TRACKDIT: 3D TRACKING FROM 2D TRACKS AND 3D RECONSTRUCTION

Anonymous authors
Paper under double-blind review

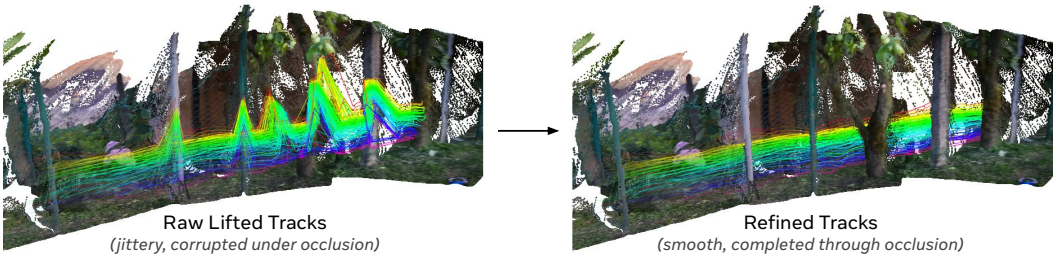


Figure 1: **Lifted 2D tracks break under occlusion; TrackDiT completes them.** Composing a 2D tracker with a 3D reconstructor gives accurate 3D tracks wherever a point is visible, but the raw lift is jittery, and at occluded frames the depth map returns the *occluder's* surface, so the tracks spike toward the camera at every tree trunk they pass behind (*left*). TrackDiT generates a correction to the lift that removes these excursions and carries each trajectory smoothly through the occlusions (*right*).

ABSTRACT

Recent 3D point trackers adopt architectures that jointly learn appearance matching, 3D reconstruction and temporal reasoning in a single yet complex end-to-end model. By evaluating a simple baseline that lifts 2D tracks into 3D using an off-the-shelf 2D tracker and 3D reconstructor, we derive two insights: (i) such a baseline outperforms specialized 3D tracking models on always-visible tracks, and (ii) when tracks are partially occluded, specialized models surpass such a baseline. This observation motivates a new paradigm: rather than training one network for everything, we first obtain precise visible track fragments from foundation models, then treat the remaining occlusion gaps as a track completion problem, addressed by a flow-matching model that generates the correction to the lifted trajectory. Because a hidden point's motion is genuinely ambiguous, a generative formulation is a better fit than regression, and operating on track-frame tokens lets the model complete a gap both from the track's own motion and from the co-visible points around it. The resulting framework is modular, simple to train, and automatically improves as its foundation-model components advance. TrackDiT achieves state-of-the-art 3D tracking results on ADT, PStudio, and SynthVerse. In particular, on occluded points it improves over the lifting baseline by 12.2 APD_{3D} points and surpasses specialized models, while also improving accuracy on visible points.

1 INTRODUCTION

Tracking 3D points in videos is an important task in video understanding, with countless applications ranging from robotics and autonomous driving to video editing. The dominant paradigm in 3D tracking consists of training monolithic neural networks that jointly reason over appearance, geometry, and temporal correspondences in a single 4D architecture (Sucar et al., 2026; Han et al., 2025; Zhang et al., 2025b; Luo et al., 2026; Liu et al., 2026; Karhade et al., 2025). While these end-to-end models have achieved impressive benchmark results, they also have some disadvantages: they require large-scale 4D training data, use significant compute, and entangle subproblems such as 2D tracking

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

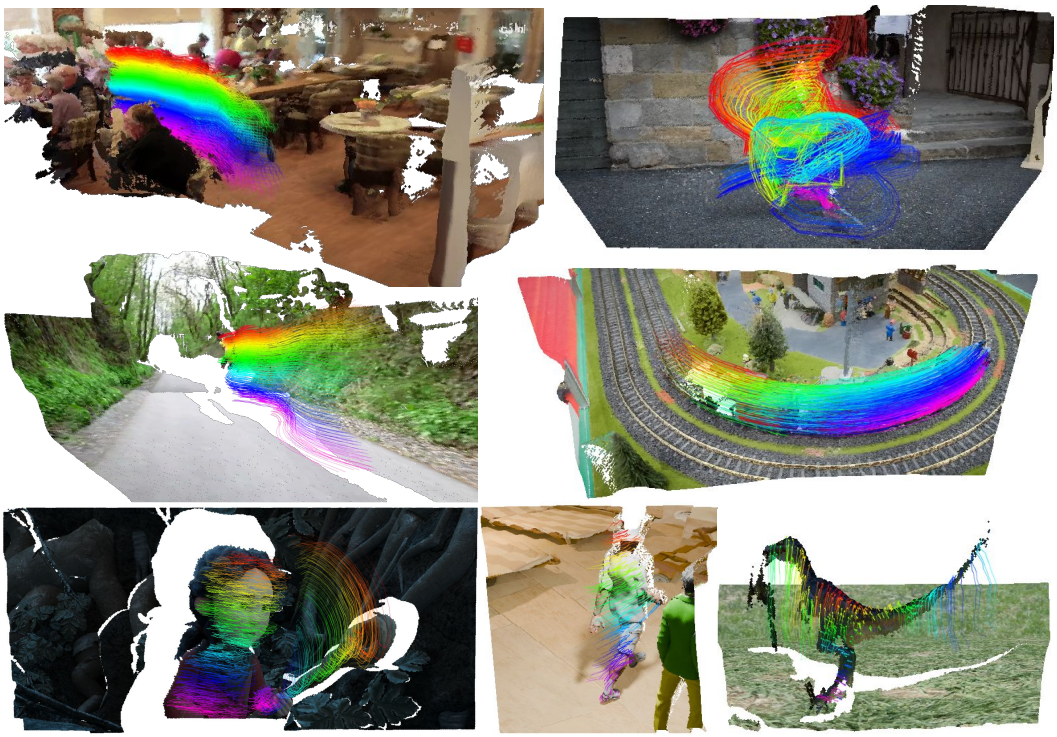


Figure 2: **Qualitative 3D point tracking results.** 3D point trajectories reconstructed by TrackDiT, spanning real (*top four*) and synthetic (*bottom three*) videos, as well as rigid scene structure and non-rigid motion.

and 3D reconstruction in an opaque manner. The latter makes it difficult to analyse failure modes and take advantage of new advances in tracking and 3D reconstruction.

An alternative is to *decompose* 3D tracking into 2D tracking and 3D reconstruction, using a state-of-the-art 2D point tracker to establish pixel-level correspondence and a 3D reconstructor to lift tracks into 3D. Such a decomposed pipeline is simple, requires no joint training, and automatically benefits from future improvements in either component. The question is whether this modular approach can compete with end-to-end models.

In this paper, we find surprising answers to this question. First, we show that existing end-to-end 3D trackers are in fact *not* good 2D trackers, significantly underperforming state-of-the-art 2D trackers when 3D tracks are projected back to 2D (section 3). This is a simple but important observation which suggests that end-to-end 3D trackers are not as well engineered as their 2D counterparts when it comes to establishing accurate pixel-level correspondences.

This strongly suggests that end-to-end 3D tracker architectures should be revisited, or that one should consider alternative approaches that leverage the best available 2D trackers. Here, we explore the latter idea and find that, as long as a track remains *visible*, composing an off-the-shelf 2D tracker with a depth and camera reconstructor already matches or exceeds representative end-to-end 3D trackers. On the other hand, end-to-end 3D trackers clearly outperform this baseline when the track becomes *occluded*, as in this case knowledge of depth is insufficient to lift the 2D track to 3D.

This finding suggests a simple, modular design for 3D tracking that builds on this baseline. The idea is to take the resulting 3D tracks and to use a new model to *complete 3D tracks behind occlusions*. This paradigm offers three key advantages: (1) it inherits the accuracy of the 2D tracker in terms of establishing pixel-level correspondences; (2) it allows one to focus on addressing the harder problem of 3D tracking behind occlusions; and (3) it takes advantage of future improvements to foundation 2D tracking and 3D reconstruction models out of the box.

We implement this idea in TrackDiT. The new technical component is a model that outputs a correction for 3D tracks obtained by lifting 2D ones to 3D, as explained above. Because the location of a track behind occlusion is ambiguous, we implement this model as a generative transformer

trained with flow matching. This transformer uses factorized spatio-temporal attention over tracks and time, which allows visible tracks to provide cues for completion of the occluded ones, for instance when these are located on the same underlying 3D object. TrackDiT achieves state-of-the-art 3D tracking results on the St4RTrack / 4RC benchmark.

Overall, our contributions are: (i) To show that current end-to-end 3D trackers are not very good 2D trackers; (ii) To show that trivially combining a state-of-the-art 2D tracker and 3D reconstructor already matches or exceeds the performance of these models except when tracked points become occluded; (iii) A reformulation of 3D tracking that builds on any off-the-shelf 2D tracker and 3D reconstructor, learning to handle occluded points as well; (iv) A concrete instantiation of this idea that sets a new state of the art on 3D point tracking.

2 Related Work

Our work concerns two families of methods, namely point trackers and feed-forward 3D reconstructors, discussed below.

2D and 3D point tracking. Modern 2D trackers usually follow the tracking-any-point (TAP) setting (Harley et al., 2022; Doersch et al., 2022). Among recent trackers, PIPs (Harley et al., 2022) revisits the classic Particle Video formulation (Sand & Teller, 2008) with neural networks, also building on ideas from RAFT for optical flow estimation (Teed & Deng, 2020), including iterative track refinement. TAP-Vid (Doersch et al., 2022) introduced further benchmarks for the TAP problem and TAPIR (Doersch et al., 2023) combined global matching with local iterative refinement to deliver a substantial accuracy improvement. PointOdyssey and PIPs++ (Zheng et al., 2023) further introduced harder synthetic data. CoTracker and CoTracker3 (Karaev et al., 2024; 2025) model the correlation between different tracks, which is especially helpful with occlusions, and DOT (Moing et al., 2024) densifies their predictions. BootsTAPIR (Doersch et al., 2024) improves TAPIR through large-scale self-training. TAPTR (Li et al., 2024), inspired by DETR (Carion et al., 2020), formulates tracking with decoder queries; LocoTrack (Cho et al., 2024) simplifies the correlation pipeline with 4D correlation volumes; and TAPVid-3D (Koppula et al., 2024), TAPIP3D (Zhang et al., 2025a), TAPNext (Zholus et al., 2025), DELTA (Ngo et al., 2024), SceneTracker (Wang et al., 2024a), SpatialTrackerV2 (Xiao et al., 2025), and AllTracker (Harley et al., 2025) investigated other extensions, including in some cases 3D tracking. In this work, we build on CoWTracker (Lai et al., 2026), the current state-of-the-art in dense 2D tracking.

End-to-end 3D reconstruction and tracking. Recent feed-forward models extend models such as DUSt3R (Wang et al., 2024b), VGGT (Wang et al., 2025a), Depth Anything (DA) (Yang et al., 2024a), DA3 (Lin et al., 2025), CUT3R (Wang et al., 2025b), π^3 (Wang et al., 2025c), MapAnything (Keetha et al., 2026) to predict, in addition to cameras and depth maps, dense correspondences, i.e., point tracks. Examples include V-DPM (Sucar et al., 2026), D^2USt3R (Han et al., 2025), POMATO (Zhang et al., 2025b), St4RTrack (Feng et al., 2025), D4RT (Zhang et al., 2026), 4RC (Luo et al., 2026), Trace Anything (Liu et al., 2026) and Any4D (Karhade et al., 2025). These models understand 3D scene holistically, both from the viewpoint of 3D camera and structure reconstruction as well as dense tracking of static and dynamic objects.

3 3D Point Tracking: Modular vs End-to-end

We ask whether models like V-DPM (Sucar et al., 2026), 4RC (Luo et al., 2026), Trace Anything (Liu et al., 2026), and Any4D (Karhade et al., 2025) that perform 3D reconstruction and tracking end-to-end can, in fact, be outperformed by modular designs where these two functions are tackled by separate models. To answer this question, we consider one such design, which uses a state-of-the-art 2D point tracker to obtain 2D tracks and a reconstruction foundation model to lift them to 3D, and compare it with end-to-end models on both real (ADT (Pan et al., 2023), PStudio (Joo et al., 2019)) and synthetic (SynthVerse (Zhao et al., 2026)) data.

Observation 0: End-to-end 3D Trackers Are Not Good 2D Trackers. We first compare end-to-end models with the 2D tracker alone. Since a 3D track can be projected back into the image, an end-to-end 3D tracker is also a 2D tracker, and one might expect it to be at least as accurate as a dedicated one. Table 1, which additionally includes the 2D-only benchmark DAVIS (Doersch et al.,

Table 1: **End-to-end 3D trackers are not good 2D trackers.** 2D point tracking accuracy (δ_{avg}, ↑) of end-to-end 3D trackers, whose 3D tracks are projected back into the image, compared with the dedicated 2D tracker CoWTracker. CoWTracker is more accurate on every subset, by 18.1 points on average over the strongest end-to-end model. Top-2 results are highlighted as best and second.

		Real-world			*SynthVerse*							
	Model	adt	pst	davis	anim	emb	film	hum	int	nav	obj	Avg
(i)	TraceAnything (Liu et al., 2026)	34.0	30.8	19.9	30.2	26.4	36.5	21.8	42.6	17.1	25.4	28.5
(ii)	Any4D (Karhade et al., 2025)	49.0	35.3	26.3	46.0	47.5	50.2	42.4	72.0	41.4	38.8	44.9
(iii)	4RC (Luo et al., 2026)	70.8	58.6	36.6	59.3	61.6	66.0	56.1	87.8	69.5	52.7	61.9
(iv)	CoWTracker (Lai et al., 2026)	87.7	73.9	78.0	77.2	79.3	86.0	79.3	93.8	73.3	71.5	80.0

Table 2: **Diagnostic analysis of visible-point vs. occluded-point 3D tracking accuracy** (APD_{3D}, ↑). *Top:* On visible points, the Simple Baseline obtained by composing a 3D reconstructor (DA3 (Lin et al., 2025) or VGGT-Ω (Wang et al., 2026)) with CoWTracker (Lai et al., 2026) matches or surpasses end-to-end models on most subsets. *Bottom:* On occluded points, end-to-end models substantially outperform both Simple Baselines, confirming that occlusion handling is their core advantage. Our final model (shaded rows vi and xii) recovers this occluded-point accuracy and surpasses the end-to-end models, while also improving on visible points; parenthesized numbers give its improvement over its corresponding Simple Baseline.

			Real-world		*SynthVerse (Zhao et al., 2026)*							
	Model	Subset	adt	pst	anim	emb	film	hum	int	nav	obj	Avg
(i)	TraceAnything (Liu et al., 2026)	vis	83.7	76.3	11.5	70.9	43.7	13.4	20.3	38.2	58.7	46.3
(ii)	Any4D (Karhade et al., 2025)	vis	77.6	81.7	18.1	73.7	40.1	27.2	22.8	50.8	61.5	50.4
(iii)	4RC (Luo et al., 2026)	vis	87.9	89.3	23.2	75.3	49.4	33.6	30.7	66.0	67.4	58.1
(iv)	Simple Bas. (DA3+CoWTracker)	vis	93.9	89.1	23.9	75.8	52.7	33.7	30.3	64.3	68.4	59.1
(v)	Simple Bas. (VGGT-Ω+CoWTracker)	vis	95.4	92.2	23.1	75.3	58.8	35.2	27.2	66.3	70.2	60.4
(vi)	Ours (VGGT-Ω)	vis	95.2	94.2	26.7	76.8	60.5	38.1	27.3	70.6	71.2	62.3 (+1.9)
(vii)	TraceAnything (Liu et al., 2026)	occ	81.2	73.9	11.0	71.2	41.5	4.9	10.4	42.2	40.8	41.9
(viii)	Any4D (Karhade et al., 2025)	occ	78.6	77.0	14.0	74.6	41.0	9.3	11.3	49.4	41.2	44.0
(ix)	4RC (Luo et al., 2026)	occ	87.8	83.3	17.9	77.4	47.0	13.8	19.6	61.0	51.9	51.1
(x)	Simple Bas. (DA3+CoWTracker)	occ	65.0	76.6	13.4	67.7	40.4	9.3	8.9	32.8	53.9	40.9
(xi)	Simple Bas. (VGGT-Ω+CoWTracker)	occ	65.2	78.5	14.3	68.1	43.0	9.4	7.4	33.7	54.7	41.6
(xii)	Ours (VGGT-Ω)	occ	90.7	91.0	19.6	78.2	52.5	15.5	19.0	57.7	59.8	53.8 (+12.2)

2022), shows the opposite: CoWTracker (iv) is more accurate than all end-to-end models (i-iii) on every dataset, by 18.1 points on average over the strongest one, 4RC, and by more than a factor of two on DAVIS. This suggests that end-to-end 3D trackers, which are optimized primarily for geometric consistency, are not as well engineered as dedicated 2D trackers for establishing fine-grained pixel-level correspondences. Their tracking component thus leaves room for improvement, and modular designs, which can directly build on the best available 2D trackers, are a strong potential alternative.

The Simple Baseline. We now turn to 3D tracking with the full modular design. We use a state-of-the-art 2D point tracker like CoTracker (Karaev et al., 2025) or CoWTracker (Lai et al., 2026) to obtain 2D tracks. Then, we use a reconstruction foundation model like Depth Anything (Yang et al., 2024b; Lin et al., 2025) or VGGT-Ω (Wang et al., 2026) to estimate depth and cameras. Finally, we combine the 2D tracks, depth, and cameras, lift the 2D tracks to 3D, and call this approach the *Simple Baseline*.

Formally, let $I_1, \dots, I_T$ be the input video frames. The 2D tracker returns 2D trajectories $p_i(t) \in \mathbb{R}^2$, $t = 1, \dots, T$, for a number of query points $q_i \in \mathbb{R}^2$ defined by the benchmark. This means that the tracks go through the query points, such that $p_i(1) = q_i$. For simplicity, we assume that the query points are defined in the first frame. The 3D reconstructor returns a depth map D_t and camera parameters $\pi_t = (K_t, R_t, \boldsymbol{t}_t)$ for each video frame. We *lift* the 2D points to 3D by unprojecting them with their sampled depth, $\boldsymbol{x}_i(t) = \pi_t^{-1}\big(p_i(t) \mid D_t(p_i(t))\big)$, as detailed in section 4.2.[1] This baseline is deliberately minimal, as it applies the off-the-shelf components with no fine-tuning or adaptation. The output of the Simple Baseline is also the input that our model refines (section 4).

While the tracked point is by definition *visible* in the query frame, it may be *occluded* in other frames of the video. Next, we show that the performance of the Simple Baseline is very different depending on these two cases.

[1]When $p_i(t)$ is out of frame, we extend the depth map D_t as needed by replicating its border.

Observation I: The Simple Baseline Outperforms End-to-End Models on Visible Points. Table 2 (rows i-v) reports tracking accuracy on held-out subsets of each dataset for the visible portions of the tracks. The Simple Baseline (iv-v) is generally better than the end-to-end models. The advantage is quite large on real data. For instance, on ADT, the Simple Baseline achieves 93.9 accuracy, whereas the state-of-the-art end-to-end model 4RC (iii) achieves 87.9—a gap of 6 points. The Simple Baseline also leads on 6 out of 9 datasets and on aggregate (59.1 vs. 58.1).

Importantly, the advantage is *not* explained by a better backbone. 4RC is built on top of DA3's pre-trained backbone, fine-tuned with additional 3D tracking supervision, and yet the Simple Baseline, built on the same backbone and with no additional training, outperforms 4RC. One explanation is that learning both tracking and 3D reconstruction exceeds the capacity of a single backbone. Another, consistent with Observation 0, is that the architectures of these backbones are not as optimized as those of 2D trackers for establishing 2D correspondences.

Observation II: The End-to-end Models Outperform the Simple Baseline on Occluded Points. The picture changes markedly when the track is occluded, as the point moves behind another object or leaves the field of view. Since the Simple Baseline (rows x-xi) has no mechanism for reasoning about occlusions, it degrades substantially. End-to-end models (vii-ix), by contrast, are trained to hallucinate plausible 3D locations behind occlusion and perform much better. This appears to be their main advantage.

Discussion. By decomposing 3D tracking into 2D tracking and 3D reconstruction, the Simple Baseline already achieves strong 3D tracking performance for visible points. Hence, instead of training a 3D reconstructor and tracker end-to-end, a simpler approach is to (1) first *extract and lift visible track fragments* using off-the-shelf 2D tracking and reconstruction models, obtaining high-quality 3D tracks wherever the point is visible; (2) then *complete the tracks behind occlusion* by training a dedicated module that fills the gaps using cues from the surrounding context.

This reformulation has several advantages compared to end-to-end models. First, it *preserves the tracking accuracy of visible points*: the tracking and 3D reconstruction components operate at their individual best, with no accuracy sacrifice from joint optimization. Second, it *concentrates model capacity* on the difficult task of tracking behind occlusions instead of solving problems that the base models already handle well. Third, it is *future-proof*: as 2D trackers and 3D reconstructors improve, this approach automatically improves too.

We stress that this paradigm is *complementary* to end-to-end reconstruction and tracking, not a refutation of it. Our analysis motivates exploring this alternative design point; it does not claim that joint optimization is inherently flawed, nor that future end-to-end models cannot improve visible-point tracking too. In the next section, we instantiate this idea concretely.

4 METHOD

We now present TrackDiT, an instantiation of the paradigm of section 3. We begin by defining formally the 3D point tracking problem. Let $\{I_t\}_{t=1}^{T}$ be a video of T frames with $I_t \in \mathbb{R}^{3\times H\times W}$. Given a set of N query points $\{q_i\}_{i=1}^{N}$ specified in a reference frame, the goal is to estimate the 3D position $\boldsymbol{x}_t(q_i) \in \mathbb{R}^3$ of each query q_i in each frame t. The 3D points are defined in the reference frame of one of the cameras (which we call the world reference frame).

4.1 OVERVIEW

TrackDiT maps a video and its query points to 3D tracks in two stages (fig. 3).

Stage 1: Visible track extraction. First, we combine the outputs of an off-the-shelf 2D point tracker and a 3D reconstruction model. The tracker returns pixel trajectories $\hat{\mathbf{u}}_{i,t} \in \mathbb{R}^2$ and visibility $\hat{v}_{i,t}$, while the 3D reconstructor returns a depth map $\hat{D}_t$ and camera parameters π_t for every frame of the video. Sampling the ray that the tracked pixel defines at the estimated depth results in the *lifted* 3D trajectory $\hat{\boldsymbol{x}}_{i,t}$, formally defined in section 4.2. The resulting 3D trajectory is usually reliable, except that: (i) if the point is occluded, the depth map reports the depth of the *occluder* instead of the tracked point; (ii) if the point is out of frame, no value of depth is available at all; and, (iii) even if the point is visible, reconstructing it independently in each frame leaves temporal jitter.

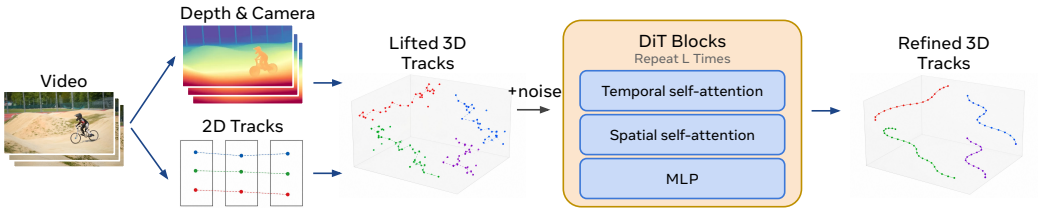


Figure 3: **Overview of our framework.** An off-the-shelf 2D tracker and a 3D foundation model turn the input video into 2D tracks and per-frame depth and cameras, which are combined into lifted 3D tracks. Treating each track-frame pair as a token, a factorized spatio-temporal DiT denoises these tracks with a conditional rectified flow, alternating spatial attention across tracks and temporal attention across frames, and outputs the refined 3D tracks.

Stage 2: Track completion via flow matching. Stage 2 is designed to correct precisely these issues. It predicts a *residual* to correct any error in the lifted tracks (section 4.2), focusing model capacity on fixing the gaps. Every track-frame pair becomes a token, carrying a compact description of the geometry that produced its lifted 3D position and of how far that geometry can be trusted (section 4.3). Because gaps in a track are genuinely ambiguous—there is no direct evidence of where the occluded point is—we *generate* the residual probabilistically with a conditional rectified flow model rather than using a regressor that would collapse to the conditional mean (section 4.4). The flow model (section 4.5) alternates attention along the two axes from which evidence about a gap can arrive: along time, it follows the track's own motion before and after the occlusion; across tracks, it follows the motion of co-visible points that may land on the same physical object. At test time, we sample a residual and add it to the lifted trajectory, obtaining the final trajectory (section 4.6).

Notably, the tokens represent 3D tracks rather than pixels, so the denoiser uses no image features: appearance is already consumed by the 2D tracker and the 3D reconstructor, and what remains is a geometric problem. This keeps the denoiser small, fast, and agnostic to the foundation models that produced its inputs, which run only once per video.

4.2 Lifting and Canonicalization

We now give more details on how tracks are lifted to 3D, and fix the coordinate frame and the prediction target that the rest of the section operates in.

Trajectory Lifting. Let $\pi_t = (K_t, R_t, \boldsymbol{t}_t)$ be the estimated camera parameters for frame t, taking a world point $\boldsymbol{x}$ to camera coordinates as $R_t\boldsymbol{x} + \boldsymbol{t}_t$, and let $\hat{D}_t$ be the estimated depth map, measured as distance along the viewing ray. The tracker's pixel $\hat{\mathbf{u}}_{i,t}$ for track i at frame t determines a ray with origin at the camera center and unit direction

$$\boldsymbol{o}_t = -R_t^\top \boldsymbol{t}_t, \qquad \boldsymbol{d}_{i,t} = R_t^\top \frac{K_t^{-1}\bar{\mathbf{u}}_{i,t}}{\|K_t^{-1}\bar{\mathbf{u}}_{i,t}\|}, \qquad \bar{\mathbf{u}}_{i,t} = [\hat{\mathbf{u}}_{i,t};\, 1], \tag{1}$$

and the depth map fixes how far along that ray the observed surface lies:

$$\hat{\boldsymbol{x}}_{i,t} = \boldsymbol{o}_t + \rho_{i,t}\,\boldsymbol{d}_{i,t}, \qquad \rho_{i,t} = \hat{D}_t\big(\hat{\mathbf{u}}_{i,t}\big). \tag{2}$$

Writing the lifted point this way makes the division of labour explicit, and with it the failure modes: the 2D tracker and camera predictions fix the ray $(\boldsymbol{o}_t, \boldsymbol{d}_{i,t})$, and the depth predictions contribute only the scalar $\rho_{i,t}$ along it. The ray remains meaningful under occlusion—assuming that the 2D tracker is, in fact, robust to occlusions—while $\rho_{i,t}$ does not, since the depth map then reports the occluder's surface instead.

Canonicalization. Since the 3D trajectories are fed to the correction model, we normalize them based on the per-clip coordinate-wise median center $\boldsymbol{m}$ and median radial scale s:

$$\tilde{\boldsymbol{x}}_{i,t} = \frac{\hat{\boldsymbol{x}}_{i,t} - \boldsymbol{m}}{s}, \qquad \boldsymbol{m} = \mathrm{med}_{i,t}\,\hat{\boldsymbol{x}}_{i,t}, \qquad s = \mathrm{med}_{i,t}\|\hat{\boldsymbol{x}}_{i,t} - \boldsymbol{m}\|. \tag{3}$$

Both statistics are computed from the lifted trajectory alone, never from ground truth, so the normalization can be applied consistently at test time too.

Residual parameterization. The model predicts a correction term $\boldsymbol{r}_{i,t}$ relative to the lifted trajectory:

$$\boldsymbol{r}_{i,t} = \tilde{\boldsymbol{x}}^{\mathrm{gt}}_{i,t} - \tilde{\boldsymbol{x}}_{i,t}, \qquad \boldsymbol{x}^{\mathrm{out}}_{i,t} = \hat{\boldsymbol{x}}_{i,t} + s\,\hat{\boldsymbol{r}}_{i,t}, \tag{4}$$

where $\tilde{\boldsymbol{x}}^{\mathrm{gt}}_{i,t}$ is the ground-truth position in the same canonical frame. Only the scale appears when mapping back: because $\boldsymbol{r}_{i,t}$ is a difference of two points in one canonical frame, the center $\boldsymbol{m}$ cancels and the prediction is returned to world coordinates by a single multiplication. This helps the model take advantage of the observation made in section 3: when the track is visible, it is generally already accurate and the model can predict a null residual, focusing its capacity on the difficult, occluded parts. The world frame is the natural space for this target, since a static point keeps a constant position there regardless of camera motion and a moving point traces a smooth curve—both properties that make a gap easy to interpolate, and neither of which holds for depth.

4.3 Geometric Conditioning

To predict the residual $\boldsymbol{r}_{i,t}$, the model has to know what produced the lifted 3D position of the track there and how far it can be trusted. For each $(\text{track}, \text{frame})$ pair this information is captured by one token vector, that combines the quantities of section 4.2:

$$\boldsymbol{g}_{i,t} = [\,\tilde{\boldsymbol{x}}_{i,t};\ \boldsymbol{d}_{i,t};\ \tilde{\boldsymbol{o}}_t;\ \tilde{\mathbf{u}}_{i,t};\ \hat{v}_{i,t};\ b_{i,t}\,], \tag{5}$$

where $\boldsymbol{d}_{i,t}$ is the world-frame unit ray direction through the tracked pixel, $\tilde{\boldsymbol{o}}_t$ the canonicalized camera center, $\tilde{\mathbf{u}}_{i,t}$ the image-relative 2D track coordinate, $\hat{v}_{i,t}$ the tracker's predicted visibility, $b_{i,t}$ an out-of-frame indicator, and $[\cdot\,;\cdot]$ denotes concatenation.

The lifted position $\tilde{\boldsymbol{x}}_{i,t}$ is the primary cue; the ray direction and camera center complement it with the viewing geometry that produced it, and the visibility cues indicate where it can be trusted (section C quantifies each contribution). The sampled distance $\rho_{i,t}$ is deliberately left out, as it is recoverable from $\tilde{\boldsymbol{x}}_{i,t}$, $\boldsymbol{d}_{i,t}$, and $\tilde{\boldsymbol{o}}_t$, and we found it redundant in practice. The 2D coordinate is, also deliberately, not clipped at the image boundary: in-frame pixels map into $[-1, 1]$ while out-of-frame extrapolations keep their displacement outside that range, so the token also tells how far outside the frame the tracker believes the point to be.

The visibility estimate is concatenated as an input feature rather than used as a multiplicative gate: this lets the network learn to modulate its reliance on the lifted geometry based on its reliability, rather than hard-zeroing potentially informative (if noisy) estimates at occluded frames.

4.4 Flow Matching on Track Residuals

Having defined both the residual $\boldsymbol{r}$ and the conditioning $\boldsymbol{g}$, we learn the conditional distribution $p(\boldsymbol{r} \mid \boldsymbol{g})$ with rectified flow. Briefly, we define the linear interpolant

$$\boldsymbol{r}^{\tau} = (1-\tau)\,\boldsymbol{\epsilon} + \tau\,\boldsymbol{r}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{6}$$

where $\tau \in (0, 1)$ is the (diffusion) time (τ is sampled from a logit-normal distribution during training); the corresponding target velocity is $\boldsymbol{v} = \boldsymbol{r} - \boldsymbol{\epsilon}$. The network f_θ predicts the *clean* residual $\boldsymbol{r}$, which we then convert to a velocity before comparing to the target one $\boldsymbol{v}$:

$$\hat{\boldsymbol{v}} = \frac{f_\theta(\boldsymbol{r}^{\tau}, \tau, \boldsymbol{g}) - \boldsymbol{r}^{\tau}}{\max(1-\tau,\ \epsilon_0)}, \qquad \epsilon_0 = 0.05. \tag{7}$$

Predicting the clean sample keeps the network's output in the same units and scale as its conditioning, which is precisely why the residual parameterization of section 4.2 is useful; supervising the velocity keeps the objective aligned with the sampler's integration step, so the two parameterizations remain directly comparable. The loss is a token-weighted velocity MSE in which occluded and out-of-frame tokens count three times as much as visible ones, concentrating the gradient on the completion sub-problem; section B gives the exact form and the outlier handling.

4.5 Spatio-Temporal Denoiser

It remains to specify f_θ, which we realize as a factorized spatio-temporal DiT over the $N \times T$ grid of track-frame tokens. At each token, the noisy residual $\boldsymbol{r}^{\tau}_{i,t}$ and its conditioning vector $\boldsymbol{g}_{i,t}$ are separately projected to channel dimension D and summed, a sinusoidal frame encoding is added, and L blocks of three operations follow.

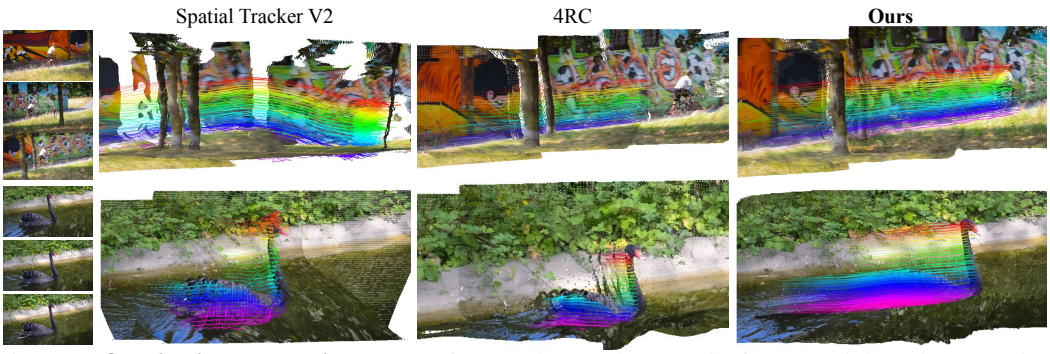


Figure 4: **Qualitative comparison.** Input frames (*left*) and 3D tracks from SpatialTrackerV2 (Xiao et al., 2025), 4RC (Luo et al., 2026), and TrackDiT for a cyclist riding behind tree trunks (*top*) and a swimming swan (*bottom*). On the cyclist, the tracks of both baselines drift until they detach from the object, and on the swan both produce short, jittery trajectories; TrackDiT's tracks stay smooth and attached through the occlusions.

Temporal self-attention. Tokens along the same track attend to each other, $\mathrm{SA}_{\mathrm{temp}}(H_{i,1:T})$. This propagates information about the trajectory observed before and after occlusions into the occluded gaps. It also helps to smooth out the estimated motion, removing temporal jitter that the per-frame depth sampling introduces into the lifted 3D points.

Spatial self-attention. Tokens that belong to the same video frame attend to each other too, $\mathrm{SA}_{\mathrm{spat}}(H_{1:N,t})$. Temporal reasoning alone can only extrapolate a point's past motion, which fails as soon as the object accelerates or turns behind the occluder. Co-visible points on the same object, in contrast, observe that motion directly, and attending across tracks lets the model transfer it to the occluded points.

MLP. A pointwise feed-forward block completes the layer. All three sublayers are pre-normalized and modulated by the flow time via AdaLN-Zero: a small network maps a sinusoidal embedding of τ to per-sublayer shift, scale, and gate parameters, with the gates initialized to zero so that every block begins as the identity. The factorization is what keeps this affordable. A joint attention over all track-frame tokens would be quadratic in $N \cdot T$, whereas the two factored attentions are quadratic in T and in N separately, letting us train with hundreds of tracks over tens of frames in one window.

4.6 Inference

At test time, we integrate the flow ODE $\mathrm{d}\boldsymbol{r}/\mathrm{d}\tau = \hat{\boldsymbol{v}}(\boldsymbol{r}, \tau, \boldsymbol{g})$ from $\boldsymbol{r}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at $\tau = 0$ to $\tau = 1$ with 20 Euler steps, then rescale the residual by s and add it to the lifted trajectory (section 4.2).

Longer videos. The frame encoding is the only component tied to sequence length; for clips longer than the training window, we resample it onto the training-length grid at fractional positions, so that the model never sees encodings beyond those it was trained on.

5 Experiments

5.1 3D Point Tracking

We evaluate on ADT (Pan et al., 2023), PStudio (Joo et al., 2019), and the SynthVerse (Zhao et al., 2026) Benchmark (across 7 subsets) under the 4RC / St4RTrack protocol (Luo et al., 2026), in which predictions are aligned to ground truth via Umeyama alignment (scale, rotation, translation). **$\mathbf{APD_{3D}}$** averages the fraction of points within $\{0.1, 0.3, 0.5, 1.0\}$ m thresholds (↑). Details are in section A.

Quantitative Results. Table 3 reports 3D tracking results on the St4RTrack / 4RC metrics, which evaluate aligned world-space trajectories. Our method achieves the highest average APD_{3D} with either 3D foundation model (58.0 with VGGT-Ω, 57.0 with DA3), outperforming all baselines. On the real-world subsets, it also outperforms the strongest baseline on each, SpatialTrackerV2 on ADT (94.0 vs. 93.6) and V-DPM on PStudio (91.2 vs. 89.6). Our two variants also inherit the ordering of their inputs from table 2—the VGGT-Ω variant leads on both real-world subsets and on average—confirming that the refinement rides the quality of whichever reconstruction model it is

Table 3: **3D point tracking results** on the St4RTrack (Feng et al., 2025) metrics (*world-space*, APD_{3D} ↑). We report our model with both 3D foundation models used in section 3, DA3 (Lin et al., 2025) and VGGT-Ω (Wang et al., 2026). Top-2 results are highlighted as best and second .

		Real-world		*SynthVerse (Zhao et al., 2026)*							
	Model	adt	pst	anim	emb	film	hum	int	nav	obj	Avg
(i)	TraceAnything (Liu et al., 2026)	83.0	75.6	10.3	69.3	41.1	8.5	18.6	36.1	47.2	43.3
(ii)	St4RTrack (Feng et al., 2025)	77.4	76.7	9.9	72.4	40.1	12.6	15.1	46.5	42.6	43.7
(iii)	Any4D (Karhade et al., 2025)	76.7	80.2	16.4	73.3	40.1	16.8	20.9	48.1	51.0	47.1
(iv)	Open-D4RT (Zhang et al., 2026)	76.0	88.9	16.8	74.7	47.5	13.8	17.3	46.3	57.1	48.7
(v)	TAPIP3D (Zhang et al., 2025a)	90.4	75.3	20.9	70.0	42.5	13.4	28.1	53.9	55.2	50.0
(vi)	SpatialTrackerV2 (Xiao et al., 2025)	93.6	75.5	21.1	72.8	43.2	19.9	25.2	52.0	57.6	51.2
(vii)	V-DPM (Sucar et al., 2026)	80.8	89.6	23.8	73.7	43.5	22.9	27.4	60.3	63.0	53.9
(viii)	4RC (Luo et al., 2026)	87.6	87.2	22.3	74.2	46.1	22.5	28.6	63.3	58.5	54.5
(ix)	Ours (DA3)	93.0	90.7	24.8	76.3	52.1	23.2	27.8	60.7	64.0	57.0
(x)	Ours (VGGT-Ω)	94.0	91.2	23.6	75.6	58.0	24.7	26.5	65.1	63.6	58.0



Figure 5: **Raw lift vs. TrackDiT.** *Left:* as in fig. 1, tracks passing behind an occluder (green box) take the occluder's depth in the raw lift. *Right:* near object boundaries, pixel quantization and small 2D tracking errors make the sampled depth alternate between foreground and background (red boxes), so the raw tracks jump back and forth. TrackDiT corrects both failures.

given. 4RC retains an advantage only on the interaction subset (28.6 vs. 26.5), likely due to its access to large-scale synthetic 4D training data—an advantage orthogonal to our architectural contributions.

Qualitative Results. Figure 4 compares TrackDiT with two strong published trackers, SpatialTrackerV2 and 4RC, on a cyclist who repeatedly disappears behind tree trunks and a swan swimming on reflective water. On the cyclist, the tracks of both baselines gradually drift until they detach from the object; on the swan, both methods produce short and jittery trajectories. Our tracks stay smooth and attached throughout, carried through the trunks by the denoiser's track completion. Figure 5 contrasts the raw lift with our refined tracks: beyond completing tracks behind occluders, TrackDiT also removes the foreground–background flicker that arises when depth is sampled at object boundaries. Figure 2 shows further results on real and synthetic videos.

5.2 Revisiting Visible and Occluded Points

Rows vi and xii of table 2 show where the gains of section 5.1 come from: compared to the Simple Baseline it refines, our model improves mostly on occluded points, as hypothesized in section 3.

Occluded points: the gap is closed. Our model improves over the Simple Baseline by 12.2 points (53.8 vs. 41.6) and overtakes the strongest end-to-end model, 4RC, by 2.7 (53.8 vs. 51.1), winning on 7 of the 9 subsets. The 9.5-point deficit identified in section 3 is thus not only closed but reversed, by a lightweight model trained on track geometry alone.

Visible points: nothing is lost. Thanks to the residual parameterization of section 4.2, a point that the lift already places correctly needs a near-zero correction. Accordingly, visible-point accuracy does not degrade but improves, from 60.4 to 62.3, on 8 of the 9 subsets (ADT is flat, 95.2 vs. 95.4).

6 Conclusion

We showed that popular end-to-end 3D reconstruction and tracking models are, in fact, not very good 2D trackers. Conversely, we showed that trivially combining state-of-the-art 2D trackers and 3D feed-forward reconstructors leads to excellent 3D tracking results away from occlusions. We also showed that one can train an efficient model to correct tracks behind occlusions. This yields a simple, modular 3D tracking framework that outperforms end-to-end 3D trackers and will automatically benefit from future improvements in the base models.

AI USE STATEMENT

In this work, we used generative AI tools to help implement methods (i.e., write and debug code), assist in launching experiments and writing code to collect, analyze, and visualize the resulting data, and proofread the manuscript. However, we did not use AI to develop concepts, design experiments, propose or refine hypotheses, prepare datasets, or interpret the results. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI. Generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in writing proofs or with translation, and supporting qualitative and thematic data analysis were not applicable to this manuscript.

REPRODUCIBILITY STATEMENT

We describe the model in section 4 and provide full implementation details, including architecture, training data, optimization, and inference settings, in section A, the training objective in section B, and ablations of the main design choices in section C. All Stage-1 components are publicly available pre-trained models, and all evaluation datasets are public benchmarks. We will release our code upon publication.

REFERENCES

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. ECCV*, 2020.

Seokju Cho, Jiahui Huang, Jisu Nam, Honggyu An, Seungryong Kim, and Joon-Young Lee. Local all-pair correspondence for point tracking. *arXiv*, 2407.15420, 2024.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. In *Proc. NeurIPS*, 2022.

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 *Proc. CVPR*, 2023.

Carl Doersch, Yi Yang, Dilara Gokay, Pauline Luc, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped training for tracking-any-point. *arXiv*, 2402.00847, 2024.

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 *Proc. ICCV*, 2025.

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 *Proc. CVPR*, 2022.

Jisang Han, Honggyu An, Jaewoo Jung, Takuya Narihira, Junyoung Seo, Kazumi Fukuda, Chaehyun Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. D^2USt3R: Enhancing 3D reconstruction with 4d pointmaps for dynamic scenes. *arXiv*, 2504.06264, 2025.

Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle videos revisited: Tracking through occlusions using point trajectories. In *Proc. ECCV*, 2022.

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. *arXiv*, 2506.07310, 2025.

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

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Nikita Karaev, Ignacio Rocco, Ben Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

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 *Proceedings of the International Conference on Computer Vision (ICCV)*, 2025.

Jay Karhade, Nikhil Varma Keetha, Tanisha Gupta, Yuchen Zhang, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4D: Towards unified feed-forward metric 4d reconstruction. In *Proc. ICCV Workshop*, 2025.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*. IEEE, 2026.

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. *arXiv*, 2407.05921, 2024.

Zihang Lai, Eldar Insafutdinov, Edgar Sucar, and Andrea Vedaldi. CoWTracker: Tracking by warping instead of correlation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, and Lei Zhang. TAPTR: Tracking any point with transformers as detection. In *Proc. ECCV*, 2024.

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

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 *Proc. ICLR*, 2026.

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

Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. In *Proc. CVPR*, 2024.

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

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng (Carl) Ren. Aria digital twin: A new benchmark dataset for egocentric 3D machine perception. In *Proc. ICCV*, 2023.

Peter Sand and Seth J. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80 (1), 2008.

Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-DPM: Video reconstruction with dynamic point maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, 2020.

Bo Wang, Jian Li, Yang Yu, Li Liu, Zhenping Sun, and Dewen Hu. SceneTracker: long-term scene flow estimation network. *arXiv*, 2403.19924, 2024a.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

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

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

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

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. *arXiv preprint arXiv:2507.13347*, 2025c.

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

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proc. CVPR*, 2024a.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. *arXiv*, 2406.09414, 2024b.

Bowei Zhang, Lei Ke, Adam W. Harley, and Katerina Fragkiadaki. TAPIP3D: tracking any point in persistent 3D geometry. *arXiv*, 2504.14717, 2025a.

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 *Proc. CVPR*, 2026.

Songyan Zhang, Yongtao Ge, Jinyuan Tian, Guangkai Xu, Hao Chen, Chen Lv, and Chunhua Shen. POMATO: marrying pointmap matching with temporal motion for dynamic 3D reconstruction. *arXiv*, 2504.05692, 2025b.

Weiguang Zhao, Haoran Xu, Xingyu Miao, Qin Zhao, Rui Zhang, Kaizhu Huang, Ning Gao, Peizhou Cao, Mingze Sun, Mulin Yu, Tao Lu, Linning Xu, Junting Dong, and Jiangmiao Pang. SynthVerse: a large-scale diverse synthetic dataset for point tracking. In *Proc. SIGGRAPH*, 2026.

Yang Zheng, Adam Harley, Bokui Shen, Gordon Wetzstein, and Leonidas Guibas. PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *Proc. CVPR*, 2023.

Artem Zholus, Carl Doersch, Yi Yang, Skanda Koppula, Viorica Patraucean, Xu Owen He, Ignacio Rocco, Mehdi S. M. Sajjadi, Sarath Chandar, and Ross Goroshin. TAPNext: tracking any point (TAP) as next token prediction. *arXiv*, 2504.05579, 2025.

A Implementation Details

Stage 1 uses either Depth Anything 3 (Lin et al., 2025) or VGGT-Ω (Wang et al., 2026) for depth and cameras, and CoWTracker (Lai et al., 2026) as the 2D tracker; all components are frozen and never fine-tuned. The denoiser has $L{=}12$ blocks with hidden dimension $D{=}512$, 8 attention heads, and MLP expansion ratio 4, for roughly 80M parameters—an order of magnitude smaller than the foundation models it corrects. Its input is the 13-dimensional geometric conditioning of section 4.3, and occluded and out-of-frame entries of the lift are reset to the per-track visible-anchor median in world coordinates before conditioning. We train on a mixture of SynthVerse (Zhao et al., 2026), Dynamic Replica (Karaev et al., 2023), Kubric (Greff et al., 2022), and Point Odyssey (Zheng et al., 2023) for 80k steps with AdamW (lr 4×10^{-4}, constant schedule) in mixed precision, keeping an EMA of the weights with decay 0.9993. Each sample contains 128 tracks over a 48-frame window, with a batch size of 8 samples per GPU across 8 GPUs. The occlusion cue in the conditioning is the 2D tracker's *predicted* visibility at both training and test time. Loss functions and training objectives are detailed in section B. At inference we integrate the flow with 20 Euler steps and a single sample per track, and resample the frame encoding onto the 48-frame training grid for longer clips.

B Loss Functions

Velocity matching. The training objective is a token-weighted velocity MSE over the $N \times T$ grid of track-frame tokens:

$$\mathcal{L}_{\text{flow}} = \frac{\sum_{i,t} w_{i,t} \|\hat{\boldsymbol{v}}_{i,t} - \boldsymbol{v}_{i,t}\|^2}{\sum_{i,t} w_{i,t}}, \tag{8}$$

where $\boldsymbol{v} = \boldsymbol{r} - \boldsymbol{\epsilon}$ is the target velocity of the interpolant and $\hat{\boldsymbol{v}}$ is derived from the network's clean-residual prediction as in section 4.4. Since the loss is always computed on velocities, the clean-sample and velocity parameterizations of the network are optimized on equal terms.

Region weighting. The weight $w_{i,t}$ is set by the region a token belongs to: 1 for visible tokens, 3 for occluded ones, and 3 for out-of-frame ones. The three regions are mutually exclusive, so the weighting simply states that the two unobserved regimes—where the model has to do the actual completion—are worth three visible tokens each. Only tokens whose supervision is non-finite are removed outright. In particular, we keep the far out-of-frame points as these are precisely the hardest completion cases, and excluding them would remove the supervision the out-of-frame weight is meant to emphasize.

Outlier handling. A small fraction of tokens carry enormous normalized residuals, typically where the 3D reconstructor fails catastrophically or the tracker leaves the scene. Left in the loss they dominate the gradient, so we drop tokens whose target residual norm exceeds a cap of 8; removing the cap degrades validation accuracy substantially.

C Ablation Studies

We ablate three design choices of TrackDiT: the geometric conditioning (section 4.3), the factorization of attention (section 4.5), and the space in which the residual is predicted (section 4.2). All variants use VGGT-Ω and differ from the full model only in the ablated component. Table 4 reports APD_{3D} averaged over the nine subsets of table 3, separately for all, visible, and occluded points.

Geometric conditioning. The full model conditions each token on the 13-dimensional vector $\boldsymbol{g}_{i,t}$ of section 4.3. We remove three groups of inputs in turn: the camera geometry, i.e., the ray direction $\boldsymbol{d}_{i,t}$ and the camera center $\tilde{\boldsymbol{o}}_t$, leaving 7 dimensions (ii); the lifted position $\tilde{\boldsymbol{x}}_{i,t}$, leaving 10 (iii); and the visibility cues $\hat{v}_{i,t}$ and $b_{i,t}$, leaving 11 (iv). Removing the camera geometry does not remove all camera information, since the lifted position is itself computed with the estimated cameras. The lifted position is by far the most important input: without it, accuracy drops by 3.2 points overall and by 5.4 on occluded points. Removing the visibility cues costs 1.3 points, and removing the camera geometry only 0.3, presumably because most of its information is already carried by the lift.

Attention factorization. The full model alternates temporal attention within each track and spatial attention within each frame (section 4.5). To disable one of the two axes, we set its attention length

Table 4: **Ablation studies.** APD_{3D} (↑) averaged over the nine subsets of table 3, for all, visible, and occluded points. Each variant changes a single component of the full model (VGGT-Ω); see section C for details.

	Variant	all	vis	occ
(i)	Ours (VGGT-Ω)	58.0	62.3	53.8
Geometric conditioning				
(ii)	w/o camera geometry ($\boldsymbol{d}$, $\tilde{\boldsymbol{o}}$)	57.7	61.7	53.1
(iii)	w/o lifted position ($\tilde{\boldsymbol{x}}$)	54.8	59.3	48.4
(iv)	w/o visibility cues ($\hat{v}$, b)	56.7	60.8	52.6
Attention factorization				
(v)	Temporal only	53.7	58.1	46.7
(vi)	Spatial only	56.4	61.0	51.3
State space				
(vii)	Camera-frame residual	56.9	61.2	52.4
(viii)	Ray-frame residual	57.1	61.3	52.1

to one: the parameters and layer structure are unchanged, but tokens along that axis can no longer communicate. Disabling spatial attention (v) is the most harmful change in the study, lowering accuracy by 4.3 points overall and by 7.1 on occluded points. Disabling temporal attention (vi) costs 1.6 points overall and 2.5 on occluded points. Both axes thus contribute, and the ability to exchange information across tracks matters most, especially behind occlusions.

State space. The full model predicts the residual $\boldsymbol{r}_{i,t}$ in the canonical world frame (section 4.2). We compare it with two other bases for the same residual: the camera frame (vii), obtained by rotating the residual into the coordinates of each frame's camera and rotating the prediction back; and a per-point ray frame (viii), whose axes are the viewing ray $\boldsymbol{d}_{i,t}$ and two directions orthogonal to it. Both are slightly worse than the world frame, by 1.1 and 0.9 points overall, in line with the argument of section 4.2 that static points and smooth motion are simplest to describe in world coordinates. A third alternative, which predicts image-plane offsets together with a change in ray depth, $(\Delta u, \Delta v, \Delta\rho)$, did not converge and is omitted from the table.

D Additional Results

Figure 6 shows further 3D tracks predicted by TrackDiT on real-world videos, complementing fig. 2. Each panel draws the world-space trajectories of the query points over the reconstructed scene. The videos span animals (the flamingo), articulated human motion (breakdancing, parkour, tennis), and vehicles crossing the frame at speed (motocross, a drifting car), many of them filmed with a moving camera. Across them, tracks on the same object move as a coherent bundle and follow smooth paths, including through the fast, non-rigid motion of the dancer and the tennis player.

E Limitations

Our method inherits the failure modes of its component models: if the 2D tracker loses a point entirely or the 3D reconstructor produces large systematic errors, the denoiser has limited ability to recover. Because it conditions only on geometry, it also cannot appeal to appearance to disambiguate a case its inputs have already resolved incorrectly. Finally, the denoiser is trained on synthetic data in which long occlusions with substantial motion are rare, which may limit how well it completes such cases in practice.

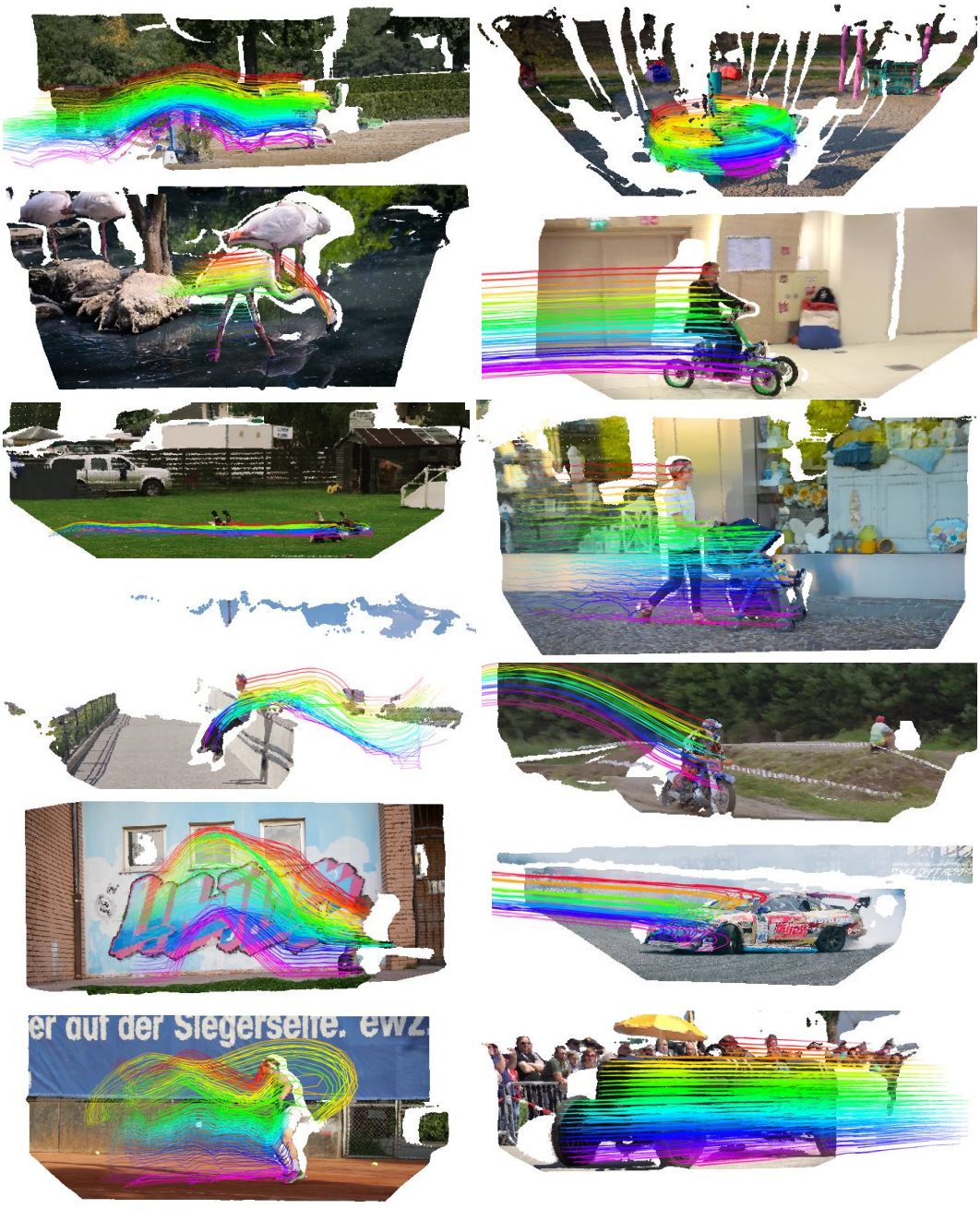


Figure 6: Additional qualitative 3D tracking results on diverse in-the-wild videos. Our method faithfully reconstructs both rigid scenes and non-rigid objects, producing temporally coherent 3D trajectories across viewpoint changes. Video results in the supplementary material better visualize the temporal coherence of the tracks.