracking methods, it could also be used to follow people in recorded video; TRACK4D is trained on synthetic and public research datasets, but improved tracking can still increase privacy risks. Its outputs are estimates conditioned on an external reconstruction and can be wrong, particularly for content that is occluded, enters after the query frame, or moves with the camera (Appendix G.9); they require validation before use as measurements.

J Existing assets and licenses

Table 16 lists the external datasets, models and baselines, how we use them, and their licenses. Code, the adapters and the re-rendered Kubric split will be released at `https://github.com/NIRVANALAN/Track4D`.

Table 16: **Existing assets and their licenses**, as stated by the official repositories and dataset pages (checked October 2026). Data and model weights may carry terms beyond the code license; we use every asset for non-commercial research only.

Asset	Use	License / terms
Kubric [30]	Rendering our training split	Apache-2.0 (code)
TartanAir [88]	Training (generative)	CC BY 4.0
PointOdyssey [44]	Training (regression), evaluation	CC BY-NC-SA 4.0 (data); MIT (code)
Dynamic Replica [45]	Training (regression), evaluation	CC BY-NC (repository); data after accepting the license agreement
Waymo Open Dataset [100]	Training (regression)	Waymo Open Dataset terms; Apache-2.0 (code)
DL3DV-10K [101]	Training (regression)	DL3DV-10K terms of use and CC BY-NC 4.0
ScanNet [102]	Training (regression)	ScanNet Terms of Use; MIT (code)
MVS-Synth [103]	Training (regression)	Official terms; BSD-2-Clause (DeepMVS code)
ADT, PStudio (TAPVid-3D)	Evaluation	Aria Digital Twin license and Panoptic Studio terms; Apache-2.0 (TAPVid-3D code)
DAVIS [70]	Qualitative results	Official terms
Wan2.1, Wan2.2 [26]	Backbones	Apache-2.0
VGGT-Ω [15]	Input geometry	FAIR Noncommercial Research License
STream3R [13]	Input geometry	S-Lab License 1.0 (non-commercial)
Depth Anything 3 [14]	Input geometry	Apache-2.0 (code); CC BY-NC 4.0 (Large/Giant weights)
CoTracker [75]	NCMV decoding, baseline	CC BY-NC 4.0
TrackCraft3R [66]	Baseline	Apache-2.0
V-DPM [22]	Baseline	MIT
4RC [34]	Baseline	S-Lab License 1.0 (non-commercial)
Any4D [23]	Baseline	Apache-2.0
TraceAnything [31]	Baseline	Apache-2.0
SpaTrackerV2 [17]	Baseline	CC BY-NC 4.0
St4RTrack [19]	Baseline	No license file in the official repository

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 Section 1 state the contributions: a study of how to represent dense 3D tracking inside a pretrained video diffusion model, the comparison of the RTV and NCMV readouts, and a regression readout of the same residual without the video VAE. They are supported by Tables 1, 2, 7 and 9 and Section 4.5; the scope (49-frame clips, point maps from an off-the-shelf reconstructor, frame-0 queries) is stated in Sections 3 and 4.

   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: Section 5 and Appendix A state the limitations: the dense-target requirement of the generative readouts, the fixed 49-frame window, frame-0 queries, and the dependence on the supplied geometry (quantified by the oracle and backend swaps of Appendix G.7). The appendix analyses the VAE boundary artifacts (Appendix G.8), the camera-following failure (Appendix G.9, Figure 11) and the inference cost (Appendix G.3).

   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 is empirical and presents no theorems requiring formal proof.

   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: Section 3 defines the representations and Section 3.4 the regression readout. Appendices C and D give the training data, LoRA configuration, optimizer, batch size, iterations and hardware of the generative model; Appendix G.6 gives the architecture, normalization, training mixture, schedule and the 49-frame protocol of TRACK4D-Reg; Appendix E gives the comparison protocol of every baseline.

   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: The LoRA adapters are public at `https://huggingface.co/yslan/Track4D-2026`; the code and the re-rendered Kubric split will be released at `https://github.com/NIRVANALAN/Track4D`. The appendix describes data, training and evaluation in enough detail for re-implementation.

   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: Appendix D specifies LoRA rank, learning rate, batch size, training length, inference noise schedule, adapted layers, frozen modules and domain tokens of the generative model; Appendix G.6 gives the same for TRACK4D-Reg, including the training mixture and the three-stage warm-start lineage. Appendix E documents the protocol used for each baseline.

   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 3D point tracking benchmarks, we report deterministic APD/EPE averaged over 50 sequences per benchmark (49 for ADT in Table 9) without error bars. The joint-versus-RTV-only comparison uses one checkpoint pair with different training histories and sampling settings, and claims no significance (Appendix G.4).

   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: The joint Wan2.1-I2V-14B model underwent 5,000 joint-training iterations on 32 NVIDIA GH200 GPUs for about one day (about 770 GPU-hours; Appendix D); TRACK4D-Reg was trained on two NVIDIA H200 GPUs for about seven days over its three stages (about 340 GPU-hours; Appendix G.6). Inference cost is reported in Appendices F and G.3: about 14 s per 49-frame clip for the sampling-ablation model at 50 steps on an A6000, and 6.5 s with 26.0 GB reserved memory for TRACK4D-Reg on an A100, excluding the input point maps (Table 6).

   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. The paper uses publicly released datasets and pretrained models, involves no human-subject studies, and evaluates tracking on existing datasets.

   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: Appendix I discusses positive and negative impacts. The paper proposes a research method for 3D motion estimation, with applications in robotics, video understanding and 4D reconstruction. As with any motion-tracking technology, downstream uses could include surveillance; improved tracking can increase privacy risks even without a specific deployment.

    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: The released artifacts are a LoRA adapter for a publicly available video diffusion model and a re-rendering of a synthetic dataset (Kubric). They do not pose a high misuse risk beyond that already present in the underlying public assets.

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: Table 16 (Appendix J) lists every external asset with its license. All datasets and pretrained models (Kubric, TartanAir, PointOdyssey, Dynamic Replica, Waymo, DL3DV, ScanNet, MVS-Synth, Aria Digital Twin, Panoptic Studio, DAVIS, Wan2.1, Wan2.2, VGGT-Ω, STream3R, Depth Anything 3, CoTracker, and the baselines' official checkpoints) are cited at first use and used under their released research licenses.

    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 will release at `https://github.com/NIRVANALAN/Track4D` the re-rendered Kubric split with dense 3D tracking annotations, the rendering scripts, and the trained adapters of the generative and regression models, with documentation of the data format, training commands and inference protocol of Appendices D and G.6.

    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: The work involves no crowdsourcing or research with human subjects.

    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: The work involves no human subjects, so IRB review 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 part of the core method. They were used only as writing assistants for editing and copy-editing, which does not require declaration under the NeurIPS LLM policy.

    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.