# TRACKEVERYTHING: LONG HORIZON DENSE TRACKING VIA DE-DUPLICATING 3D SCENE REPRESENTATIONS

**Anonymous authors**
Paper under double-blind review

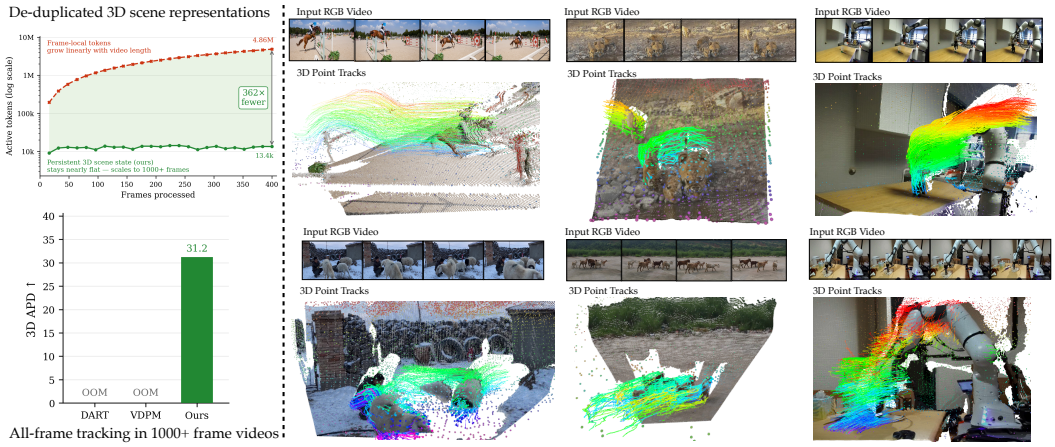


Figure 1: **TrackEverything** tracks *all* points across *all* frames of long videos (1000+ frames) in 3D world-coordinate space. **Left:** a persistent, voxel-de-duplicated 3D scene representation grows with unique scene content rather than video length, remaining practical while prior all-frame dense 3D trackers run out of memory. **Right:** for each input video (top) we show the output dense 3D tracks (bottom). Trails are shown for dynamic points; static content is rendered as a 3D point cloud.

## ABSTRACT

Existing point tracking models face a fundamental tradeoff: they can either track a sparse set of query points over long horizons, or track all points across only short clips. We introduce **TrackEverything**, a 3D point tracker that breaks this trade-off by representing videos as persistent 3D scene tracks in world coordinates. Grounded in the insight that videos are 2D projections of an underlying 3D world, TrackEverything decouples model complexity from video duration, allowing it to scale with unique physical scene geometry instead. Our approach introduces three key innovations. **First**, we employ a *voxelization-based de-duplication* mechanism at sliding-window boundaries to merge co-located tracks, preventing repeated observations of the same surface from redundantly accumulating. **Second**, we decompose tracking into an *endpoint refiner* that predicts each point's destination and static-versus-dynamic classification, followed by a lightweight *trajectory refiner* that decodes dense trajectories exclusively for dynamic points. **Third**, we propose *3D WAFT*, replacing memory-prohibitive 4D correlation volumes with efficient feature sampling in the scene cloud. To the best of our knowledge, TrackEverything is the first 3D tracker capable of tracking all visible points across videos exceeding 1000 frames within 40 GB of GPU memory. On TAPVid-3D, TrackEverything outperforms all open-source all-frame dense 3D trackers by more than 20% APD on short clips, while remaining competitive with state-of-the-art sparse trackers on long sequences, despite tracking far more points.

## 1 INTRODUCTION

Video is the primary medium through which modern AI models perceive the dynamic physical world, underpinning vision-language models (Team et al., 2025), vision-language-action policies (Intelligence et al., 2026), and point trackers (Harley et al., 2025) alike. Yet processing video in frame-by-frame 2D representations incurs a computational cost and memory footprint that scales linearly with video duration: each new frame introduces a full grid of visual tokens that must be



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.

processed, regardless of whether it reveals novel scene content or merely re-observes surfaces already seen. This per-frame redundancy severely limits scalability to long videos.

In point tracking, for example, this redundancy forces an unnatural divide. *Sparse trackers* (Zhang et al., 2026a; Xiao et al., 2025; Karaev et al., 2025) can track over extended time horizons, but only for a sparse set of user-specified query points, leaving the rest of the scene unmodeled. *Dense trackers*, in contrast, operate in frame-local pixel space where redundant re-encoding causes computation to explode with video length. Consequently, existing dense methods are forced into severe compromises: they either track only points visible in the first frame (Harley et al., 2025; Ngo et al., 2025; Feng et al., 2025; Karhade et al., 2026), ignoring newly disoccluded surfaces, or track all points but remain restricted to short clips of 48–64 frames (Zhang et al., 2026b; Sucar et al., 2026). As a result, no prior method can track all visible scene points across full-length videos.

These standard video processing regimes under-exploit the fact that videos are merely streams of 2D camera projections of an underlying 3D world. In a 3D-based scene representation, there is an opportunity to scale with unique scene content rather than video duration. Exploiting this gap between *frame-based scaling* and *3D scene-centric scaling* is the central motivation of our work.

We introduce **TrackEverything**, a 3D point tracker representing videos as *non-redundant* and *long-horizon* 3D scene tracks in world coordinates (Figure 1). Instead of redundantly tracking every 2D pixel across time, TrackEverything maintains a 3D representation that scales with unique physical geometry rather than frame count. Our architecture achieves this through three key innovations: **First**, we introduce a *voxelization-based de-duplication* mechanism at temporal window boundaries. When points from different frames converge onto the same physical surface, they occupy the same 3D coordinates and are merged into a single canonical track, preventing redundant copies from accumulating over time. **Second**, we decompose tracking into an *endpoint refiner* and a *trajectory refiner*. The endpoint refiner estimates each point's destination at the end of the window and classifies it as static or dynamic; the trajectory refiner then decodes dense within-window trajectories exclusively for dynamic points. Because static points remain stationary in world coordinates, this factorization concentrates dense decoding on the small moving subset of the scene. **Third**, we propose *3D WAFT*, an extension of warp-aligned feature transforms (Wang & Deng, 2026) to 3D point clouds. By sampling feature maps at projected source and target locations, 3D WAFT replaces memory-intensive 4D correlation volumes with efficient template matching at a fraction of the compute cost.

Concretely, TrackEverything processes a video with a sliding window strategy. Within each window, 2D visual features are unprojected into a world-coordinate feature cloud using camera poses and depth from sensors or off-the-shelf estimators (e.g., VGGT (Wang et al., 2026a)). An *endpoint refiner* transformer iteratively updates each point's estimated 3D position at the end of the window using 3D WAFT feature sampling, while simultaneously predicting its static-vs.-dynamic classification. For points classified as dynamic, a *trajectory refiner* decodes dense within-window trajectories conditioned on their motion and local scene context, while static points trivially retain their positions. Finally, at each window boundary, the 3D feature cloud is voxelized at the updated positions to merge co-located points, yielding a compact, de-duplicated set of tracks for the next window, to be tracked alongside newly revealed content.

We evaluate TrackEverything on standard 3D point tracking benchmarks including TAPVid-3D (Koppula et al., 2024). On short clips, where prior all-frame dense baselines remain computationally viable, TrackEverything outperforms all open-source all-frame dense 3D tracking methods by more than 20% APD. Crucially, while existing all-frame dense trackers exhaust GPU memory beyond roughly 96 frames, TrackEverything is the first 3D tracker that scales to tracking all points across videos exceeding 1000 frames—operating within 40 GB of GPU memory. Despite tracking orders of magnitude more points, TrackEverything remains competitive with state-of-the-art sparse trackers and first-frame dense methods on long sequences, while providing dense tracks for the entire scene.

Beyond point tracking, we believe our persistent, de-duplicating 3D dynamic scene representations provide a long-sought substrate for downstream foundation models. While explicit 3D representations have shown clear benefits for vision-language models (Lin et al., 2026; Jain et al., 2025), robotic manipulation policies (Ke et al., 2025), and video generation (Zhou et al., 2025), existing approaches remain largely confined to static or quasi-static scenes. By making all-point tracking computationally tractable over extended horizons, TrackEverything opens an exciting path toward extending these models to dynamic, real-world environments.

Our contributions are summarized as follows:

- We introduce TrackEverything, which to the best of our knowledge is the first 3D point tracker capable of tracking all visible points across long horizons (1000+ frames).
- We propose a voxelization-based de-duplication mechanism, making our representation complexity scale with unique physical scene geometry rather than video duration.
- We introduce two efficiency gains: (1) "endpoints before trajectories" as a problem decomposition, to focus dense decoding on dynamic points, and (2) 3D WAFT to replace expensive 4D correlations with efficient feature sampling, together making all-frame tracking practical.
- On TAPVid-3D, TrackEverything outperforms all open-source all-frame dense 3D trackers by more than 20% APD on short clips, and matches state-of-the-art sparse trackers on long sequences, and is the only method that scales all-point tracking to 1000+ frames.

We invite readers to view full-length dense 3D tracking visualizations on our project website: `https://iclr-8302.netlify.app/`.

## 2 Related Work

**Sparse Point Tracking.** PIPs (Harley et al., 2022) and TAP-Vid (Doersch et al., 2022) established the modern paradigm for sparse point tracking: tracking a small set of user defined query points across videos. Subsequent methods (Karaev et al., 2025) follow the same recipe: sliding-window inference for long videos, and recurrent iterative refinement guided by 4D correlation volumes between query features and dense video features (Teed & Deng, 2020). However, constructing and querying dense correlation volumes scales with the number of query points, thus limiting the number of points that can be tracked. Recent work extends sparse tracking into 3D: SpatialTracker (Xiao et al., 2024) represents points via tri-plane representations, TAPIP-3D (Zhang et al., 2026a) performs neighborhood attention within 3D feature clouds, and SpatialTracker-v2 (Xiao et al., 2025) jointly refines geometry and motion using both 2D and 3D correlations. While these methods use 3D representations, their internal feature clouds retain a full grid of points for every video frame, causing memory to scale linearly with video duration. TrackEverything adopts the recurrent sliding-window methodology, but diverges in three respects: (1) our 3D representation is not tied to pixel space, but grows only when new scene content appears, via voxelization-based de-duplication; (2) we replace expensive 4D correlations with a cheaper 3D extension of WAFT (Wang & Deng, 2026); and (3) we track *all* points across all frames of a long video, rather than being restrictedto a sparse query set.

**Dense Point Tracking.** Dense point tracking aims to track points sampled densely in a grid instead of user-specified sparse query points. In 2D, DELTA (Ngo et al., 2025), AllTracker (Harley et al., 2025), and CoWTracker (Lai et al., 2026) track dense point grids in long videos, but are strictly limited to points visible in the first frame. Omnimotion (Wang et al., 2023) can track all points across all frames, but requires 8+ hours of per-video test-time optimization. In 3D, ST4RTrack (Feng et al., 2025), DPM (Sucar et al., 2025), and Any4D (Karhade et al., 2026) use DUSt3R/VGGT-style backbones for dense tracking, but remain limited to tracking points visible in first frame and only on short clips. VDPM (Sucar et al., 2026), TraceAnything (Liu et al., 2026), and D4RT (Zhang et al., 2026b) aim to track all points across all frames, but remain confined to short videos as well due to the growing cost of pixel-space representations. Since the same surface is represented independently in every frame, tracking all points in a $512\times512$, 200-frame video requires on the order of $10^{10}$ predictions. TrackEverything represents videos within a persistent, world-coordinate 3D scene space. By continuously de-duplicating co-located points across sliding windows, our representation collapses temporal redundancy and expands only when genuinely new geometry is revealed, making all-frame dense tracking across 1000+ frame videos computationally viable.

## 3 Method

Given an RGB video $\mathbf{I} \in \mathbb{R}^{T\times H\times W\times 3}$ and per-frame 3D pointmaps $\mathbf{P} \in \mathbb{R}^{T\times H\times W\times 3}$ in world coordinates (obtained from sensors or feedforward reconstruction models such as VGGT-$\Omega$ (Wang et al., 2026a)), our goal is to track all visible scene surfaces over time. Specifically, TrackEverything predicts dense 3D trajectories $\mathbf{X} \in \mathbb{R}^{N\times T\times 3}$, visibility logits $\mathbf{V} \in \mathbb{R}^{N\times T}$, and static/dynamic logits $\mathbf{S} \in \mathbb{R}^{N}$, for all $N$ unique scene points (Figure 2), with support also for user-specified subpixel query points. Standard frame-by-frame 2D tracking incurs memory and compute costs that scale linearly with video duration $T$, as each new frame redundantly re-introduces previously observed surfaces. TrackEverything resolves this bottleneck by tracking directly on persistent 3D physical entities in

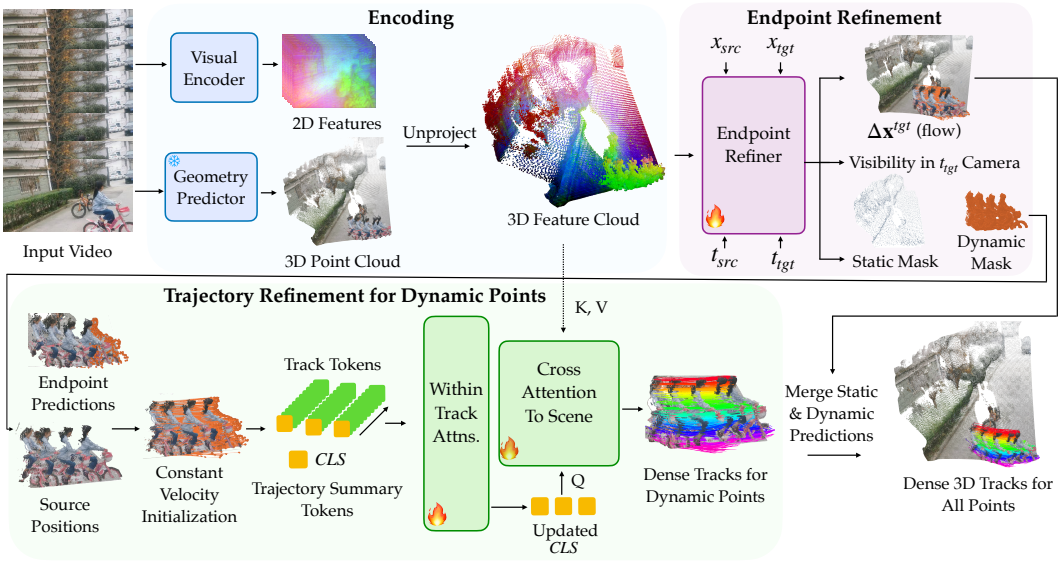


Figure 2: **TrackEverything architecture.** TrackEverything processes videos in sliding windows of length $L$. *Encoding:* Video frames are encoded and unprojected into a world-coordinate 3D feature cloud using depths/pointmaps from sensors or geometry predictors. *Endpoint Refinement:* Conditioned on source/target position embeddings and 3D WAFT feature sampling, a recurrent transformer iteratively predicts each point's 3D offset $\Delta\mathbf{x}$ at the target timestep, visibility, and a static/dynamic classification. *Trajectory Refinement:* Trajectories for dynamic points are initialized via constant-velocity interpolation and refined via within-track temporal self-attention and cross-attention from track summary tokens ([CLS]) to the scene cloud. *Output and De-duplication:* At window boundaries, points arriving at identical 3D voxels are merged by mean-pooling before initializing the subsequent window, bounding representation size across long videos.

world coordinates across temporal windows of length $L$. Within each window, frames are encoded and unprojected into a world-coordinate *feature cloud* (Section 3.1). An iterative refinement process then decouples tracking into an *endpoint refiner* (Section 3.2.1) that estimates end-of-window locations and classifies points as static or dynamic, and a *trajectory refiner* (Section 3.2.2) that decodes dense within-window paths only for dynamic points. Crucially, TrackEverything achieves dense tracking with low inference complexity by coupling sliding-window inference with cross-window voxelization (Section 3.3): rather than carrying all accumulated trajectories forward, we merge tracks that arrive at the same 3D voxel, since two physical entities cannot occupy the same position at the same time.

### 3.1 Encoding

We process the video in non-overlapping windows of $L{=}16$ frames. Within each window, every frame is encoded by a frozen DINOv3-small (Siméoni et al., 2025) backbone followed by a trainable ViT Adapter head (Chen et al., 2023), yielding a feature map of shape $(L, H/4, W/4, D)$ with $D{=}384$. Each frame's feature map is reshaped into a set of $\frac{H}{4}\times\frac{W}{4}$ tokens, paired with world-coordinate 3D positions given by the corresponding entry in the pointmap (downsampled to the same resolution). Following Wang et al. (2024), the scene geometry is normalized to a canonical scale globally in the reference frame of the first camera ($\mathrm{cam}_0$). To reduce spatial redundancy within each frame, tokens are quantized into voxels of size $v$ via $\lfloor \mathbf{x}/v \rfloor$, and tokens sharing a voxel index are merged via mean-pooling. This voxelization is performed per-frame rather than across frames to avoid erroneous merges when distinct surfaces occupy the same 3D region at different times. The per-frame voxelized tokens are concatenated across all $L$ frames to form a space-time feature cloud of shape $(N, D)$. We refer to these features individually as $\mathbf{f}_i$. Note that until we finish processing the window (Section 3.3), we retain multiple representatives of any surface observed in multiple timesteps.

### 3.2 Iterative Refinement

Following the standard point tracking approach (Harley et al., 2022; Doersch et al., 2023), we begin with zero-velocity initializations, coming from either observed 3D position (for newly disoccluded points) or the previous window's output (for tracked points), and we seek to iteratively refine these initializations into trajectories that track the scene.

Directly decoding full $L$-frame tracks for every point in the feature cloud is computationally prohibitive and largely wasteful, as many scene points in real-world environments are stationary in world

coordinates. Furthermore, *propagating tracking state across sliding windows does not require full trajectories*—to hand points off to the next window and merge co-located surfaces, the model only needs to know each point's position at the window boundary. We therefore decouple tracking into two stages: (1) An **endpoint refiner** (Section 3.2.1) that runs over all active scene points to predict their 3D positions at the window boundary and classify them as static or dynamic. (2) A **trajectory refiner** (Section 3.2.2) that decodes dense within-window tracks exclusively for dynamic points.

#### 3.2.1 ENDPOINT REFINER

The goal of the endpoint refiner is to iteratively update each point's estimated 3D location at a designated target timestep $t_{\mathrm{tgt}}$, while predicting its visibility and classifying it as static or dynamic.

*Timestep definitions.* Each point is associated with a *source timestep* $t_{\mathrm{src}}$: the frame in which it was first observed in the current window ($t_{\mathrm{src}} \in \{0, \ldots, L{-}1\}$), or $t_{\mathrm{src}} = -1$ for points carried over from prior windows. The *target timestep* $t_{\mathrm{tgt}}$ designates the destination frame for which positions are estimated. At inference time and for all intermediate windows during training, we set $t_{\mathrm{tgt}} = L - 1$ (the window boundary), as boundary destinations are precisely what is needed to advance the sliding window. On the final window during training, $t_{\mathrm{tgt}}$ is sampled uniformly at random from $\{0, \ldots, L{-}1\}$, teaching the model to predict flow to arbitrary timesteps.

*Position embeddings:* To inform the transformer of both where/when points originate and where/when they are being tracked, we enrich each point's feature $\mathbf{f}_i$ with four sinusoidal embeddings (Mildenhall et al., 2021): a spatial embedding for the initial 3D coordinate $\mathbf{x}_{\mathrm{src},i}$, a spatial embedding for the current estimated destination $\mathbf{x}_{\mathrm{tgt},i}^{(k-1)}$, a temporal embedding for the source observation frame $t_{\mathrm{src},i}$, and a temporal embedding for the target frame $t_{\mathrm{tgt}}$. Each sinusoidal embedding is projected to $D$ dimensions with a dedicated linear layer ($E_a^b$) before the embeddings are merged via a sum, yielding $\mathbf{f}_i \leftarrow \mathbf{f}_i + E_{\mathrm{src}}^{\mathrm{xyz}}(\mathbf{x}_{\mathrm{src},i}) + E_{\mathrm{tgt}}^{\mathrm{xyz}}(\mathbf{x}_{\mathrm{tgt},i}^{(k-1)}) + E_{\mathrm{src}}^{t}(t_{\mathrm{src},i}) + E_{\mathrm{tgt}}^{t}(t_{\mathrm{tgt}})$.

*Feature construction:* Before each transformer pass, we apply a 3D variant of the Warp-Aligned Feature Transform (WAFT) (Wang & Deng, 2026). For every point, we project its current target-position estimate onto the 2D feature map of frame $t_{\mathrm{tgt}}$ and bilinearly sample the image feature there, producing a *target feature* $\mathbf{g}_{\mathrm{tgt}}$. We similarly look up the *source feature* $\mathbf{g}_{\mathrm{src}}$ at its birth location, so that the model cannot "forget" the original appearance. These are concatenated with the main feature $\mathbf{f}_i$ to form the actual transformer input: $[\mathbf{f}_i;\ \mathbf{g}_{\mathrm{src},i};\ \mathbf{g}_{\mathrm{tgt},i}]$. Providing both source and target features gives the model an implicit template-matching signal: when the two features are similar, the current estimate is likely accurate; when they are dissimilar, refinement is needed. As shown by WAFT (Wang & Deng, 2026), self-attention over features constructed this way is sufficient for correspondence-finding. We note that the source feature $\mathbf{g}_{\mathrm{src}}$ only needs to be sampled once.

*Attention and decoding:* The concatenated features are processed by $N_{\mathrm{attn}}$ layers of multi-head self-attention over the full feature cloud. A small MLP head then decodes three outputs from each $\mathbf{f}_i$: a 3D residual $\Delta\mathbf{x}$, a visibility logit $v$, and a static/dynamic logit $s$. The target position is updated as $\mathbf{x}_{\mathrm{tgt}}^{(k)} = \mathbf{x}_{\mathrm{tgt}}^{(k-1)} + \Delta\mathbf{x}^{(k)}$. The updated $\mathbf{f}_i$ are carried forward to the next module and the next window, but we do not backpropagate through time.

#### 3.2.2 TRAJECTORY REFINER

After the endpoint refiner updates each point's target-time position $\mathbf{x}_{\mathrm{tgt}}^{(k)}$, and static/dynamic classification logit $s$, we refine the $L$-frame (window-length) trajectory of each dynamic point.

*Trajectory initialization:* In the first iteration, each dynamic point's trajectory is initialized by *constant*-velocity interpolation between its source and target positions. Otherwise, we use the previously estimated trajectory but with the $t_{\mathrm{tgt}}$ position updated by the endpoint refiner.

*Feature construction:* For each timestep along the trajectory, we concatenate three vectors: (i) the point's feature from the transformer output, (ii) its source feature $\mathbf{g}_{\mathrm{src}}$ (sampled at the point's birth location), and (iii) a feature bilinearly sampled from the 2D feature map at the position predicted by the constant-velocity estimate for that timestep. We also append a [CLS] token (initialized from the cloud feature) to serve as a track-level summary, yielding track features of shape $(N_{\mathrm{dyn}}, L{+}1, D)$.

*Attention and decoding:* The decoder applies multiple layers of two alternating operations: (1) within-track temporal self-attention, which lets each point's tokens exchange information along its own trajectory, and (2) cross-attention from the [CLS] token to the full feature cloud, which injects scene context back into each track. Tracks do not attend to one another, following PIPs (Harley et al.,

2022) and D4RT (Zhang et al., 2026b); this independence means we can decode a random subset of tracks during training yet decode all tracks at test time without distribution shift. After the final attention layer, an MLP decodes a per-point per-timestep 3D flow residual and visibility logit, for each of the $N_{\text{dyn}} \cdot L$ tokens (discarding the [CLS] token of each track).

**Iterative loop.** The iterative refinement steps (endpoint update and trajectory update) are repeated for $K$ iterations within each window. Each iteration re-samples the WAFT features at the newly updated target positions, giving the transformer progressively better evidence from the bilinear samples. Additionally, the xyz position embeddings to the feature cloud are updated with the most recent 3D estimates. This recurrent design is what allows the architecture to also chain predictions across sliding windows: the mechanism is identical, differing only in that the initialization comes from the previous window rather than from the previous iteration. We describe this next.

### 3.3 Sliding Window Inference and De-Duplication

At the end of each window, we voxelize the feature cloud based on predicted target positions. Unlike the per-frame voxelization used during encoding, this operation merges points across multiple origin frames. This cross-frame merging is valid because all points are now tracked to the same target timestep; we thus leverage the rule that two entities cannot occupy the same voxel at the same time.

## 4 Experiments

We evaluate TrackEverything on standard 3D point tracking benchmarks and compare against both sparse and dense tracking baselines. Beyond benchmarking our own model, we aim to provide a comprehensive evaluation across method paradigms to elucidate the trade-offs in this space.

### 4.1 3D Point Tracking Overview

**Benchmark.** We use the TAPVid-3D benchmark (Koppula et al., 2024), which comprises three datasets spanning diverse settings: Aria Digital Twin (ADT) (Pan et al., 2023) with egocentric indoor videos ($\sim$300 frames), DriveTrack (Balasingam et al., 2024) with egocentric driving videos (25–300 frames), and Panoptic Studio (PStudio) (Joo et al., 2015) with multi-camera lab captures ($\sim$150 frames). Because dense 3D ground truth is infeasible to annotate for real-world dynamic scenes, TAPVid-3D annotates and evaluates on a sparse set of query points. Evaluating on these sampled points provides an unbiased indicator of tracking fidelity across the full scene, while enabling direct comparison with sparse tracking baselines that track only the queried points. Following prior work (Karhade et al., 2026), we evaluate our method and baselines on the official minival split of 50 video clips from each TAPVid-3D dataset. Where dense ground truth is available in simulation, we evaluate dense tracking on held-out splits of PointOdyssey (Zheng et al., 2023) and Dynamic Replica (Karaev et al., 2023) in the supplementary (Table 3). We also provide extensive dense tracking visualizations on diverse real-world scenes on our project page.

**Metrics.** We report the official **APD** metric (Koppula et al., 2024), which measures the fraction of predicted points within a distance threshold of the ground truth, averaged over a set of thresholds. Prior work uses two threshold conventions. **APD-P** (point-wise) uses depth-relative thresholds obtained by unprojecting pixel thresholds $\{1, 2, 4, 8, 16\}$ px into 3D via the camera intrinsics; this is the TAPVid-3D metric used by sparse trackers. **APD-M** (metric) uses fixed distance thresholds $\{0.1, 0.3, 0.5, 1.0\}$ m, as adopted by several dense trackers. APD-P is typically stricter than APD-M. We report both to enable fair comparison across methods. We additionally report Average Jaccard (AJ) and Occlusion Accuracy (OA) for all methods in the supplementary material.

**Baselines.** We compare against three classes of methods. (a) *Sparse trackers* track user-specified query points: SpatialTracker-v2 (Xiao et al., 2025), TAPIP-3D (Zhang et al., 2026a), and CoTracker3 (Karaev et al., 2025) (a 2D tracker lifted to 3D). (b) *First-frame dense trackers* track all points visible in the first frame but cannot track points that appear later: DeltaV2 (Ngo et al., 2025) and Any4D (Karhade et al., 2026).(c) *All-frame dense trackers* track all points across all frames: VDPM (Sucar et al., 2026) and D4RT (Zhang et al., 2026b). We include D4RT's reported numbers for reference, though we note that D4RT has no publicly available code or weights, uses $\sim 20\times$ our parameter count with private training data, and is evaluated only on 48-frame clips.

**Geometry and evaluation protocol.** Several baselines accept or require external 3D geometry. We provide VGGT-$\Omega$ (Wang et al., 2026a) pointmaps to SpatialTracker-v2, TAPIP-3D, CoTracker3, Any4D, DeltaV2, and our model; for SpatialTracker-v2 and Any4D we also report numbers with their built-in geometry. VDPM and D4RT do not support external geometry. Prior dense trackers have only

Table 1: **3D point tracking on TAPVid-3D**. Ω: VGGT-Ω geometry; –: native geometry. **Bold** = best, underline = second-best in that block. D4RT is closed-source and is excluded from ranking.

| | | first 48 frames | | | | | | | | full-length video | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ADT | | DriveTrack | | PStudio | | Avg | | ADT | | DriveTrack | | PStudio | | Avg | |
| **Method** | Geo | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M | APD-P | APD-M |
| *Sparse tracking* | | | | | | | | | | | | | | | | | |
| CoTracker3 | Ω | 43.1 | 87.8 | **29.7** | **42.8** | 29.6 | 87.9 | 34.1 | **72.8** | 34.5 | 84.4 | 27.4 | 43.9 | 27.1 | 87.9 | 29.7 | **72.1** |
| SpatialTracker-v2 | – | 37.7 | 87.0 | 28.6 | 41.3 | 25.1 | 84.1 | 30.5 | 70.8 | OOM | OOM | OOM | OOM | 23.9 | 84.0 | – | – |
| SpatialTracker-v2 | Ω | 43.8 | 87.4 | 29.4 | 42.4 | 24.9 | 84.2 | 32.7 | 71.3 | 36.6 | 84.6 | **28.1** | **45.1** | 23.7 | 84.6 | 29.5 | 71.4 |
| TAPIP-3D | Ω | 45.6 | **89.0** | 26.7 | 39.4 | 30.0 | 88.2 | 34.1 | 72.2 | 37.2 | 85.0 | 24.5 | 40.3 | **27.6** | 88.2 | 29.8 | 71.2 |
| **TrackEverything (Ours)** | Ω | **45.7** | 88.7 | 28.2 | 40.3 | **30.1** | **88.7** | **34.7** | 72.6 | **40.0** | **85.1** | 26.5 | 42.4 | 27.2 | **88.5** | **31.2** | 72.0 |
| *First-frame dense tracking* | | | | | | | | | | | | | | | | | |
| ST4RTrack | – | – | – | – | – | 1.0 | – | 53.1 | – | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Any4D | – | 6.0 | 64.4 | 7.1 | 11.8 | 3.0 | 63.8 | 5.4 | 46.7 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Any4D | Ω | 6.8 | 63.2 | 7.7 | 10.9 | 4.5 | 65.8 | 6.3 | 46.6 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| DeltaV2 | Ω | 44.3 | **88.7** | **28.4** | **41.5** | **30.4** | **88.7** | 34.4 | **73.0** | 34.7 | 79.9 | **27.3** | **44.2** | **27.7** | 87.9 | 29.9 | 70.7 |
| **TrackEverything (Ours)** | Ω | **45.7** | **88.7** | 28.2 | 40.3 | 30.1 | **88.7** | **34.7** | 72.6 | **40.0** | **85.1** | 26.5 | 42.4 | 27.2 | **88.5** | **31.2** | **72.0** |
| *All-frame dense tracking* | | | | | | | | | | | | | | | | | |
| D4RT | – | 40.8 | – | 41.0 | – | 49.6 | – | 43.8 | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| VDPM | – | 4.8 | 65.0 | 14.7 | 23.3 | 13.1 | 82.1 | 10.9 | 56.8 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **TrackEverything (Ours)** | Ω | **45.7** | **88.7** | **28.2** | **40.3** | **30.1** | **88.7** | **34.7** | **72.6** | **40.0** | **85.1** | **26.5** | **42.4** | **27.2** | **88.5** | **31.2** | **72.0** |

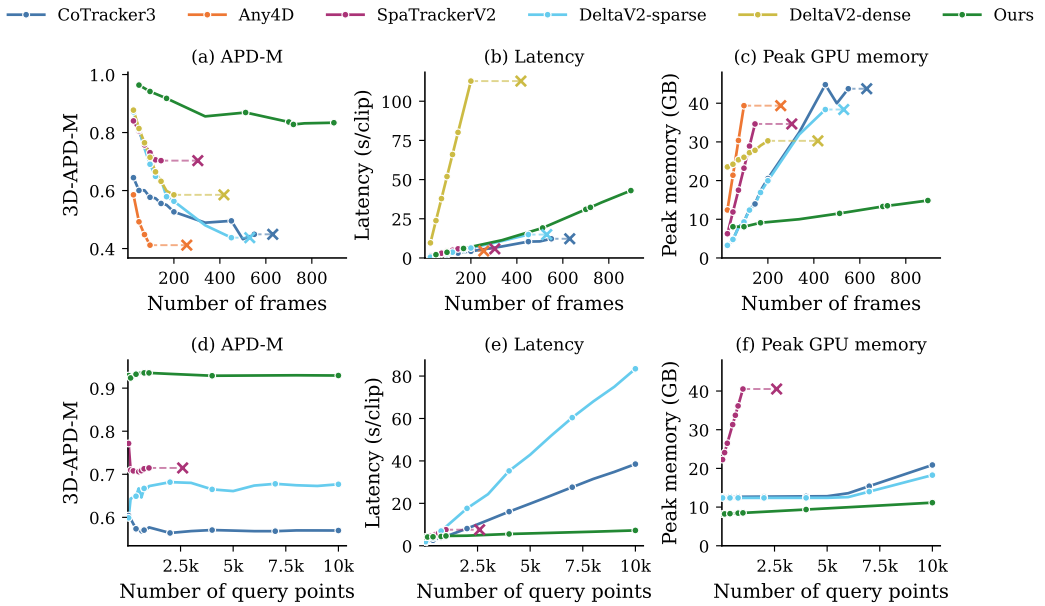


Figure 3: **Complexity analysis on PointOdyssey.** Accuracy (APD-M), end-to-end latency, and peak GPU memory vs. video length at 384 query points (top row, a–c), and query point count at 120 frames (bottom row, d–f). 'x' marks out-of-memory. Dense baselines exceed memory in 100-200 frames; sparse baselines track only a few thousand points, and OOM at 600 frames. TrackEverything uniquely tracks all points across all frames with modest latency and under 30GB GPU memory.

been evaluated on short clips (48 or 64 frames); we therefore report all methods on 48-frame clips and additionally report sparse trackers, DeltaV2, and TrackEverything on the full-length sequences.

### 4.2 Main Results

Table 1 presents the full comparison. Among the all-frame dense trackers, TrackEverything outperforms VDPM by over 20% APD-P on average. D4RT reports higher numbers on 48-frame DriveTrack and PStudio clips (and trails behind on ADT), but uses $\sim 20\times$ our parameters, trains on significantly more data including private datasets, has no public code or weights, and does not track beyond 48 frames. Neither VDPM nor D4RT scales to full-length sequences (marked N/A); TrackEverything is the only all-frame dense tracker that does. Compared to first-frame dense trackers, TrackEverything is competitive with DeltaV2 using same VGGT-Ω backend: ahead on average and on ADT and slightly behind on DriveTrack and PStudio. DeltaV2 can only track dense points originating in the first frame (and sparse user-specified points from other frames), whereas TrackEverything densely tracks all points across all frames—a strictly more demanding setting. On full-length videos, TrackEverything is competitive with sparse 3D trackers despite tracking orders of magnitude more points. Please check the appendix and our project website for qualitative results (Figure 6).

### 4.3 Complexity Analysis of Dense and Sparse Point Tracking

To understand how existing methods scale with video length and query point count, we measure accuracy (APD-M), inference latency, and peak GPU memory as we independently vary (a) input frame count and (b) query point count. All experiments run on a single L40S-46G GPU and include end-to-end inference time (Figure 3). Notably, TrackEverything densely tracks *all* points in the video in addition to any supplied query points, whereas the sparse baselines track only the queried subset. We include similar APD-P analysis in the supplementary.

**Setup.** We compare against SpatialTracker-v2 (sparse 3D), CoTracker3-3D (sparse 2D lifted to 3D), and dense tracking methods DeltaV2 and Any4D. We evaluate on the PointOdyssey (Zheng et al., 2023) validation set, which is one of the only datasets with long videos ($1000+$ frames) and fairly dense annotations ($10k+$ points). Our method, Any4D, and SpatialTracker-v2 all include PointOdyssey in their training data; CoTracker3 and DeltaV2 do not. The focus in these experiments is mainly on latency and memory trends, and not on absolute accuracy. We report APD-M here and APD-P in the supplementary. When a method exceeds GPU memory we mark it as OOM.

**Scaling with video length.** We fix the number of query points to $384$ for methods that accept this parameter. As shown in Figure 3a, accuracy degrades for all methods as the video grows longer, but the decline is notably steeper for Any4D, which processes all frames jointly. In terms of latency (Figure 3b), CoTracker3 is the fastest method, though it tracks only the $384$ queried points whereas TrackEverything tracks every point in the video. DeltaV2-dense is the slowest, exceeding 100 s per clip at 200 frames for tracking first-frame points alone. For peak GPU memory (Figure 3c), TrackEverything exhibits the lowest footprint and the slowest growth rate: it remains under 30 GB even at 900 frames. Any4D exceeds memory at 96 frames, SpatialTracker-v2 and DeltaV2-dense at approximately 200 frames, DeltaV2-sparse at approximately 500 frames, and CoTracker3 at approximately 600 frames. Three design choices contribute to our efficiency: (i) a compact 3D representation that de-duplicates spatially redundant points, (ii) WAFT-based attention in place of expensive 4D correlation volumes, and (iii) a sliding-window inference scheme.

**Scaling with number of query points.** We fix the video length to 120 frames and vary the number of query points. We omit Any4D and DeltaV2-dense, as they do not accept query-point input. As shown in Figure 3d, all methods largely maintain accuracy as the query count increases. However, their computational profiles diverge sharply. CoTracker3 and DeltaV2-sparse start with the lowest latency but surpass TrackEverything at approximately 750 points (DeltaV2-sparse) and 2,000 points (CoTracker3), as shown in Figure 3e. SpatialTracker-v2's latency rises steeply and exceeds memory at 1,000 points. TrackEverything maintains near-constant latency because additional query points do not incur expensive 4D correlation operations. In terms of memory (Figure 3f), SpatialTracker-v2 is the most demanding method, while CoTracker3 and TrackEverything both remain below 30 GB.

**Takeaways.** Existing dense trackers (Any4D, VDPM, DeltaV2-dense) do not scale to long videos in accuracy, latency, or memory. Sparse 3D trackers like SpatialTracker-v2 exhibit rapid growth in both latency and memory as either axis increases. CoTracker3 (2D sparse lifted to 3D) scales well in both dimensions but exceeds memory at approximately 600 frames and becomes slow beyond several thousand query points. TrackEverything is the only method that can densely track all points across all frames of long videos while also accepting up to 10,000 additional query points, with high accuracy, modest latency, and a GPU footprint that remains under 30 GB. Overall, TrackEverything is the first method that makes dense, long-video 3D point tracking practical on a single GPU.

### 4.4 Additional Experiments and Ablation Studies

**Geometry Source.** Table 2(a) evaluates TrackEverything across different input pointmap sources without any retraining. Replacing Pi3 with VGGT-$\Omega$ improves APD-P by $+6.6$ on PStudio ($30.1$) and $+6.4$ on ADT ($45.7$). Sensor geometry further boosts accuracy to $71.7$ on PStudio and $46.5$ on ADT, surpassing D4RT ($49.6$ and $40.8$) despite using $\sim 20\times$ fewer parameters. Crucially, while monolithic trackers like D4RT cannot ingest external geometry, TrackEverything seamlessly inherits improvements in geometry backbones and sensor hardware without retraining.

**Static/dynamic decomposition.** Table 2(b) evaluates the static/dynamic decomposition. The classifier reliably segments moving objects ($93.1$ and $92.5$ F1 on Dynamic Replica and PointOdyssey). Forcing all points through the dynamic trajectory decoder leaves tracking accuracy identical ($31.2$ APD-P) but increases latency by $4.8\times$ and peak GPU memory by $2.5\times$, confirming that concentrating compute on dynamic surfaces provides substantial efficiency gains without accuracy loss.

Table 2: **Additional experiments and ablations.** †reported by D4RT. TE: TrackEverything.

(a) Input Geometry source.

| | PStudio | ADT |
|---|---|---|
| D4RT† | 49.6 | 40.8 |
| TE+Pi3 | 23.5 | 39.3 |
| TE+VGGT-Ω | 30.1 | 45.7 |
| TE+sensor | **71.7** | **46.5** |

(b) Static/dynamic decomposition.

| | APD-P | s/fr↓ | Mem.↓ |
|---|---|---|---|
| TE | **31.2** | **0.05** | **9.1 GB** |
| − static/dyn cls | **31.2** | 0.24 | 22.4 GB |

(c) Model design.

| | APD-P |
|---|---|
| Full model | **31.2** |
| − WAFT | 29.3 |
| − iter. refin. | 26.4 |

(d) Voxel size.

| | no vox | 0.005 | 0.01 | 0.02 | 0.05 | 0.1 | 0.2 |
|---|---|---|---|---|---|---|---|
| APD-P | – | **31.4** | 31.3 | 30.9 | 28.5 | 25.8 | 23.5 |
| s/fr↓ | – | 0.23 | 0.18 | 0.11 | 0.04 | 0.04 | **0.03** |
| Peak mem.↓ (GB) | OOM | 24.0 | 21.0 | 15.0 | 11.2 | 10.8 | **10.7** |

(e) Window length.

| $L$ | APD-P | s/fr↓ | Mem.↓ |
|---|---|---|---|
| 8 | **31.3** | **0.04** | **8.1 GB** |
| 16 | 31.2 | 0.05 | 9.1 GB |
| 24 | 29.8 | 0.06 | 11.3 GB |

**Model design choices.** Table 2(c) ablates key architectural components. Removing iterative trajectory refinement reduces APD-P from 31.2 to 26.4 ($-4.8$). Removing 3D WAFT feature sampling drops APD-P to 29.3 ($-1.9$). We include qualitatives for these ablations in the supplementary.

**Voxel size.** Table 2(d) sweeps the de-duplication voxel size $v$ on PointOdyssey. Without voxelization, the model quickly runs out of memory (OOM) as it retains as many tokens as 2D representations, highlighting the necessity of de-duplication for tracking all points across all frames. Tracking accuracy remains stable from a fine voxel size of $0.005$ up to $0.02$ ($\sim$31 APD-P), dropping thereafter. Coarser voxels increase inference speed and reduce peak memory (from $0.23$ s/frame and $24.0$ GB at $0.005$ to $0.03$ s and $10.7$ GB at $0.2$). We adopt $v$=0.02 as the default, an effective trade-off that retains peak accuracy while halving latency and memory.

**Window length.** Table 2(e) ablates the window length $L \in \{8, 16, 24\}$ on TapVid-3D. $L$=8 and $L$=16 achieve similar APD-P (31.3 vs. 31.2), while $L$=24 drops to 29.8. Although $L$=8 has lower test latency, training with $L$=16 is 26% faster per step. We therefore set $L$=16.

## 5 Limitations

While TrackEverything enables all-frame dense 3D tracking over long videos, limitations remain. First, while voxelized de-duplication bounds memory growth with scene content, allowing scaling to 1000+ frames instead of 100, videos showing perpetual open-world exploration contain *linearly increasing scene content*, which leads our model to linearly increase memory. We hope our work opens the door to new research on how to deal with hours-long instead of minutes-long video. Second, like prior 3D tracking models (Zhang et al., 2026a; Xiao et al., 2025), TrackEverything relies on external geometry. This modularity is largely an advantage, as it leverages ongoing advances in feedforward 3D reconstruction (or additional sensors) without retraining, but errors in the supplied pointmaps can propagate into tracking errors. Third, the cross-window de-duplication mechanism merges tracks purely based on spatial proximity at temporal window boundaries. Because this merging is executed via irreversible mean-pooling, tracking errors can cause distinct physical surfaces to permanently collapse into a single canonical track that cannot be disentangled in subsequent frames. While users can already protect important points from being merged by specifying them through our non-voxelized query-point branch (Section A.1), future work to improve the de-duplication mechanism will be useful. Finally, scaling model capacity and incorporating broader pretraining data is likely to yield further gains on complex real-world dynamics.

## 6 Conclusion

We presented TrackEverything, a 3D point tracker that tracks all visible surfaces across long videos by representing them as persistent scene tracks in world coordinates. Grounded in the principle that complexity should scale with unique physical geometry rather than frame count, TrackEverything resolves the computational bottleneck of dense tracking via three key innovations: voxelized de-duplication across sliding windows, an endpoint-then-trajectory decomposition focusing dense decoding on moving surfaces, and 3D WAFT for efficient feature sampling without costly 4D correlations. Empirically, TrackEverything is the first tracker capable of tracking all visible points across 1000+ frame sequences within 40 GB of GPU memory, outperforming open-source all-frame dense baselines on TAPVid-3D by over 20% APD while matching state-of-the-art sparse trackers on long sequences. By making long-horizon dense 3D tracking practical, TrackEverything provides an essential foundation for extending persistent 3D world representations to dynamic environments.

## GENERATIVE AI DISCLOSURE

In this work, we used generative AI tools to assist in implementing our method. This was limited to coding assistance from AI coding agents while implementing components of the system. We have not used generative AI tools to design or provide feedback on research methodology or experiments, to support qualitative or thematic data analysis, or to interpret results. Generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, assisting with translation, and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools, to refine the writing, and to restructure tables in the paper. We have reviewed all AI-assisted work. The authors reviewed AI-assisted code before using it. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

## REFERENCES

Arjun Balasingam, Joseph Chandler, Chenning Li, Zhoutong Zhang, and Hari Balakrishnan. Drivetrack: A benchmark for long-range point tracking in real-world videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22488–22497, June 2024.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions, 2023. URL https://arxiv.org/abs/2205.08534.

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

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

Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4rtrack: Simultaneous 4d reconstruction and tracking in the world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8503–8513, October 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. 2022.

Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *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. In *ICCV*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Ayush Jain, Alexander Swerdlow, Yuzhou Wang, Sergio Arnaud, Ada Martin, Alexander Sax, Franziska Meier, and Katerina Fragkiadaki. Unifying 2D and 3D vision-language understanding. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj,

Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26717–26739. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/jain25b.html.

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

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

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6013–6022, 2025.

Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4d: Unified feed-forward metric 4d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14578–14589, June 2026.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1949–1974. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/ke25a.html.

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

Zihang Lai, Eldar Insafutdinov, Edgar Sucar, and Andrea Vedaldi. Cowtracker: Tracking by warping instead of correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 42571–42580, June 2026.

Lucy Lin, Ayush Jain, Yifan Liu, and Katerina Fragkiadaki. Qwen-3d: A generalist 3d vision-language model for spatial understanding. In *European Conference on Computer Vision*, pp. 186–204. Springer, 2026.

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 *International Conference on Learning Representations*, volume 2026, pp. 37738–37767, 2026.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Tuan 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 *International Conference on Learning Representations*, volume 2025, pp. 44620–44641, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng (Carl) Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20133–20143, October 2023.

Jenny Seidenschwarz, Qunjie Zhou, Bardienus P Duisterhof, Deva Ramanan, and Laura Leal-Taixé. Dynomo: Online point tracking by dynamic online monocular gaussian reconstruction. In *2025 International Conference on 3D Vision (3DV)*, pp. 1012–1021. IEEE, 2025.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. URL https://arxiv.org/abs/2508.10104.

Edgar Sucar, Zihang Lai, Eldar Insafutdinov, and Andrea Vedaldi. Dynamic point maps: A versatile representation for dynamic 3d reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7295–7305, October 2025.

Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-dpm: 4d video reconstruction with dynamic point maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14502–14511, June 2026.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, and Radu Soricut et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Bo Wang, Jian Li, Yang Yu, Li Liu, Zhenping Sun, and Dewen Hu. Scenetracker: Long-term scene flow estimation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(9): 8305–8312, 2025.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-■. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21486–21499, June 2026a.

Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19795–19806, October 2023.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, June 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. Pi3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations*, volume 2026, pp. 10481–10497, 2026b.

Yihan Wang and Jia Deng. Waft: Warping-alone field transforms for optical flow. In *International Conference on Learning Representations*, volume 2026, pp. 157389–157402, 2026.

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

Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: Advancing 3d point tracking with explicit camera motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6726–6737, October 2025.

Bowei Zhang, Lei Ke, Adam Harley, and Katerina Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. *Advances in Neural Information Processing Systems*, 38:135284–135303, 2026a.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7382–7392, June 2026b.

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.

Siyuan Zhou, Yilun Du, Yuncong Yang, Lei Han, Peihao Chen, Dit-Yan Yeung, and Chuang Gan. Learning 3d persistent embodied world models, 2025. URL https://arxiv.org/abs/2505.05495.

# A Appendix

## A.1 Implementation details

**Supervision.** We supervise the predicted target positions/trajectories with an L2 loss against 3D ground-truth. Visibility and static/dynamic classification are each supervised with cross-entropy loss. Because our training datasets provide ground-truth tracks for only a subset of points, we compute losses only over the labelled points. To save compute, we decode full trajectories for a subset of dynamic points at training time.

**Training recipe.** TrackEverything has 41M trainable parameters. The DINOv3-small backbone (20M) is frozen; the ViT Adapter head and all other modules are trained from scratch, except for the transformer attention layers which are initialized from the last layers of DINOv2-small (Oquab et al., 2024). We train on 8 L40S-46GB GPUs with a per-GPU batch size of 1 and a learning rate of $1\times10^{-4}$. Training proceeds in two stages: 100k iterations on 32-frame videos, followed by 300k iterations on a mix of 32- and 64-frame videos. Point clouds are normalized to unit norm following DUSt3R (Wang et al., 2024). We apply color jittering, random image/depth blur, and erase augmentations following TAPIP-3D (Zhang et al., 2026a). Our training data consists of Kubric (Greff et al., 2022), PointOdyssey (Zheng et al., 2023), and Dynamic Replica (Karaev et al., 2023), each of which provide 3D track annotations for a sparse set of points.

**Sparse query points (optional).** In addition to dense tracking, TrackEverything can accept a set of query points specified as $(t, x, y)$ tuples. This is useful for tracking points at subpixel locations. Their features are obtained by bilinearly sampling the 2D feature map at the given frame and pixel coordinates. These query features are concatenated with the feature cloud and processed identically to grid-sampled points throughout the transformer and trajectory decoder, with the sole difference that they are excluded from voxelization so that they are never merged with nearby points.

## A.2 Dense Tracking on Dynamic Replica and PointOdyssey

Because TAPVid-3D provides only sparse query annotations, we additionally evaluate tracking on 48 frame clips of Dynamic Replica and PointOdyssey under estimated (EST) and ground-truth (GT) geometry (Table 3).

Table 3: **Tracking Performance on Dynamic Replica and PointOdyssey.** DeltaV2 and TrackEverything use geometry from Pi3 in EST setups and simulator GT geometry in GT setups. All evaluations run on 48 frames.

| | Dynamic Replica (EST) | | | | | Dynamic Replica (GT) | | | | | PointOdyssey (EST) | | | | | PointOdyssey (GT) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | APD-P | APD-M | AJ | OA | EPE↓ | APD-P | APD-M | AJ | OA | EPE↓ | APD-P | APD-M | AJ | OA | EPE↓ | APD-P | APD-M | AJ | OA | EPE↓ |
| VDPM | 13.1 | 73.5 | 8.1 | 93.1 | 0.23 | – | – | – | – | – | 15.8 | **71.8** | 9.3 | 84.1 | **0.29** | – | – | – | – | – |
| Any4D | 19.5 | 79.6 | 12.7 | **93.7** | 0.18 | – | – | – | – | – | 10.6 | 62.2 | 5.8 | 81.3 | 0.45 | – | – | – | – | – |
| DeltaV2 | 36.2 | **86.9** | 25.8 | 88.1 | **0.14** | 67.3 | 98.4 | 55.0 | 90.0 | 0.03 | 23.8 | 69.9 | 14.9 | 83.0 | 0.88 | 33.2 | 81.0 | 22.2 | 73.1 | 0.23 |
| **TrackEverything (Ours)** | **36.6** | **86.9** | **26.3** | 91.8 | **0.14** | **85.2** | **99.7** | **76.5** | **95.8** | **0.02** | **24.9** | 70.2 | **16.3** | **84.2** | 0.46 | **73.2** | **96.9** | **59.0** | **89.2** | **0.05** |

TrackEverything matches or outperforms all baselines on both datasets. Under estimated Pi3 geometry, TrackEverything is competitive with DeltaV2 (36.6 vs. 36.2 APD-P on Dynamic Replica; 24.9 vs. 23.8 on PointOdyssey) while still tracking all points in all frames rather than first-frame points only; with GT geometry the gap widens substantially (85.2 vs. 67.3 APD-P on Dynamic Replica; 73.2 vs. 33.2 on PointOdyssey).

On full-length 1000+ frame videos of PointOdyssey, and evaluation on all labelled points (10k+), TrackEverything achieves APD-P of 31.8, APD-M of 74.5, static/dynamic accurary of 91.2% (78.5% dynamic F1, 92.5% static F1). Since, no existing baselines can run in this setting, we do not report any quantitative comparisons here.

We also show dense tracking results on diverse real-world scenes on our project page.

### A.3 Inference Speed Breakdown

Table 4: **Early merge on full-length TAPVid-3D.** APD-P uses VGGT-Ω, as in Table 1. FPS averages are the mean of ADT, DriveTrack, and PStudio.

(a) Accuracy and average speed, with and without early merge.

| | APD-P | | | | Avg FPS↑ | |
|---|---|---|---|---|---|---|
| | ADT | DriveTrack | PStudio | Avg | w/ VGGT-Ω | tracker |
| w/o early merge | 40.0 | **26.5** | 27.2 | 31.2 | 5.6 | 12.6 |
| w/ early merge | **40.1** | 26.4 | **27.3** | **31.3** | **7.2** | **23.1** |

(b) FPS with early merge.

| | ADT | DriveTrack | PStudio | Avg |
|---|---|---|---|---|
| w/ VGGT-Ω | 4.5 | 8.7 | 8.4 | 7.2 |
| tracker only | **8.4** | **19.5** | **41.5** | **23.1** |

**Early Merge at Endpoints After First Iteration.** While point de-duplication is primarily performed at sliding-window boundaries, our architecture supports an optional early-merge step (`voxelize_on_endpoints`) immediately following iteration 0 of the endpoint refiner. Because static/dynamic classification converges within a single iteration, points classified as static that project to the same 3D spatial voxel at their predicted destinations are de-duplicated early. This aggressively prunes redundant static tokens before iterations $k \geq 1$ and completely bypasses trajectory refinement for these points. Crucially, dynamic points and evaluated query cells are preserved 1:1, ensuring moving surface tracks are not conflated prior to full within-window trajectory decoding.

As shown in Table 4, this static-only early merge acts as a zero-cost acceleration: on full-length TAPVid-3D, tracking accuracy remains unchanged (31.3 vs. 31.2 APD-P) while providing substantial speedups (Table 4a). Average tracker-only throughput nearly doubles from 12.6 to 23.1 FPS (1.8×), and end-to-end throughput (including feedforward VGGT-Ω geometry estimation) rises from 5.6 to 7.2 FPS (1.3×). In settings where geometry is provided directly by onboard sensors or precomputed pointmaps, only the tracker runs, operating at 23.1 FPS on average (Table 4b). Per-dataset throughput directly reflects physical scene extent: focused lab scenes like PStudio run at speeds of 41.5 FPS (and 19.5 FPS on DriveTrack), whereas expansive multi-room environments in ADT run at 8.4 FPS due to higher unique surface accumulation.

### A.4 Training Data and Model Sizes

TrackEverything is trained on Kubric (Greff et al., 2022), PointOdyssey (Zheng et al., 2023), and Dynamic Replica (Karaev et al., 2023), and has 61M parameters (with a frozen 20M DINOv3-small backbone). SpatialTracker-v2 is trained on these three plus 14 additional datasets. Any4D is trained on these three, Waymo-DriveTrack, and 4 more datasets. VDPM initializes from a VGGT backbone and trains on Kubric, PointOdyssey, Waymo, and 2 static datasets. TAPIP-3D and DeltaV2 are trained only on Kubric. CoTracker3 is trained on Kubric and on additional 15k real-world videos. D4RT is trained on all of the above, 8 additional public datasets, and undisclosed internal datasets, with a 1B-parameter backbone and 144M-parameter decoder. Please check the respective papers for more details.

### A.5 Additional Method and Implementation Details

**Visual Illustrations.** We visually illustrate two components of our model, sliding-window de-duplication and 3D WAFT, in Figures 4 and 5.

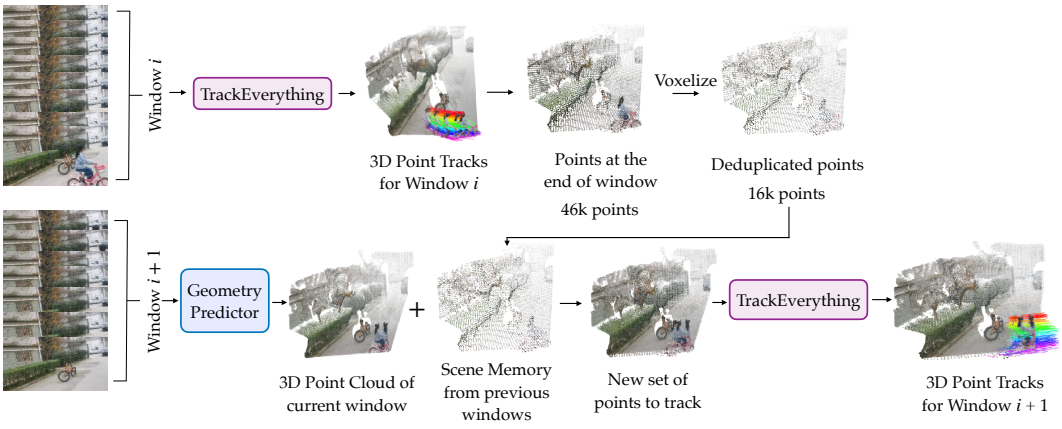


Figure 4: **Sliding-window tracking and cross-window de-duplication pipeline.** *Top row (Window $i$):* A temporal window of frames is processed by TrackEverything to predict dense 3D tracks across all points. At the final timestep of the window, the predicted 3D point locations (here 46k points) are voxelized by quantizing coordinates into spatial voxels and mean-pooling co-located tokens. This cross-frame de-duplication prunes redundant surface observations (reducing point count from 46k to 16k). *Bottom row (Window $i+1$):* When the subsequent frame window arrives, its frames are lifted into a 3D point cloud using a feedforward geometry predictor (or sensor depths). These newly observed points are concatenated with the de-duplicated points carried over from previous windows (serving as persistent 3D scene memory) to construct the active set of points to track. TrackEverything then produces dense 3D trajectories for all points across all timesteps of Window $i+1$. By chaining sliding-window processing with boundary de-duplication, TrackEverything tracks extended video sequences while bounding active token counts to unique physical scene geometry rather than video duration.

**Track Lineage and Historical Trajectory Reconstruction.** While cross-window voxelization bounds active GPU memory by collapsing co-located points, full historical trajectories $\mathbf{X} \in \mathbb{R}^{N\times T\times 3}$ for all original surface points across the video can be reconstructed. During streaming inference, each voxelization step records an integer point-to-voxel mapping array $\mathbf{p} \in \mathbb{N}^{N_{\text{pre}}}$ and a spatial offset residual $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}_{\lfloor \mathbf{x}_i/v \rfloor} \in \mathbb{R}^3$. When multiple observations merge into a single active voxel token, birth metadata is preserved by prioritizing the earliest observation: the source birth timestep $t_{\text{src}}$ and initial unprojected coordinate $\mathbf{x}_{\text{src}}$ are retained via an `argmin` reduction over the merged cluster. To export full dense trajectories for all original pixels $(t, h, w)$, the model performs an exact inverse unvoxelization by traversing the sequence of chained point-to-voxel mappings in reverse and restoring the stored spatial residuals:

$$\mathbf{X}_{\text{orig}}(t) = \mathbf{X}_{\text{vox}}[\mathbf{p}](t) + \mathbf{r}. \tag{1}$$

Because the spatial residual $\mathbf{r}$ broadcasts across all timesteps, tracking fidelity is preserved for individual surface elements. Crucially, this design decouples active tracking state from image features: once a sliding window finishes processing, its large 2D image feature maps are immediately released from GPU memory, retaining only the compact 3D point tracks and the 3D to 2D mapping tables.

**Static/Dynamic Ground-Truth Formulation.** We derive binary static/dynamic ground truth labels $y_{\text{dyn}} \in \{0, 1\}$ from the world-space displacement of each trajectory over the temporal window $[t_{\text{start}}, t_{\text{end}}]$. For each track, we evaluate the spatial extent across its annotated frames $\mathcal{V} \subseteq [t_{\text{start}}, t_{\text{end}}]$ as the L2 norm of its axis-aligned bounding box:

$$\Delta_{\text{motion}} = \left\| \max_{t\in\mathcal{V}} \mathbf{x}(t) - \min_{t\in\mathcal{V}} \mathbf{x}(t) \right\|_2. \tag{2}$$

A trajectory is labeled dynamic ($y_{\text{dyn}} = 1$) if $\Delta_{\text{motion}} > \tau_{\text{dyn}}$ and static ($y_{\text{dyn}} = 0$) otherwise, where $\tau_{\text{dyn}} = 0.05$ in normalized scene units (or $0.10$ for unnormalized synthetic clips).

**Static/Dynamic Classifier Robustness and Error Dynamics.** Our static/dynamic decomposition relies on a binary classifier at the endpoint stage. The classifier is designed such that edge-case misclassifications degrade gracefully. Specifically, if a truly static surface point is erroneously predicted as dynamic, it passes into the trajectory refiner. Because the trajectory refiner is trained on arbitrary 3D motions, including stationary points, it can simply predict a zero-displacement trajectory, leaving tracking accuracy unaffected at the expense of marginal extra compute for that token. Conversely, if a dynamic point is erroneously classified as static within a window, its trajectory is held stationary for that window duration; upon entering subsequent sliding windows, its motion state is re-evaluated, allowing it to resume dynamic tracking once the displacement becomes prominent.

**3D WAFT Projection and Boundary Conditions.** The 3D Warp-Aligned Feature Transform (3D WAFT) samples appearance evidence by projecting 3D candidate positions onto 2D feature maps. Camera extrinsics and intrinsics are either obtained from onboard sensors or estimated via feedforward geometry backbones (Wang et al., 2026a;b). When points project outside the visible image frame during rapid camera ego-motion, coordinates are clamped within an extended image canvas $[-2W, 2W] \times [-2H, 2H]$ and bilinearly sampled using boundary-replicate padding. Points that are occluded or remain outside the true field of view produce an appearance mismatch between the source feature $\mathbf{g}_{\mathrm{src}}$ and target sample $\mathbf{g}_{\mathrm{tgt}}$, which provides an explicit signal for the transformer's visibility head to predict low visibility logits.

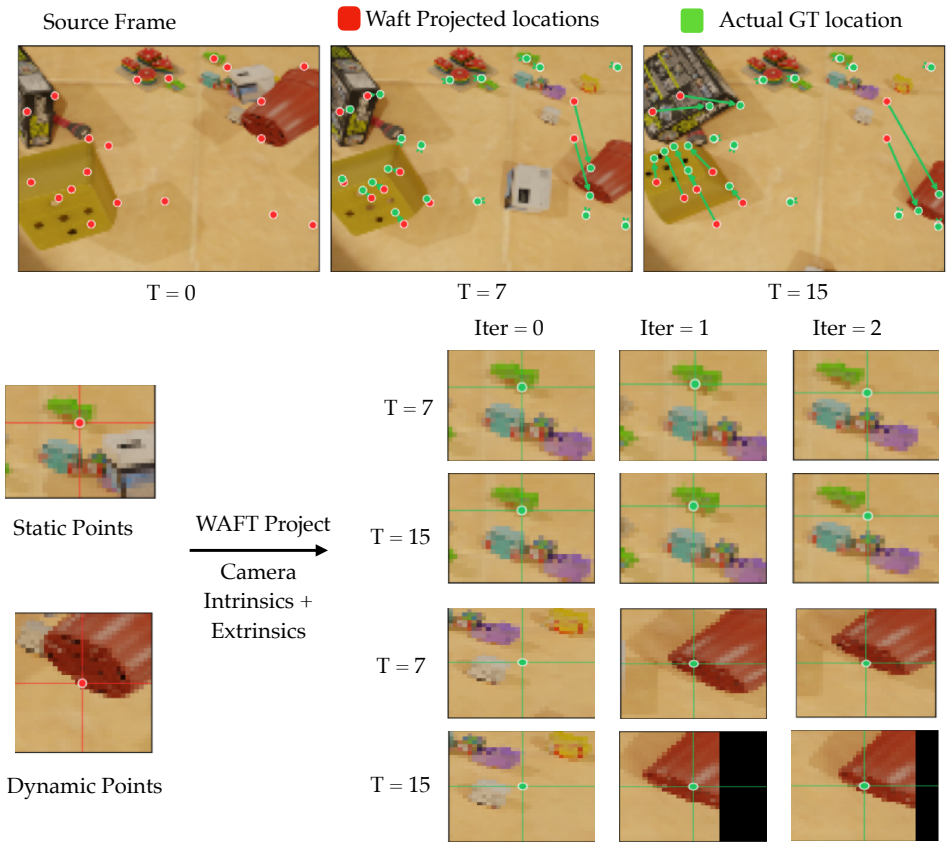


Figure 5: **3D WAFT projection and iterative refinement dynamics.** *Top:* Projections across time under zero-velocity initialization. Source points initialized at frame $T = 0$ (red dots) are projected to future timesteps ($T = 7$ and $T = 15$) using camera extrinsics and intrinsics. Red dots denote projected 2D locations under the zero-velocity prior ($\mathbf{x}_{\text{tgt}}^{(0)} = \mathbf{x}_{\text{src}}$ in world space); green dots mark true ground-truth (GT) locations, with green arrows at $T = 15$ indicating displacement error. For static objects, camera projection accounts for ego-motion and aligns projected points near GT; for dynamic objects, world motion leads to significant spatial offset. *Bottom:* Zoomed-in patches across refinement iterations (Iter $\in \{0, 1, 2\}$) at $T = 7$ and $T = 15$. *Static point (top):* The zero-velocity initialization at Iter $= 0$ already places the sampled point close to the true location. The bilinearly sampled target feature $\mathbf{g}_{\text{tgt}}$ closely matches the source feature $\mathbf{g}_{\text{src}}$, signaling to the model that no large displacement is required and maintaining stability across iterations 1 and 2. *Dynamic point (bottom):* At Iter $= 0$, the zero-velocity projection misses the moving object by multiple pixels (landing on the background), creating an appearance mismatch between $\mathbf{g}_{\text{src}}$ and $\mathbf{g}_{\text{tgt}}$ in the concatenated WAFT feature. Driven by this mismatch, the endpoint refiner iteratively updates 3D positions: by Iter $= 1$ the point moves substantially closer, and by Iter $= 2$ the sampled point locks onto the correct surface feature.

**Coordinate System and Voxelization Details** All 3D scene points and camera poses are expressed in a unified world coordinate frame anchored to the initial camera ($\text{cam}_0$, with pose $[\mathbf{I} \mid \mathbf{0}]$). Following DUSt3R (Wang et al., 2024), the coordinate system is normalized by a global scale factor defined once at the sequence level, ensuring that the metric scale and spatial origin remain strictly consistent throughout the video. As consecutive sliding windows stream in, newly observed surfaces are unprojected directly into this existing, shared $\text{cam}_0$ world coordinate frame.

Within this persistent world space, voxelization assigns each 3D point $\mathbf{x} \in \mathbb{R}^3$ an integer lattice coordinate by quantizing with a fixed edge length $v$ (default $0.02$):

$$\mathbf{k} = \lfloor \mathbf{x}/v \rfloor \in \mathbb{Z}^3. \tag{3}$$

Tokens that share identical integer lattice coordinates are merged via mean-pooling. Because voxelization is implemented via sparse hashing over the discrete integer coordinates $\mathbf{k}$, it dynamically allocates indices only for occupied cells rather than discretizing a bounded volumetric grid. Consequently, our representation naturally supports arbitrary, expanding scenes without requiring spatial bounds, coordinate clipping, or truncation.

**Optimization Mechanics and Gradient Flow.** To train our model across extended sequences without running out of GPU memory, we avoid backpropagating through time across sliding windows while still allowing the 2D visual encoder (the ViT Adapter) to receive supervisory signal across the entire video. Specifically, 2D feature maps $\mathbf{F} = \mathcal{E}(\mathbf{I})$ are extracted across the sequence and wrapped in a detached leaf tensor $\mathbf{F}_{\text{param}}$ with gradients enabled. During sliding-window execution, forward and backward passes are evaluated window-by-window with recurrent state detached at window boundaries. Within each window, the combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{coord}} + \lambda_{\text{vis}}\mathcal{L}_{\text{vis}} + \lambda_{\text{dyn}}\mathcal{L}_{\text{dyn}} + \lambda_{\text{conf}}\mathcal{L}_{\text{conf}} \tag{4}$$

is computed, and a local backward pass immediately accumulates gradients into the parameter gradient buffer $\mathbf{F}_{\text{param}}.\text{grad}$ and the tracker modules. This releases intermediate activations after each window. Once all windows have completed, a single backward pass is executed through the 2D visual backbone via the scalar inner product $\langle \mathbf{F}_{\text{param}}.\text{grad}, \mathbf{F} \rangle$, propagating the accumulated visual gradients into the trainable ViT Adapter head in a single step before calling `optimizer.step()`.

**Fair comparisons.** Reproducing sparse baselines on TAPVid-3D has proven difficult due to the lack of standardized evaluation code. Some prior dense 3D tracking papers (Sucar et al., 2026) do not compare against sparse trackers at all. We carefully re-evaluate all baselines under a unified protocol and find, for example, that SpatialTracker-v2 (Xiao et al., 2025) achieves over 20% higher APD-M than what prior work (Karhade et al., 2026) reports for it on TAPVid-3D.

### A.6 Fixes to the PointOdyssey Dataset

During our training runs, we observed significant loss spikes, which we could trace back to the PointOdyssey dataset. We identified several critical annotation errors and developed a programmatic correction pipeline:

1. *Off-by-One Camera Extrinsics:* Across 16 sequences (about 25k frames), the recorded extrinsic $\mathbf{T}_t$ is the pose that rendered frame $t-1$, while the 3D tracks are indexed correctly. We reindex the camera to frame $t+1$ and recompute the 2D tracks from that pose. The shift is a few pixels when the camera is slow and exceeds $600\,\text{px}$ during fast pans (up to $59^\circ/\text{frame}$).
2. *Corrupted Visibility Flags:* Visibility was computed from the uncorrected pose, so the stored flags are wrong where the two poses disagree. We do not rewrite them. Point supervision is dropped on the 502 frames where the repair moves tracks by at least $8\,\text{px}$ ($2\%$ of the affected sequences; at most $3.8\%$ of any one sequence). The images stay in the clip, and the camera reindexing above still supervises the remaining frames.
3. *Temporally Incoherent Sequences:* The sequence `scene_recording_20210910_S05_S06_0_ego1` exhibits sub-frame camera jitter that cannot be resolved by an integer frame shift; we exclude it from training and evaluation splits.

We also briefly explored using object segmentation masks to remove wrong point track labels by using the fact that a point born within the segmentation mask of an object should always stay within the mask. However, we found that the segmentation mask had bugs too, which makes this approach infeasible.

We will release the corrections to the PointOdyssey dataset with our codebase. With these fixes, we noticed reduced loss spikes. We note, however, that some noisy points still remain in the dataset, which we could not fix.

### A.7 QUALITATIVES

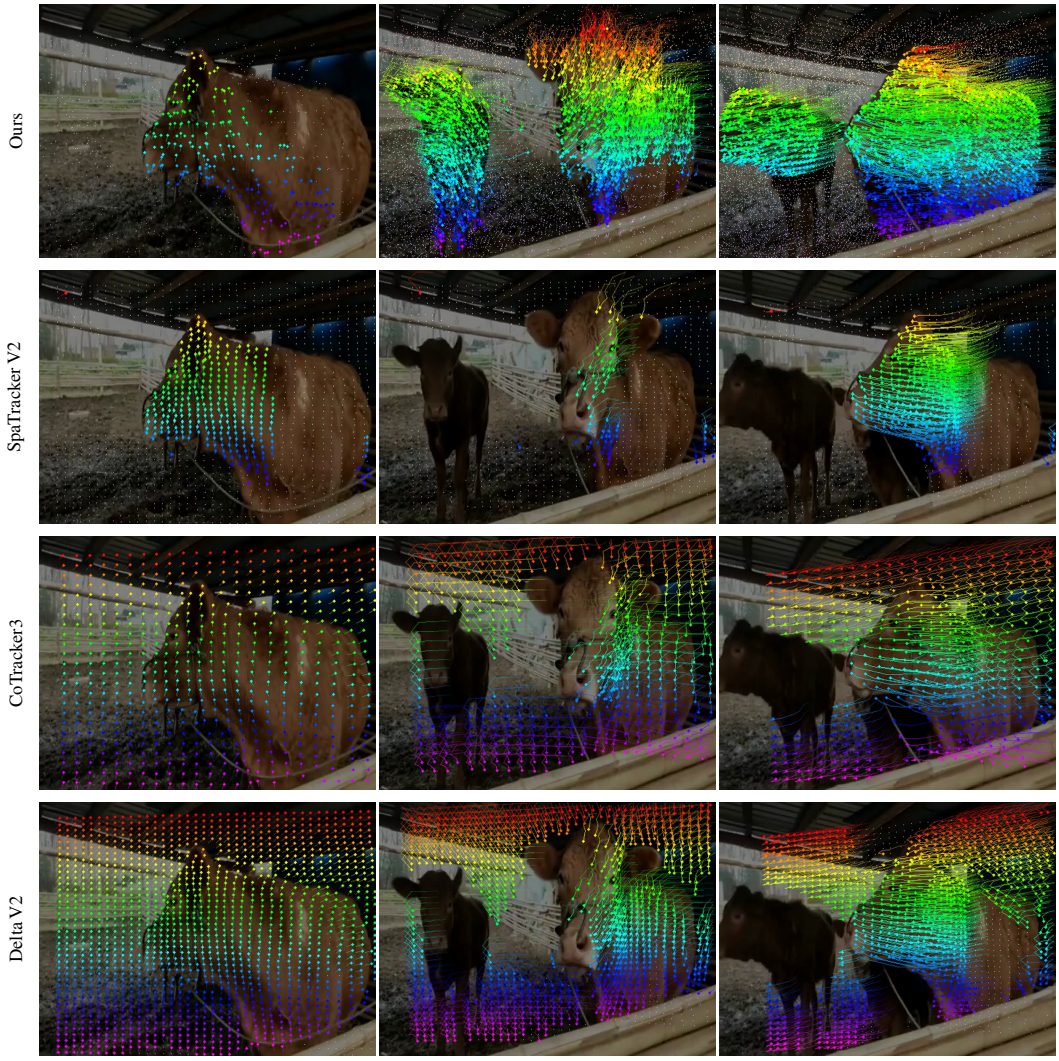


Figure 6: **Qualitative comparison against baselines.** TrackEverything tracks every visible point across time, including newly emerging content. Notice that only one cow is visible in the first frame, while the second cow emerges later. Sparse trackers need human-specified queries and are usually demoed with a first-frame grid; densifying queries over later frames quickly becomes prohibitive (Figure 3). First-frame dense trackers have the same blind spot by design. In all visualizations, we show trails for dynamic points only; the method tracks all points and accounts for camera motion, but we omit static-point trails in the 2D renderings for clarity.

Figure 6 compares TrackEverything against SpatialTracker-v2, CoTracker3, and DeltaV2. TrackEverything produces dense tracks over all visible points, including regions that appear after the first frame, while baselines either track only queried/first-frame points or leave large untracked areas.

We show additional qualitative comparisons against baselines on Breakdance, Dance-jump, and a hand-manipulation sequence (Figure 7), qualitative ablations of iterative refinement and WAFT (Figure 8), and static/dynamic classification visualizations (Figure 9). Without iterative refinement or WAFT, tracks fragment or drift on fast motion, while enabling each recovers coherent long trails. Figure 9 shows per-point static/dynamic labels over time: dynamic points in red, static in blue.




Figure 7: **Qualitative comparison against baselines** on the Breakdance sequence (Dance-jump and hand manipulation continue on the following pages). Rows show TrackEverything, SpatialTracker-v2, CoTracker3, and DeltaV2; time advances left to right. TrackEverything tracks every visible point across time, including newly emerging content. Sparse trackers need human-specified queries and are usually demoed with a first-frame grid; densifying queries over later frames quickly becomes prohibitive (Figure 3). First-frame dense trackers have the same blind spot by design. In all visualizations, we show trails for dynamic points only; the method tracks all points and accounts for camera motion, but we omit static-point trails in the 2D renderings for clarity.




Figure 7: **Qualitative comparison against baselines** (continued). Dance-jump sequence.

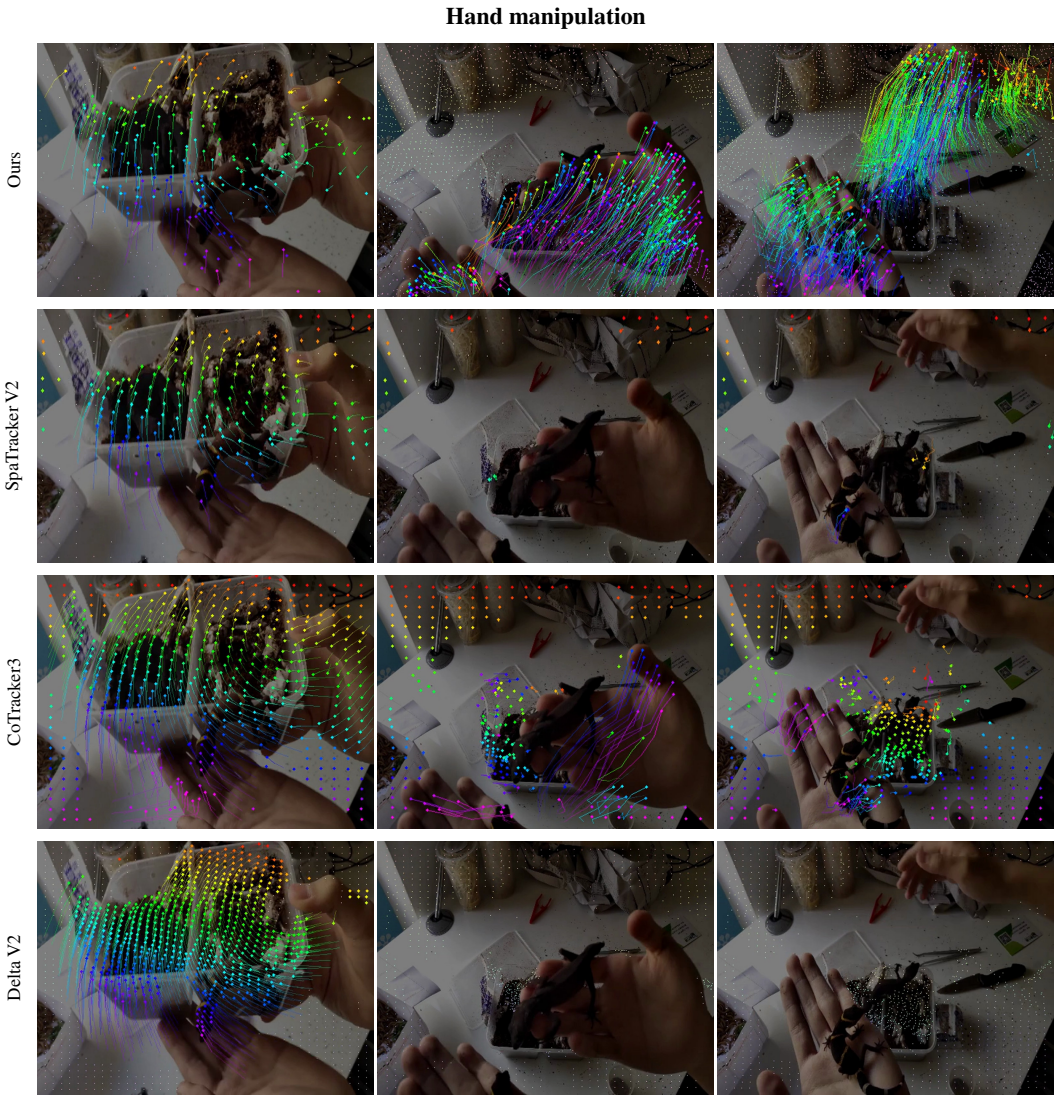


Figure 7: **Qualitative comparison against baselines** (continued). Hand-manipulation sequence.

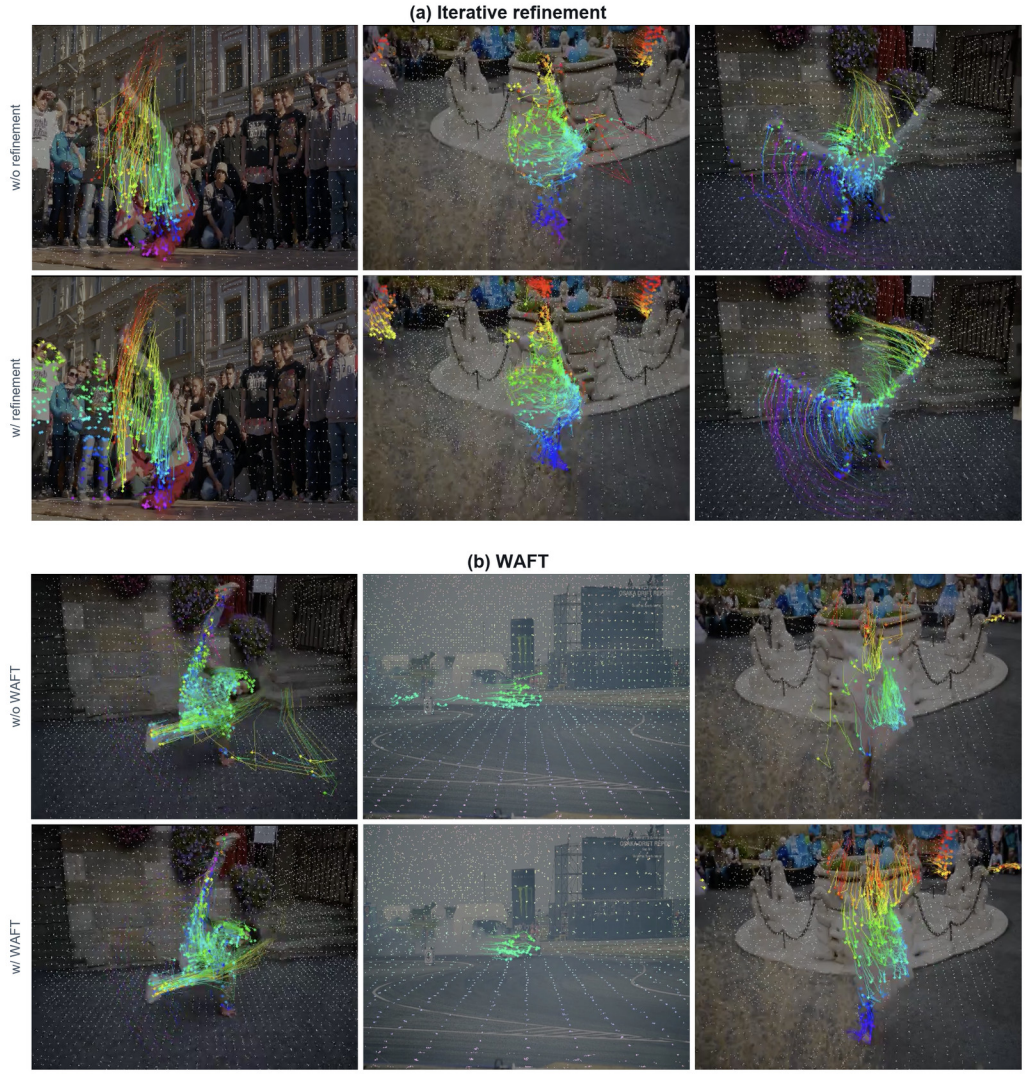


Figure 8: **Qualitative ablations of iterative refinement and WAFT.** (a) Without iterative refinement, tracks are noisier and less coherent; with refinement, trajectories follow motion more cleanly. (b) Without WAFT, tracks drift with long stray trails on fast motion; with WAFT, trajectories remain tightly localized on the moving subject.

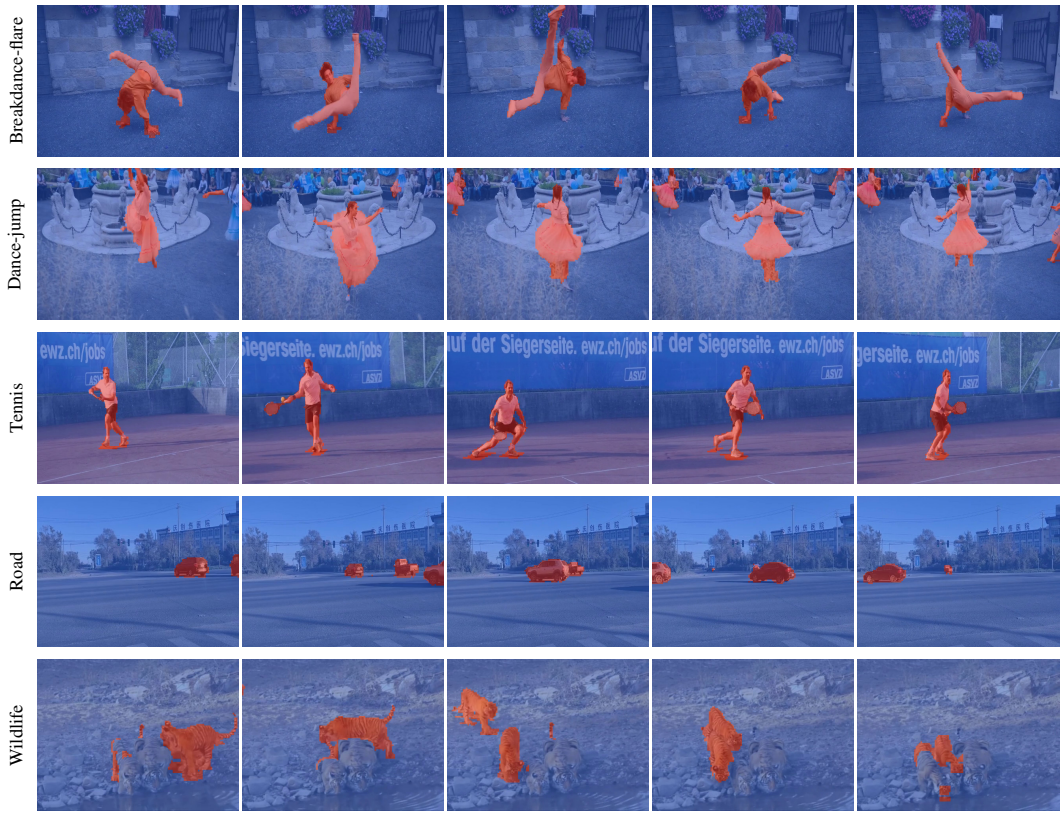


Figure 9: **Static/Dynamic Classification.** Points classified as dynamic are shown in red, static in blue. Time advances left to right. Notice how in the last example, the model correctly classifies some tigers as static while others as dynamic, and the static/dynamic assignment shifts as objects start or stop moving within a window.

### A.8 TAPVID-3D DETAILED METRICS

Tables 5 and 6 report AJ, OA, and EPE alongside APD-P / APD-M on TAPVid-3D under the same 48-frame and full-length regimes as Table 1. All conclusions remain similar to those drawn from Table 1 in the main paper.

Table 5: **TAPVid-3D detailed metrics, 48-frame.** Bold / underline: best / second-best within the block (D4RT excluded). P/M: APD-P/APD-M; AJ: Average Jaccard; OA: occlusion accuracy; EPE: end-point error. Ω: VGGT-Ω; Pi3: Pi3; UD: UniDepth; –: native geometry.

| | | ADT | | | | | DriveTrack | | | | | PStudio | | | | | Avg | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | Geo | P | M | AJ | OA | EPE↓ | P | M | AJ | OA | EPE↓ | P | M | AJ | OA | EPE↓ | P | M |
| *Sparse tracking* | | | | | | | | | | | | | | | | | | |
| CoTracker3 | Pi3 | 38.7 | 88.0 | 27.2 | 88.8 | 0.16 | 29.2 | 41.9 | 20.3 | 88.5 | 2.13 | 23.7 | 83.1 | 14.5 | 85.0 | 0.18 | 30.5 | 71.0 |
| CoTracker3 | Ω | 43.1 | 87.8 | 31.3 | 88.8 | 0.16 | 29.7 | **42.8** | **21.4** | 88.5 | 2.33 | 29.6 | 87.9 | 19.0 | 85.0 | 0.13 | 34.1 | **72.8** |
| SpatialTracker-v2 | – | 37.7 | 87.0 | 26.9 | 90.9 | **0.15** | 28.6 | 41.3 | 20.0 | **89.1** | **1.62** | 25.1 | 84.1 | 17.2 | 82.1 | 0.15 | 30.5 | 70.8 |
| SpatialTracker-v2 | Pi3 | 39.9 | 88.1 | 28.9 | **91.0** | 0.16 | **29.8** | 42.5 | 20.6 | 89.0 | 1.63 | 20.6 | 80.7 | 12.5 | 83.7 | 0.20 | 30.1 | 70.4 |
| SpatialTracker-v2 | Ω | 43.8 | 87.4 | 32.9 | 90.7 | 0.16 | 29.4 | 42.4 | 21.1 | 88.8 | 1.97 | 24.9 | 84.2 | 15.8 | 82.2 | 0.17 | 32.7 | 71.3 |
| TAPIP-3D | Pi3 | 39.6 | 88.5 | 28.0 | 89.2 | 0.16 | 28.5 | 41.0 | 19.8 | 87.3 | 1.84 | 23.9 | 83.4 | 14.8 | 85.1 | 0.16 | 30.7 | 71.0 |
| TAPIP-3D | Ω | 45.6 | **89.0** | 34.0 | 88.6 | **0.15** | 26.7 | 39.4 | 19.2 | 87.3 | 2.31 | 30.0 | 88.2 | 19.5 | 85.5 | 0.12 | 34.1 | 72.2 |
| **TrackEverything (Ours)** | Pi3 | 39.3 | 88.9 | 28.5 | 88.8 | 0.16 | 27.7 | 39.8 | 19.4 | 84.6 | 2.04 | 23.5 | 83.6 | 15.1 | 88.4 | 0.16 | 30.2 | 70.8 |
| **TrackEverything (Ours)** | Ω | **45.7** | 88.7 | **34.4** | 90.2 | **0.15** | 28.2 | 40.3 | 20.1 | 83.4 | 2.26 | **30.1** | **88.7** | **20.2** | **88.9** | **0.11** | **34.7** | 72.6 |
| *First-frame dense tracking* | | | | | | | | | | | | | | | | | | |
| ST4RTrack | – | – | – | – | – | – | – | 1.0 | – | – | 14.63 | – | 53.1 | – | – | 0.21 | – | – |
| Any4D | – | 6.0 | 64.4 | 3.5 | 90.0 | 0.35 | 7.1 | 11.8 | 4.9 | 82.2 | 4.89 | 3.0 | 63.8 | 1.7 | 86.2 | 0.30 | 5.4 | 46.7 |
| Any4D | Pi3 | 6.9 | 64.0 | 4.3 | 86.4 | 0.35 | 6.9 | 9.7 | 3.8 | 81.2 | 4.52 | 3.7 | 64.1 | 2.1 | 83.5 | 0.31 | 5.8 | 45.9 |
| Any4D | Ω | 6.8 | 63.2 | 4.0 | 86.4 | 0.35 | 7.7 | 10.9 | 4.2 | 81.7 | 4.51 | 4.5 | 65.8 | 2.5 | 84.0 | 0.29 | 6.3 | 46.6 |
| DeltaV2 | Pi3 | 39.5 | 88.4 | 28.2 | 88.2 | 0.16 | **28.4** | 41.1 | 19.6 | 84.1 | **1.71** | 24.9 | 83.7 | 15.5 | 84.3 | 0.16 | 30.9 | 71.1 |
| DeltaV2 | Ω | 44.3 | 88.7 | 32.4 | 87.6 | **0.15** | **28.4** | **41.5** | **20.2** | 84.3 | 2.02 | **30.4** | **88.7** | 20.0 | 85.1 | **0.11** | 34.4 | **73.0** |

Table 5 (continued)

| Method | Geo | ADT P | ADT M | ADT AJ | ADT OA | ADT EPE↓ | DriveTrack P | DriveTrack M | DriveTrack AJ | DriveTrack OA | DriveTrack EPE↓ | PStudio P | PStudio M | PStudio AJ | PStudio OA | PStudio EPE↓ | Avg P | Avg M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **TrackEverything (Ours)** | Pi3 | 39.3 | **88.9** | 28.5 | 88.8 | 0.16 | 27.7 | 39.8 | 19.4 | **84.6** | 2.04 | 23.5 | 83.6 | 15.1 | 88.4 | 0.16 | 30.2 | 70.8 |
| **TrackEverything (Ours)** | Ω | **45.7** | 88.7 | **34.4** | **90.2** | **0.15** | 28.2 | 40.3 | 20.1 | 83.4 | 2.26 | 30.1 | **88.7** | **20.2** | **88.9** | **0.11** | **34.7** | 72.6 |
| *All-frame dense tracking* | | | | | | | | | | | | | | | | | | |
| D4RT | – | 40.8 | – | 30.7 | 92.6 | – | 41.0 | – | 30.4 | 87.5 | – | 49.6 | – | 37.2 | 89.7 | – | 43.8 | – |
| VDPM | – | 4.8 | 65.0 | 2.7 | 86.3 | 0.34 | 14.7 | 23.3 | 10.4 | **85.3** | 2.86 | 13.1 | 82.1 | 8.6 | 82.2 | 0.15 | 10.9 | 56.8 |
| **TrackEverything (Ours)** | Pi3 | 39.3 | **88.9** | 28.5 | 88.8 | 0.16 | 27.7 | 39.8 | 19.4 | 84.6 | **2.04** | 23.5 | 83.6 | 15.1 | 88.4 | 0.16 | 30.2 | 70.8 |
| **TrackEverything (Ours)** | Ω | **45.7** | 88.7 | **34.4** | **90.2** | **0.15** | **28.2** | **40.3** | **20.1** | 83.4 | 2.26 | **30.1** | **88.7** | **20.2** | **88.9** | **0.11** | **34.7** | **72.6** |

Table 6: **TAPVid-3D detailed metrics, full-length.** Bold / underline: best / second-best within the block (D4RT excluded). N/A: method cannot run on full-length video. Column definitions match Table 5.

| Method | Geo | ADT P | ADT M | ADT AJ | ADT OA | ADT EPE↓ | DriveTrack P | DriveTrack M | DriveTrack AJ | DriveTrack OA | DriveTrack EPE↓ | PStudio P | PStudio M | PStudio AJ | PStudio OA | PStudio EPE↓ | Avg P | Avg M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Sparse tracking* | | | | | | | | | | | | | | | | | | |
| CoTracker3 | Pi3 | 31.4 | 82.3 | 18.9 | 84.8 | 0.20 | 24.3 | 40.5 | 17.0 | 89.7 | 2.30 | 18.7 | 80.2 | 10.6 | 81.7 | 0.20 | 24.8 | 67.7 |
| CoTracker3 | Ω | 34.5 | 84.4 | 21.5 | 84.8 | **0.18** | 27.4 | 43.9 | 19.8 | 89.7 | 2.21 | 27.1 | 87.9 | 16.5 | 81.7 | 0.13 | 29.7 | **72.1** |
| SpatialTracker-v2 | – | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | 23.9 | 84.0 | 15.5 | 80.5 | 0.15 | – | – |
| SpatialTracker-v2 | Pi3 | 32.1 | 82.4 | – | – | – | 26.2 | 43.5 | 18.7 | **91.0** | **1.46** | 17.2 | 78.5 | 10.3 | 82.3 | 0.22 | 25.2 | 68.1 |
| SpatialTracker-v2 | Ω | 36.6 | 84.6 | 23.5 | 85.1 | 0.19 | **28.1** | **45.1** | **20.2** | 90.7 | 1.58 | 23.7 | 84.6 | 15.0 | 80.9 | 0.17 | 29.5 | 71.4 |
| TAPIP-3D | Pi3 | 32.7 | 82.0 | 21.9 | 89.3 | 0.21 | 25.0 | 40.9 | 17.7 | 88.0 | 2.01 | 19.1 | 80.7 | 11.0 | 82.9 | 0.19 | 25.6 | 67.9 |
| TAPIP-3D | Ω | 37.2 | 85.0 | 25.6 | **89.4** | **0.18** | 24.5 | 40.3 | 17.6 | 88.3 | 2.17 | **27.6** | 88.2 | 17.2 | 83.1 | **0.12** | 29.8 | 71.2 |
| **TrackEverything (Ours)** | Pi3 | 31.0 | 81.9 | 21.1 | 87.9 | 0.21 | 24.7 | 40.7 | 17.3 | 84.2 | 2.04 | 18.9 | 81.0 | 11.4 | 86.0 | 0.18 | 24.9 | 67.9 |
| **TrackEverything (Ours)** | Ω | **40.0** | **85.1** | **29.1** | 88.9 | 0.19 | 26.5 | 42.4 | 18.9 | 83.1 | 2.03 | 27.2 | **88.5** | **17.7** | **87.1** | **0.12** | **31.2** | 72.0 |
| *First-frame dense tracking* | | | | | | | | | | | | | | | | | | |
| ST4RTrack | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Any4D | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Any4D | Pi3 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Any4D | Ω | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| SceneTracker | UD | 23.1 | – | – | – | – | 6.8 | – | – | – | – | 12.7 | – | – | – | – | 14.2 | – |
| DeltaV2 | UD | 24.7 | – | – | – | – | 23.8 | – | – | – | – | 14.4 | – | – | – | – | 21.0 | – |
| DeltaV2 | Pi3 | 30.9 | 78.5 | 21.8 | 88.1 | 0.40 | 26.8 | 43.7 | 18.6 | 82.2 | **1.62** | 19.8 | 80.8 | 11.8 | 82.0 | 0.19 | 25.8 | 67.7 |
| DeltaV2 | Ω | 34.7 | 79.9 | 25.1 | 87.6 | 0.37 | **27.3** | **44.2** | **19.5** | 82.4 | 1.73 | **27.7** | 87.9 | 17.6 | 83.5 | 0.13 | 29.9 | 70.7 |
| **TrackEverything (Ours)** | Pi3 | 31.0 | 81.9 | 21.1 | 87.9 | 0.21 | 24.7 | 40.7 | 17.3 | **84.2** | 2.04 | 18.9 | 81.0 | 11.4 | 86.0 | 0.18 | 24.9 | 67.9 |
| **TrackEverything (Ours)** | Ω | **40.0** | **85.1** | **29.1** | **88.9** | **0.19** | 26.5 | 42.4 | 18.9 | 83.1 | 2.03 | 27.2 | **88.5** | **17.7** | **87.1** | **0.12** | **31.2** | **72.0** |
| *All-frame dense tracking* | | | | | | | | | | | | | | | | | | |
| D4RT | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| VDPM | – | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **TrackEverything (Ours)** | Pi3 | 31.0 | 81.9 | 21.1 | 87.9 | 0.21 | 24.7 | 40.7 | 17.3 | **84.2** | 2.04 | 18.9 | 81.0 | 11.4 | 86.0 | 0.18 | 24.9 | 67.9 |
| **TrackEverything (Ours)** | Ω | **40.0** | **85.1** | **29.1** | **88.9** | **0.19** | **26.5** | **42.4** | **18.9** | 83.1 | **2.03** | **27.2** | **88.5** | **17.7** | **87.1** | **0.12** | **31.2** | **72.0** |

### A.9 Complexity Analysis

We repeat the complexity analysis of Figure 3 in the main paper using the APD-P metric instead of APD-M; results are shown in Figure 10. All conclusions remain similar.

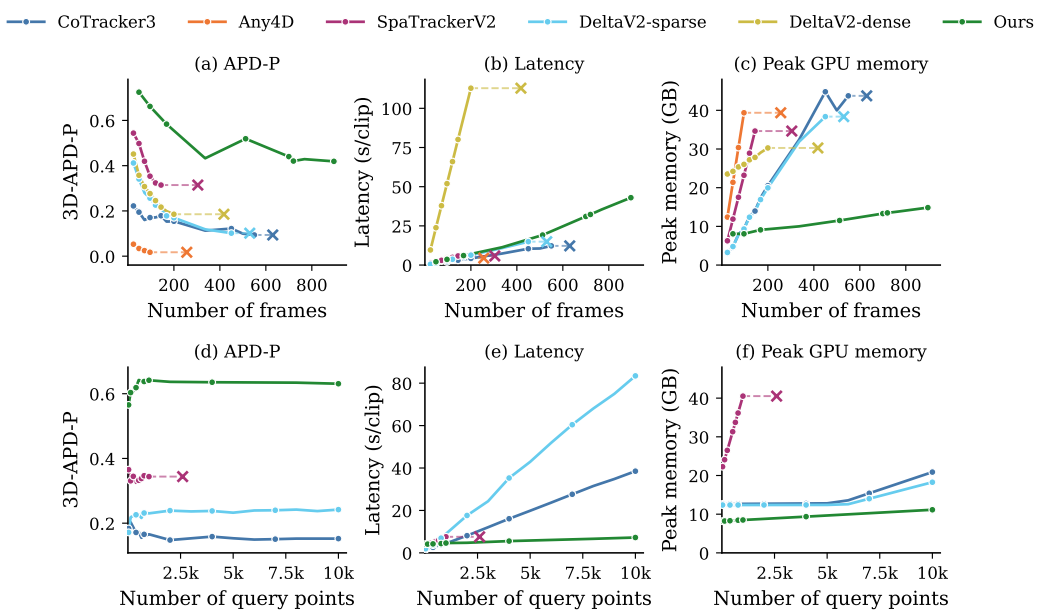


Figure 10: **Complexity analysis on PointOdyssey (APD-P).** Accuracy, latency, and peak GPU memory as functions of (top row, a–c) video length at 384 query points, and (bottom row, d–f) number of query points at 120 frames. 'x' marks out-of-memory.

### A.10 Rebuttal for ICLR's AI Reviewing tool

We utilized the preliminary AI review facility provided by ICLR. The majority of constructive feedback was incorporated directly into the main paper; below, we clarify two remaining concerns:

**"Comparison with additional baselines":** The automated review suggested comparisons against *TAPIR-3D*, *SceneTracker*, and *DynOMo*. We do not include these baselines for the following reasons:

- **TAPIR-3D:** To our knowledge, "TAPIR-3D" does not exist in the literature; The actual TAPIR (Doersch et al., 2023) model is an early 2D point tracker that is substantially superseded by modern baselines such as CoTracker3 (Karaev et al., 2025), which we already evaluate.
- **SceneTracker (Wang et al., 2025):** Under identical geometry estimation (UniDepth), SceneTracker trails DeltaV2 (Ngo et al., 2025) by a significant margin (6.8 APD-P on TAPVid-3D). Because we already benchmark DeltaV2, evaluating SceneTracker would offer no additional insight over the stronger, modern baseline. We nevertheless include SceneTracker in the supplementary material for completeness.
- **DynOMo (Seidenschwarz et al., 2025):** DynOMo is an optimization-based method requiring $\sim$50 minutes per video at test time, while achieving tracking accuracy comparable only to early feedforward trackers (e.g., original CoTracker), which CoTracker3 (and TrackEverything) decisively outperform.

We therefore focus our benchmarking on the strongest representative methods across sparse, first-frame dense, and all-frame dense paradigms, standardizing all compatible models on VGGT-$\Omega$ as the geometry estimation backbone for a fair, state-of-the-art comparison.

**"End-to-end video tracker claim":** The review noted that TrackEverything is presented as an "end-to-end video tracker" despite relying on 3D geometry estimators. In fact, the term "end-to-end" does not appear anywhere in our manuscript. Like several contemporary 3D trackers (e.g., DeltaV2), TrackEverything explicitly takes estimated 3D pointmaps as input and clearly describes this dependency.