RELIFT: DENSE 3D TRACKING AS TRAJECTORY REPAIR

Anonymous authors
Paper under double-blind review

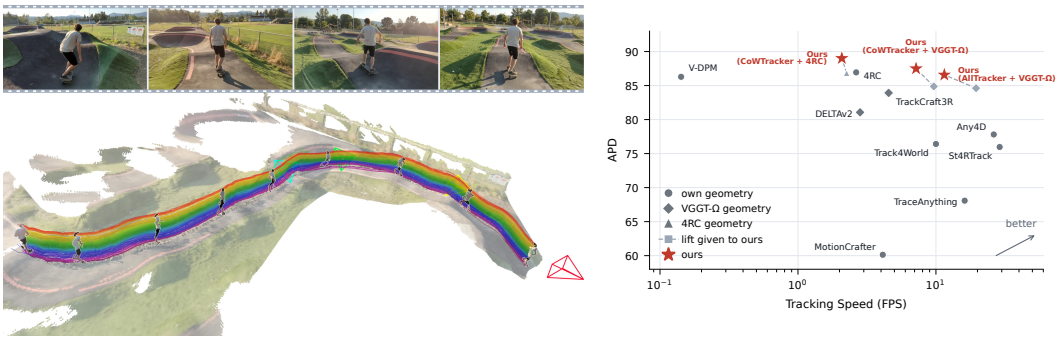


Figure 1: **Left: 3D motion from monocular video.** Selected input frames and our predicted trajectories of the skateboarder, visualized in a shared 3D scene. **Right: accuracy against throughput.** APD on all points of WorldTrack (Table 1) against the frames per second of the complete pipeline, every model a method needs included, for 84 frames at 512^2 on one RTX PRO 6000; the rows that read Depth Anything 3 are omitted. Dashed segments join each lift to its corrected result. Videos: `index.html` in the supplementary zip file.

ABSTRACT

Pretrained 2D trackers and geometry models already provide strong estimates for monocular 3D tracking where their observations are reliable. Lifting 2D tracks with estimated depth and cameras yields trajectories that are accurate in most places but jitter along the viewing ray, jump to other surfaces near occlusions, and go missing while points are hidden. We recast dense 3D tracking as repairing this lift. A feed-forward corrector reads only the geometry of the lifted trajectories and predicts their corrections. Since it never sees an image, we train it only on clean PointOdyssey trajectories with simulated lifting errors, and one trained model serves every tracker and geometry source we evaluate. On WorldTrack, it achieves state-of-the-art mean accuracy, and with VGGT-Ω geometry it is competitive with the strongest 3D trackers at a fraction of their runtime.

1 INTRODUCTION

Dense 3D point trajectories describe how scene content moves through space over time, and are key in downstream tasks such as manipulation policy learning, dynamic scene reconstruction, and motion-controlled video generation (Chen et al., 2025; Bharadhwaj et al., 2024; Lu et al., 2024; Geng et al., 2025). Recovering them from monocular video is usually posed as a single learning problem: one feed-forward network is trained end to end to output 3D tracks.

Such a network has to learn three things at once: finding correspondence across frames, recovering the geometry of the scene, and inferring motion behind occlusions. However, dedicated 2D trackers already establish strong correspondences: native CoWTracker exceeds the projected trajectories of every evaluated 3D method on dynamic points in each dataset (Appendix B.8), and even before learned correction, lifting such tracks with an off-the-shelf depth model and initializing gaps can match or exceed the mean accuracy of learned 3D trackers using the same geometry (Tables 1 and 2). Geometry is a separate matter. While dedicated 3D trackers learn internal geometry, monocular

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.

depth estimators benefit from massive, scalable training regimes that do not require dynamic 4D supervision (Lin et al., 2025; Wang et al., 2026). Tying motion recovery to a fixed geometry encoder prevents a tracker from adopting these rapidly advancing depth models.

This suggests a different division of labour, which we realise in a method we call ReLift. We take the viewing ray of each point from a 2D tracker and its range along that ray from a depth model, and lift the two into a 3D trajectory we call the lift. Where the tracker reports an occlusion we discard its prediction and interpolate, since a track behind an occluder samples the depth of the occluder rather than of the point. The result is a trajectory with strong observations where tracking and depth are reliable, but characteristic failures elsewhere: gaps under occlusion, jumps where the sampled depth crosses a boundary, and temporal jitter as per-frame depth fluctuates. What is left to learn is not how to track. It is how to repair. We call the network that does it the corrector.

A natural way to train a corrector is to lift trajectories from synthetic video, which seems ideal for minimizing train-test mismatch. But that ties the training inputs to the errors of the chosen estimators. We test this choice in Sec. 4.4: under a matched training budget, a corrector trained on DA3-generated lifts is less accurate than one trained on simulated lifts on every dataset, including when evaluated with DA3.

We avoid this by training without ever running a tracker, a depth model, or a renderer. We model the geometric effects of lifting failures directly as corruptions of clean 3D trajectories from PointOdyssey (Zheng et al., 2023), and supervise a network to undo them. Training uses geometric annotations without images or provider predictions, concentrating supervision on learning motion priors for trajectory repair and completion (Sec. 3.4).

A single trained ReLift model serves two 2D trackers, AllTracker (Harley et al., 2025) and CoWTracker (Lai et al., 2026), and three geometry sources, Depth Anything 3 (Lin et al., 2025), VGGT-Ω (Wang et al., 2026), and the geometry that 4RC (Luo et al., 2026) estimates for itself, without retraining for any of them. Correction improves all six lifts (Fig. 3), supporting transfer across the evaluated providers. The last pairing directly tests the decomposition: given 4RC's own geometry, the corrected lift of an off-the-shelf 2D tracker improves on 4RC's mean tracking accuracy (Table 1).

Our contributions are:

- We introduce ReLift, which recasts monocular 3D tracking as repairing a lifted trajectory. By decoupling viewing rays (from 2D tracking) and range (from modular depth), we concentrate learning on integrating observations and completing motion, with pretrained models supplying correspondence and geometry (Sec. 3).
- We train this corrector without running a tracker or depth model, by corrupting clean synthetic trajectories in ray direction, range and availability, so one set of weights serves every tracker and geometry source we evaluate (Fig. 3).
- We achieve state-of-the-art mean dense 3D tracking accuracy on WorldTrack, and show that the decomposition is modular: swapping in a better geometry provider improves accuracy without retraining, and given 4RC's own geometry, our CoWTracker variant improves on 4RC's mean accuracy by 2.1 (All) and 5.4 (Dyn) points (Table 1). With VGGT-Ω geometry, we are competitive with 4RC at 2.7–4.4$\times$ lower runtime.

The appendix details the method, training and evaluation (Appendix A) and reports additional results, including Kubric, image-plane accuracy and runtime (Appendix B).

2 Related Work

Dynamic 3D reconstruction from monocular video. Classical non-rigid structure-from-motion methods recover time-varying 3D structure from image correspondences using local rigidity and low-rank constraints to reduce monocular ambiguity (Bregler et al., 2000; Akhter et al., 2008; Taylor et al., 2010; Dai et al., 2014). Modern per-video optimization methods combine image evidence with depth, tracking, and deformation priors (Zhang et al., 2022; Liu et al., 2023). Shape of Motion (Wang et al., 2025b) and MoSca (Lei et al., 2024) combine pretrained depth and tracking cues with structured motion representations for reconstruction and rendering. Uni4D (Yao et al., 2025) and TrackingWorld (Lu et al., 2025) optimize camera poses and dense 3D trajectories from

pretrained depth, tracking and segmentation models, or from densified 2D tracks. These methods demonstrate the strength of combining 2D motion with 3D geometry, but require lengthy per-scene optimization; we pursue this combination with a feed-forward model.

Feed-forward 3D geometry models. DUSt3R and MASt3R predict geometry and correspondences from image pairs (Wang et al., 2024; Leroy et al., 2024), while VGGT, VGGT-Ω, and Depth Anything 3 extend geometry estimation to multiple views (Wang et al., 2025a; 2026; Lin et al., 2025). MonST3R adapts point-map prediction to dynamic scenes and aligns predictions across a video (Zhang et al., 2025b); St4RTrack and Dynamic Point Maps also represent motion and correspondences (Feng et al., 2025; Sucar et al., 2025), with V-DPM extending pairwise dynamic point maps to video (Sucar et al., 2026). Geometry and tracking capabilities thus overlap. In our pipeline, frozen geometry models supply depth and cameras, while 2D trackers establish point identities across frames.

2D tracking. Long-range 2D correspondences offer another route to recovering 3D trajectories. CoTracker jointly tracks multiple points (Karaev et al., 2024), while AllTracker and CoWTracker provide dense correspondences for every reference-frame pixel (Harley et al., 2025; Lai et al., 2026). DINO-Tracker refines pretrained image features through per-video self-supervision, retaining strong tracking accuracy even without optical-flow supervision (Tumanyan et al., 2024). TracksTo4D learns a feed-forward encoder for low-rank 3D reconstruction from 2D tracks without 3D supervision (Kasten et al., 2024). We use AllTracker or CoWTracker to initialize depth-lifted 3D trajectories for learned repair. Appendix B.8 compares native and projected 2D accuracy, supporting the strength of these correspondence inputs.

Dense feed-forward 3D tracking. Training a 3D tracker serves two central goals: amortizing inference to avoid lengthy per-scene optimization, and learning motion priors that resolve ambiguous observations. Such priors are particularly important during occlusion, when depth at a tracked image location describes the occluder rather than the hidden point. DELTA and DELTAv2 predict dense long-range 3D tracks using video and depth inputs (Ngo et al., 2025b;a), while SpatialTrackerV2 jointly estimates geometry, camera motion, and point motion (Xiao et al., 2025). Trace Anything, Track4World, and 4RC predict dense trajectories or queryable geometry and motion (Liu et al., 2025; Lu et al., 2026; Luo et al., 2026). TrackCraft3R repurposes pretrained video-diffusion models for dense 3D tracking (Nam et al., 2026); MotionCrafter uses video generative priors for geometry and scene flow (Zhu et al., 2026). Learning transferable motion priors remains constrained by supervision: PointOdyssey (Zheng et al., 2023) supplies complete synthetic trajectories, while Stereo4D (Jin et al., 2025) expands real-world supervision through reconstructed motion at valid visible observations. We concentrate this supervision on repairing corrupted and completing missing 3D motion, using frozen correspondence and geometry estimates as observations. Our corrector trains on synthetically corrupted PointOdyssey trajectories without provider-generated inputs. Matched-geometry comparisons and gains over initialized lifts support this decomposition (Section 4.2).

3 METHOD

Given a monocular video, our goal is to recover dense 3D trajectories in the reference camera's coordinate system. Frozen pretrained models supply 2D tracks, depth and cameras, which we combine into lifted trajectories (Sec. 3.1). The lift is accurate where tracking and depth are reliable, but it jitters, has gaps where the point is hidden, and near occlusions and depth boundaries some of its available observations carry the range of another surface. The corrector must therefore both complete missing motion and repair observations that are available but wrong. Both are visible without images: gaps are marked by availability, and a wrong-surface range moves a point abruptly along its viewing ray, in a way that disagrees with the track's own history or with other tracks on the same surface. We therefore correct the lift from geometry alone. The corrector reads each track's positions, rays and availability (Sec. 3.2) and predicts coordinate residuals with attention along and across tracks (Sec. 3.3). Because it never sees an image, we train it on clean trajectories with simulated lifting errors, without running any tracker or geometry model (Sec. 3.4), and use one set of weights across all evaluated provider combinations. Temporal chunking and batched groups make correction tractable over all reference pixels (Fig. 2).

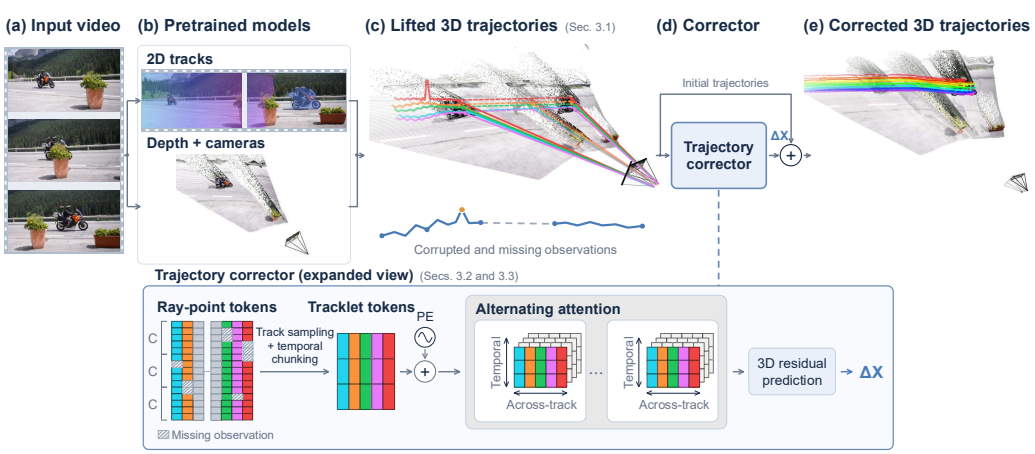


Figure 2: **Inference pipeline of ReLift.** Frozen pretrained models lift video observations (a–b) into corrupted and incomplete 3D trajectories (c). The corrector adds predicted residuals to initialized XYZ coordinates to produce the final trajectories (d–e). Below: groups of tracks are represented by ray–point tokens $\boldsymbol{\phi}_{ti}$, concatenated into tracklet tokens $\mathbf{h}_{ni}$, and processed by alternating temporal and across-track attention. The output head restores frame-level resolution and predicts XYZ residuals $\boldsymbol{\Delta}_{ti}$; the skip connection carries initialized coordinates $\bar{\mathbf{x}}_{ti}$. The corrector is trained separately on simulated lifts constructed from clean trajectory annotations (Sec. 3.4).

3.1 Lifting Tracks into 3D

For a video $\{I_t\}_{t=0}^{T-1}$, a dense 2D tracker provides image trajectories $\mathbf{u}_{ti}$ and visibility scores $v_{ti} \in [0,1]$ for pixels i of the reference frame I_0. A geometry model provides optical-axis depth maps Z_t, intrinsics K_t and camera poses. We express geometry in the reference camera coordinate system: $R_t \in \mathrm{SO}(3)$ rotates from camera t to camera 0, and $\mathbf{o}_t$ is camera t's centre in that coordinate system, with $R_0 = I$ and $\mathbf{o}_0 = \mathbf{0}$. For homogeneous image coordinates $\tilde{\mathbf{u}}_{ti} = (\mathbf{u}_{ti}, 1)$, let $\mathbf{q}_{ti} = K_t^{-1}\tilde{\mathbf{u}}_{ti}$. The unit viewing direction, range and lifted position are

$$\mathbf{d}_{ti} = \frac{R_t\,\mathbf{q}_{ti}}{\|\mathbf{q}_{ti}\|}, \qquad r_{ti} = Z_t(\mathbf{u}_{ti})\,\|\mathbf{q}_{ti}\|, \qquad \mathbf{y}_{ti} = \mathbf{o}_t + r_{ti}\,\mathbf{d}_{ti}. \tag{1}$$

Since the third coordinate of $\mathbf{q}_{ti}$ is 1, its norm converts optical-axis depth into distance along the ray.

The lifted observations inherit three types of failure. *Depth jitter and drift* produce inconsistent positions over time. *Wrong-surface observations* occur at depth boundaries or when a track follows another surface. *Missing observations* arise from occlusion or tracker failure. Occlusion can also produce a wrong-surface observation if the tracker continues to report a point as visible while the sampled depth belongs to its occluder.

We mark an observation as available, $m_{ti} = 1$, when v_{ti} exceeds a tracker-specific threshold γ; otherwise $m_{ti} = 0$. We set $m_{0i} = 1$ for all tracks to provide a reference observation. Unavailable lift and ray entries are masked, while track identities connect observations before and after gaps. Retaining unavailable rays during retraining lowers occluded-point ADT accuracy by about eight points with AllTracker and DA3 or VGGT-Ω (Appendix B.5). Availability indicates the presence of evidence, not its correctness. Depth sampling and visibility-threshold settings are given in Appendix A.1.

3.2 Trajectory Representation

Normalization and initialization. We divide positions and camera centres by s, the median reference depth over sampled tracks during training and over all reference pixels during inference. This removes a common geometric scale factor. To predict residuals at every frame, we initialize a complete trajectory $\bar{\mathbf{x}}_{ti}$. At available observations, $\bar{\mathbf{x}}_{ti} = \mathbf{y}_{ti}/s$; inside a gap we interpolate linearly in

time between the bounding observations, and after the last observation we hold the last observed position.

Ray–point tokenization. Let $\mathbf{a}_i = \bar{\mathbf{x}}_{0i} = \mathbf{y}_{0i}/s$ be the normalized reference anchor, and let g_{ti} be the distance in frames to the nearest available observation. For each track and frame, we construct

$$\boldsymbol{\phi}_{ti} = \big(\bar{\mathbf{x}}_{ti} - \mathbf{a}_i,\ m_{ti}\,(\mathbf{y}_{ti}/s - \mathbf{a}_i),\ m_{ti},\ g_{ti},\ m_{ti}\,\mathbf{d}_{ti},\ \mathbf{o}_t/s\big) \in \mathbb{R}^{14}. \tag{2}$$

The first entry describes the initialized motion relative to the anchor; the second provides the observed position in the same coordinates. Inside gaps, the initialization remains defined while the observed-lift and ray entries are set to zero. The mask and gap distance distinguish these estimates from observation evidence. The ray gives the direction along which a range error displaces an observation, and with the camera centre completes the viewing geometry.

We concatenate C consecutive ray–point tokens from each track into a temporal chunk, reducing the attention token count while retaining the frame-level features. Padding the final chunk with unobserved frames gives $N = \lceil T/C \rceil$ chunks per track. A learned projection forms a *tracklet token* for track i and chunk $\mathcal{T}_n$:

$$\mathbf{h}_{ni} = W_{\text{in}}\big[\boldsymbol{\phi}_{ti}\big]_{t\in\mathcal{T}_n} + \mathbf{p}_n + \boldsymbol{\psi}(\mathbf{a}_i) \in \mathbb{R}^D. \tag{3}$$

Here $W_{\text{in}} \in \mathbb{R}^{D\times 14C}$ maps the concatenated features to width D, $\mathbf{p}_n$ encodes temporal position, and $\boldsymbol{\psi}(\mathbf{a}_i)$ embeds the reference anchor. Additional details are in Appendix A.2.

3.3 Trajectory Corrector

The corrector operates on a *group* of P tracks spanning all T frames, since attention across all reference pixels at once is infeasible: during training a group is a random sample of trajectories, and at inference random groups cover the reference pixels (Sec. 3.3). Inference groups can be larger than training groups; Appendix B.6 varies their size.

For a group of P tracks, the tracklet tokens form a grid $H \in \mathbb{R}^{P\times N\times D}$, with one row per track and one column per chunk.

We process H with alternating temporal and across-track self-attention, following the factorization along and across tracks used by TracksTo4D (Kasten et al., 2024). Temporal attention connects all N chunks of each track, aggregating observations before and after gaps along the correspondences supplied by the 2D tracker. Across-track attention connects the P tracks within the same chunk. Intuitively, temporal attention lets the model distinguish a brief range spike, where the track jumps along its ray and back, from true acceleration, and across-track attention lets it draw on related motion elsewhere in the group, both to complete gaps and to expose observations that disagree with it. For fixed feature width, attention costs $O(PN^2 + NP^2)$ per block, compared with $O(P^2N^2)$ for joint attention over all tokens.

A final prediction head maps each tracklet token to C three-dimensional residuals $\boldsymbol{\Delta}_{ti}$, restoring the original frame-level resolution. The output is

$$\hat{\mathbf{x}}_{ti} = s\,\big(\bar{\mathbf{x}}_{ti} + \boldsymbol{\Delta}_{ti}\big). \tag{4}$$

Rays condition these predictions without constraining corrections to lie along them. The corrector changes positions only; where evaluation requires visibility, we return the 2D tracker's scores v_{ti} unchanged.

Efficient dense inference. We randomly partition the reference pixels into groups of P tracks, correct the groups independently in batches with the clip's shared scale s, and write each prediction back to its pixel. For fixed sequence length and group size, the cost grows linearly with the number of pixels (Appendices B.7 and B.6).

3.4 Training

We train the corrector on clean 3D trajectories from the PointOdyssey training split (Zheng et al., 2023). Each sample is a group of clean trajectories $\mathbf{x}_{ti}$ over T frames, ground-truth visibility $v^\star_{ti} \in \{0, 1\}$, and camera centres $\mathbf{o}_t$ in reference camera coordinates. We generate simulated lifts directly from these annotations, without images or provider inference, so lifting errors can be varied independently of any tracker or geometry model.

Simulated lifting errors. We model the failures in Section 3.1 through perturbations to viewing direction, range and observation availability. From each clean point's range $r^\star_{ti}$ and unit ray $\mathbf{d}^\star_{ti}$ from $\mathbf{o}_t$, we construct

$$\mathbf{d}_{ti} = Q_{ti}\mathbf{d}^\star_{ti}, \qquad r_{ti} = r^\star_{ti}\exp(\varepsilon_{ti}), \qquad \mathbf{y}_{ti} = \mathbf{o}_t + r_{ti}\,\mathbf{d}_{ti}. \tag{5}$$

The rotation $Q_{ti} \in \mathrm{SO}(3)$ applies a small angular perturbation to approximate *correspondence errors*. The log-range perturbation ε_{ti} combines dense noise and temporally correlated offsets to model *depth jitter and drift*, with sparse large offsets to model *wrong-surface observations*. Its components and sampling distributions are specified in Appendix A.3. Camera centres remain unperturbed; these corruptions do not explicitly simulate camera-pose errors or persistent switches to another object's trajectory.

To simulate *missing observations*, we initialize m_{ti} from ground-truth visibility $v^\star_{ti}$, then remove observations with synthetic occluders and contiguous tracker dropouts. Near the start or end of a synthetic occlusion, we sometimes keep the hidden point as visible but give it the occluder's depth, as a tracker does when it follows a point into occlusion. The corrector thus sees observations that are available but wrong. Clean targets supervise every frame, including those with $m_{ti} = 0$. The simulated observations undergo the normalization, initialization and tokenization of Section 3.2, with scale and anchors computed from the corrupted input. Section 4.4 ablates these corruption families; removing wrong-surface corruptions produces the largest accuracy drop (Table 2).

Training for transfer. We randomize corruption parameters across samples, drawing on domain randomization in robotics (Tobin et al., 2017; Peng et al., 2018). Here, randomization acts on geometric observations: the simulation reproduces the geometric signatures of lifting failures at varied magnitudes, rather than fitting the error statistics of one provider. A single trained model improves the lift of every evaluated tracker–geometry combination (Fig. 3). In a separate comparison with matched training steps, simulated-lift training outperforms training on DA3-generated lifts on every dataset, including with DA3 itself (Fig. 5).

Reconstruction objective. We supervise 3D positions and frame-to-frame displacements, including where input observations are unavailable, using

$$\mathcal{L} = \frac{\sum_{t,i} w_{ti}\|\mathbf{e}_{ti}\|^2}{\sum_{t,i} w_{ti}} + \lambda\,\frac{\sum_{t\geq 1,i} w_{ti}\|\mathbf{e}_{ti} - \mathbf{e}_{t-1,i}\|^2}{\sum_{t\geq 1,i} w_{ti}}. \tag{6}$$

Here $\mathbf{e}_{ti} = (\hat{\mathbf{x}}_{ti} - \mathbf{x}_{ti})/(s\rho_i)$ is the relative position error, with $\rho_i = \max(\|\mathbf{a}_i\|, \rho_{\min})$ the floored reference range in normalized coordinates. This scaling prevents distant points from dominating the objective. The first term supervises positions; the second, weighted by λ, matches predicted and true displacements between consecutive sampled frames, discouraging jitter while preserving the target motion. The weights w_{ti} count moving tracks four times as much as static ones, and unavailable observations twice as much as available ones (Appendix A.3).

4 Experiments

4.1 Experimental Setup

Evaluation Datasets. We evaluate dense 3D point tracking in world coordinates on the four WorldTrack (Feng et al., 2025) datasets: the real-world Aria Digital Twin (ADT) (Pan et al., 2023) and Panoptic Studio (PS) (Joo et al., 2019) from TAPVid-3D (Koppula et al., 2024), and the synthetic PointOdyssey (PO) (Zheng et al., 2023) and Dynamic Replica (DR) (Karaev et al., 2023). We score dense predictions at the sparse ground-truth trajectories of points visible in the reference frame, over the first 84 frames. We also evaluate on TrackCraft3R's (Nam et al., 2026) 50 24-frame Kubric (Greff et al., 2022) clips, with ground-truth trajectories for every reference pixel (Appendix B.2).

Metrics. We report the Average Percentage of 3D Points within Delta (APD3D↑) (Koppula et al., 2024), averaged over distance thresholds $\delta \in \{0.1, 0.3, 0.5, 1.0\}$ m. Following 4RC (Luo et al., 2026), we evaluate all points (*All*) and dynamic points (*Dyn*) after RANSAC Sim(3) alignment, fitted separately to each scoring set. We also evaluate image-plane accuracy using APD2D (Appendix B.8). Appendix B provides scoring details, complementary metrics (AJ, OA and EPE), and global-scale alignment results.

Table 1: **Dense 3D tracking on WorldTrack.** All six of our rows use one trained corrector, paired with two 2D trackers and three geometry sources without retraining. Every pairing outperforms, in mean accuracy, the baselines that use the same geometry, and our best pairing leads mean accuracy on all and dynamic points. APD3D (%) after RANSAC Sim(3) alignment; bold best, underline second best. Time: complete pipeline. †At 256^2 instead of 512^2. ‡Scored only on the 63.6% of ADT queries inside MotionCrafter's crop; on all queries: 60.98 (All), 78.51 (Dyn) (App. B.9).

	ADT		PS		PO		DR		Mean		
Method	All	Dyn	All	Dyn	All	Dyn	All	Dyn	All	Dyn	Time (s)
4RC	87.07	78.57	86.25	86.10	**85.92**	81.17	88.51	80.56	86.94	81.60	31.8
Any4D	76.17	78.32	77.57	77.61	73.96	73.56	83.54	72.83	77.81	75.58	3.2
MotionCrafter	67.04‡	83.34‡	62.60	62.63	46.29	49.09	70.83	62.39	61.69	64.36	20.4
OpenD4RT	71.71	79.68	84.36	85.27	68.39	74.21	77.55	79.69	75.50	79.71	284†
St4RTrack	81.39	79.39	72.84	72.24	71.36	71.44	78.29	67.23	75.97	72.58	2.9
Track4World	77.48	91.80	80.10	79.39	69.13	69.43	78.94	71.98	76.41	78.15	8.4
TraceAnything	81.76	77.01	71.88	71.66	49.77	53.14	68.91	61.80	68.08	65.90	5.2
V-DPM	88.42	79.40	89.26	89.49	84.46	**85.95**	82.99	80.80	86.28	83.91	591
DELTAv2 + DA3	83.63	92.19	85.38	85.37	66.32	67.35	76.88	64.50	78.06	77.35	33.4
DELTAv2 + VGGT-Ω	84.57	91.67	89.41	89.22	67.49	68.60	82.79	72.23	81.06	80.43	29.8
TrackCraft3R + DA3	91.75	90.82	84.39	83.95	74.04	71.42	78.44	65.84	82.16	78.01	22.1
TrackCraft3R + VGGT-Ω	92.56	89.54	86.56	86.65	72.66	71.48	83.89	73.74	83.92	80.36	18.5
Ours, one trained corrector for all rows (2D tracker + geometry provider)											
AllTracker + DA3	93.69	**94.98**	85.46	85.59	76.04	77.44	79.50	68.94	83.68	81.74	11.4
AllTracker + VGGT-Ω	94.04	92.17	90.40	90.46	76.10	78.96	85.74	78.10	86.57	84.92	7.3
AllTracker + 4RC	91.07	94.13	88.73	88.65	83.63	80.26	88.81	81.08	88.06	86.03	36.4
CoWTracker + DA3	93.78	94.85	86.99	86.64	77.63	79.19	79.72	68.19	84.53	82.22	15.3
CoWTracker + VGGT-Ω	**94.14**	92.09	**92.20**	**92.06**	77.63	80.49	86.00	78.70	87.49	85.84	11.7
CoWTracker + 4RC	91.41	94.31	89.97	89.83	85.60	82.14	**89.05**	**81.49**	**89.01**	**86.95**	40.4

Methods in comparison. We pair AllTracker (Harley et al., 2025) or CoWTracker (Lai et al., 2026) with geometry from Depth Anything 3 (DA3) (Lin et al., 2025), VGGT-Ω (Wang et al., 2026) or 4RC, and compare with 4RC, V-DPM (Sucar et al., 2026), St4RTrack (Feng et al., 2025), Any4D (Karhade et al., 2025), Track4World (Lu et al., 2026), TraceAnything (Liu et al., 2025), MotionCrafter (Zhu et al., 2026), OpenD4RT (RHOS Team, 2026) (an open reimplementation of D4RT (Zhang et al., 2025a)), TrackCraft3R (Nam et al., 2026) and DELTAv2 (Ngo et al., 2025a). TrackCraft3R and DELTAv2 receive the same estimated depth, cameras and intrinsics as ours, from DA3 or VGGT-Ω; no method receives ground-truth intrinsics (Appendices A and B.9).

4.2 Quantitative Results

Accuracy. Table 1 summarizes the results. All six of our configurations use the same trained corrector, so the pipeline is modular: a better geometry provider improves tracking without retraining, and for both trackers mean accuracy rises from DA3 to VGGT-Ω to 4RC geometry. Our method achieves state-of-the-art mean accuracy on both all and dynamic points, with CoWTracker and 4RC geometry. It is particularly strong on the real-world datasets ADT and PS, where our variants lead both metrics. Appendix B.7 analyzes runtime and memory.

Combining 2D motion and 3D geometry. To test the decomposition introduced in Section 1, we compare methods with the same geometry inputs. Our method outperforms TrackCraft3R and DELTAv2 in every APD3D column of Table 1. Given 4RC's own geometry, our CoWTracker variant improves on 4RC's mean accuracy by 2.1 (All) and 5.4 (Dyn) points, isolating the benefit of 2D correspondence and repair over joint learning. On Kubric (Appendix B.2), where every method using DA3 or VGGT-Ω geometry has several times 4RC's end-point error, our variants with that geometry trail 4RC but lead the other methods that share it on all points; with 4RC geometry, ours outperform 4RC on dynamic points. Even before correction, the lift matches or exceeds TrackCraft3R and DELTAv2 given the same geometry (Table 2, first row); correction then adds 1.6 to 3.6 points, lifting CoWTracker with VGGT-Ω past 4RC and V-DPM.

Image-plane accuracy. We project 3D trajectories through their associated cameras (Table 12; Appendix B.8). Switching 2D trackers changes our mean APD2D by about two points, versus at

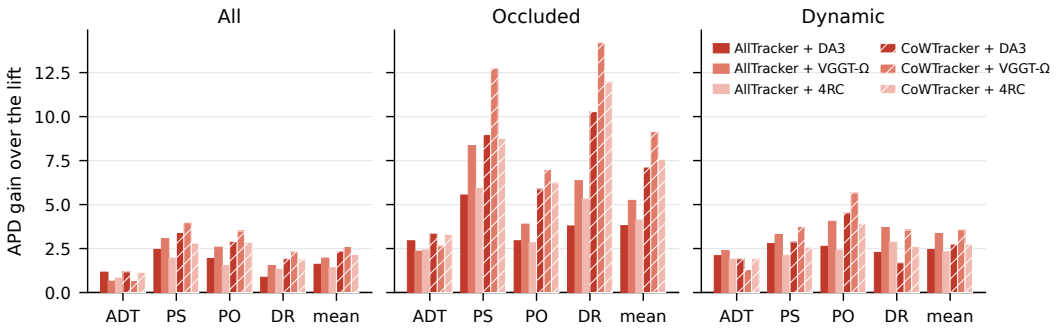


Figure 3: **Gains from learned motion integration.** APD3D improvement over lifted trajectories with interpolated gaps and held terminal positions (Sec. 3.2), per dataset and averaged, for all tracker–geometry combinations; hatching denotes CoWTracker. *Occluded* restricts *All* to ground-truth-occluded point–frames, retaining its alignment. Ground-truth occlusion differs from predicted input availability.

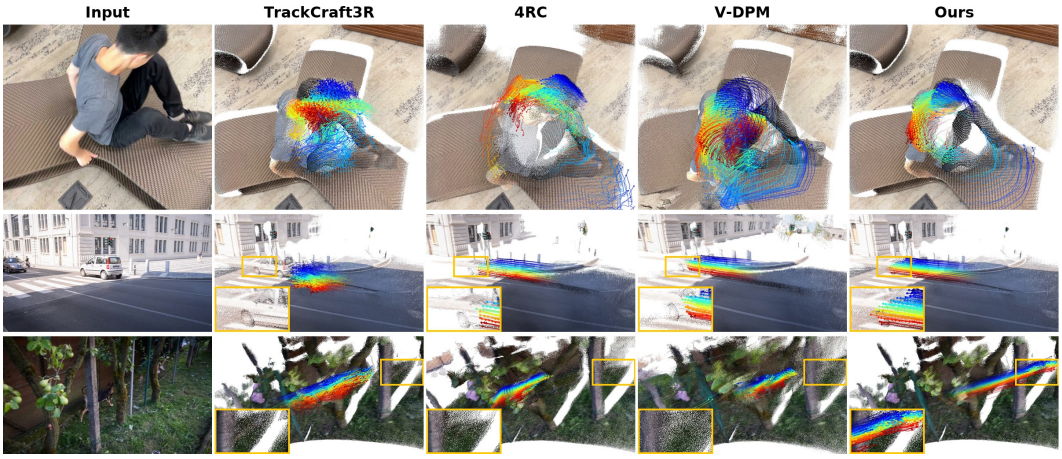


Figure 4: **Qualitative comparison.** Top to bottom: DyCheck *spin* (Gao et al., 2022) and DAVIS *car-shadow* and *libby* (Perazzi et al., 2016). Methods track the same reference pixels from matched novel viewpoints. Ours uses AllTracker and VGGT-Ω; TrackCraft3R shares this geometry, while 4RC and V-DPM use their own reconstructions. Yellow boxes mark magnified regions

most 0.6 across geometry providers. Track4World leads our best configuration by 0.6 on dynamic-point APD2D but trails by 8.8 on dynamic-point APD3D. APD2D cannot capture errors along a viewing ray, which leave projections unchanged. Our CoWTracker with 4RC geometry remains below native CoWTracker by 1.5 (All) and 3.8 (Dyn) points.

Contribution of learned motion integration. Fig. 3 measures what correction adds to its own initialized lift (Sec. 3.2). Accuracy improves across datasets and all six provider combinations, with the largest gains on occluded points.

4.3 Qualitative Results

Figure 4 shows continuous arcs in *spin*, trajectories reaching the car's front through shadow-induced appearance changes in *car-shadow*, and recovery of the dog's head after occlusion in *libby*. Competing trajectories tangle, zigzag or stop short. Videos are in `index.html` in the supplementary zip file; the appendix adds qualitative comparisons, including against the naive lift with initialized gaps (Appendix B.10), illustrating the repair and completion gains measured in Figure 3.

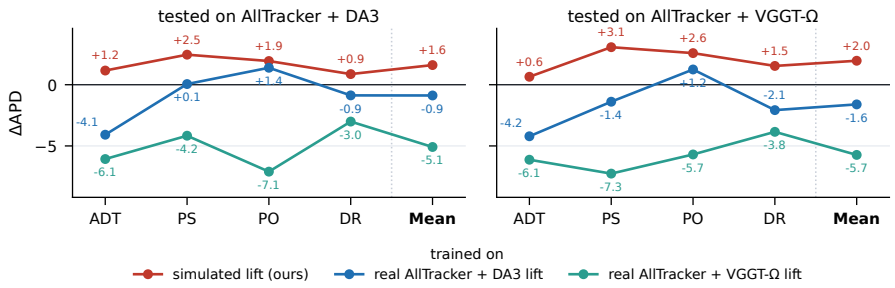


Figure 5: **Simulated versus provider-generated lifts.** APD3D gains on all points over the initialized lift, with AllTracker and DA3 (left) or VGGT-Ω (right). "Real" denotes provider-generated training lifts; training steps are matched. The VGGT-Ω-lift run is discussed in App. A.3.

Table 2: **Contribution of training corruptions.** Removing wrong-surface corruptions causes the largest drop. APD3D×100, mean of four datasets, gain over the initialized lift in parentheses. Each variant is retrained with the same data, network, loss and budget (App. A.3).

	AllTracker + DA3		AllTracker + VGGT-Ω		CoWTracker + DA3		CoWTracker + VGGT-Ω	
Training corruptions	All	Dyn	All	Dyn	All	Dyn	All	Dyn
Lift (the corrector's input)	82.1	79.3	84.6	81.6	82.2	79.5	84.9	82.3
Full simulation (ours)	83.7 (+1.6)	81.7 (+2.4)	86.6 (+2.0)	84.9 (+3.4)	84.5 (+2.4)	82.2 (+2.8)	87.5 (+2.6)	85.8 (+3.6)
without correlated noise	83.5 (+1.5)	81.3 (+2.0)	86.4 (+1.8)	84.5 (+2.9)	84.2 (+2.0)	82.1 (+2.7)	87.3 (+2.5)	85.7 (+3.5)
without wrong surface	82.3 (+0.2)	79.6 (+0.3)	84.9 (+0.3)	82.1 (+0.5)	82.8 (+0.6)	80.2 (+0.7)	85.5 (+0.6)	83.1 (+0.9)
without synthetic gaps	83.7 (+1.6)	81.5 (+2.2)	86.6 (+2.0)	84.9 (+3.4)	84.5 (+2.4)	82.3 (+2.8)	87.5 (+2.6)	85.8 (+3.6)
white noise only	82.4 (+0.3)	79.8 (+0.5)	85.0 (+0.4)	82.6 (+1.0)	82.8 (+0.6)	80.5 (+1.0)	85.5 (+0.6)	83.2 (+1.0)

4.4 Ablation Study

Simulated versus provider-generated lifts. Figure 5 compares training lifts from PointOdyssey with the same network and training budget. Simulated-lift training improves every dataset with both geometry models, and outperforms training on DA3-generated lifts on every dataset, even when tested with DA3. The run trained on VGGT-Ω-generated lifts degrades even its own in-domain lift, which suggests a problem in its training data; we draw no conclusions from it (Appendix A.3).

Contribution of corruption families. We test the simulation of Section 3.4 by removing *correlated noise* (flicker, tilt, bias and ray noise), *wrong-surface observations* (range outliers and observations retained at an occluder's depth), or *synthetic gaps* (occluders and tracker dropouts). Table 2 shows that removing wrong-surface corruptions causes the largest loss of accuracy across both trackers and geometry models, supporting their role in learning to repair incorrect observations. Correlated noise provides smaller mean improvements. Additional synthetic gaps have little effect on mean accuracy in this setup; natural PointOdyssey occlusions remain in every training variant.

Chunking and track-group size. Chunk size $C = 4$ and group size $P = 1024$ balance accuracy and cost: $C = 8$ is faster but slightly less accurate, and larger groups add little (App. B.6).

5 Conclusion

We introduced ReLift, a feed-forward approach to dense monocular 3D tracking that combines pretrained 2D correspondences and geometry with learned trajectory repair. Training on corrupted PointOdyssey trajectories, rather than on any provider's outputs, concentrates 4D supervision on repair. One corrector improves all six provider combinations, most on occluded points; with 4RC geometry it achieves state-of-the-art mean WorldTrack accuracy, and with VGGT-Ω it is competitive at substantially shorter runtime than 4RC and V-DPM. Input geometry remains a limiting factor (App. B.2). Extending repair to camera errors and persistent correspondence failures, and matching the native tracker's image-plane accuracy, are future work.

AI USE STATEMENT

We used large language models to assist with manuscript writing and editing, literature search, reference management, consistency checks between the text, tables and figures, and LaTeX editing and compilation checks. We have reviewed all AI-assisted text, and every edit was accepted or revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available datasets, under their respective licenses; some evaluation sequences contain people, and we collect no new data. Like other dense tracking methods, ours could be applied to monitoring people; we release it for research on scene understanding, robotics and video generation, and it adds no identification capability beyond its input models.

REPRODUCIBILITY STATEMENT

Section 3 describes the method, and Appendix A gives the details needed to reimplement it: observation preprocessing and visibility thresholds (App. A.1); the corrector's architecture, embeddings and dimensions (App. A.2, Table 3); training data, the corruption model with all sampling distributions, loss settings and optimization (App. A.3); and dense inference (App. A.4). Training uses only clean trajectories from the PointOdyssey training split, and the pretrained trackers and geometry models are used as released, without fine-tuning. Appendix B specifies the evaluation protocol (App. B.1), the input settings of every baseline (App. B.9), the image-plane evaluation (App. B.8), the controls of each ablation (App. A.3) and the timing setup (App. B.7). We will release our code and trained weights.

REFERENCES

Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. *Advances in Neural Information Processing Systems*, 21, 2008.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pp. 306–324, 2024. URL https://arxiv.org/abs/2405.01527.

Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, volume 2, pp. 690–696. IEEE, 2000.

Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. OmniRe: Omni urban scene reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014.

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 *Advances in Neural Information Processing Systems*, volume 35, pp. 13610–13626, 2022.

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*, 2025. URL https://st4rtrack.github.io/.

Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 2022.

Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Carl Doersch, Yusuf Aytar, Michael Rubinstein, Chen Sun, Oliver Wang, Andrew Owens, and Deqing Sun. Motion prompting: Controlling video generation with motion trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://motion-prompting.github.io/.

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 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3739–3751, 2022. doi: 10.1109/CVPR52688.2022.00373.

Adam W Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Suya You, et al. Alltracker: Efficient dense point tracking at high resolution. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5253–5262. IEEE, 2025.

Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4D: Learning how things move in 3D from internet stereo videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2412.09621.

Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic Studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):190–204, 2019. doi: 10.1109/TPAMI.2017.2782743.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13229–13239, 2023. doi: 10.1109/CVPR52729.2023.01271.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *European Conference on Computer Vision*, pp. 18–35. Springer, 2024.

Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4D: Unified feed-forward metric 4D reconstruction. *arXiv preprint arXiv:2512.10935*, 2025. URL https://arxiv.org/abs/2512.10935.

Yoni Kasten, Wuyue Lu, and Haggai Maron. Fast encoder-based 3D from casual videos via point track processing. *Advances in Neural Information Processing Systems*, 37:96150–96180, 2024.

Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, João Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. TAPVid-3D: A benchmark for tracking any point in 3D. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.05921.

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*, pp. 42571–42580, 2026. arXiv:2602.04877.

Jiahui Lei, Yijia Weng, Adam W. Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic gaussian fusion from casual videos via 4D motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024. URL https://arxiv.org/abs/2405.17421v2.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3D with MASt3R. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 71–91, 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 preprint arXiv:2511.10647*, 2025. URL https://arxiv.org/abs/2511.10647.

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

Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13–23, 2023.

Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *European Conference on Computer Vision*, pp. 349–366. Springer, 2024.

Jiahao Lu, Weitao Xiong, Jiacheng Deng, Peng Li, Tianyu Huang, Zhiyang Dou, Cheng Lin, Sai-Kit Yeung, and Yuan Liu. TrackingWorld: World-centric monocular 3D tracking of almost all pixels. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2512.08358.

Jiahao Lu, Jiayi Xu, Wenbo Hu, Ruijie Zhu, Chengfeng Zhao, Sai-Kit Yeung, Ying Shan, and Yuan Liu. Track4World: Feedforward world-centric dense 3D tracking of all pixels. *arXiv preprint arXiv:2603.02573*, 2026.

Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4RC: 4D reconstruction via conditional querying anytime and anywhere. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.10094.

Jisu Nam, Jahyeok Koo, Soowon Son, Jaewoo Jung, Honggyu An, Junhwa Hur, and Seungryong Kim. TrackCraft3R: Repurposing video diffusion transformers for dense 3D tracking. *arXiv preprint arXiv:2605.12587*, 2026. URL https://arxiv.org/abs/2605.12587.

Tuan Duc Ngo, Ashkan Mirzaei, Guocheng Qian, Hanwen Liang, Chuang Gan, Evangelos Kalogerakis, Peter Wonka, and Chaoyang Wang. DELTAv2: Accelerating dense 3D tracking. *arXiv preprint arXiv:2508.01170*, 2025a. URL https://arxiv.org/abs/2508.01170.

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. In *International Conference on Learning Representations*, 2025b. URL https://snap-research.github.io/DELTA/.

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria Digital Twin: A new benchmark dataset for egocentric 3D machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20076–20086, 2023. doi: 10.1109/ICCV51070.2023.01842.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*, 2018. doi: 10.1109/ICRA.2018.8460528. URL https://arxiv.org/abs/1710.06537.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

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

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, 2025.

Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-DPM: 4D video reconstruction with dynamic point maps. *arXiv preprint arXiv:2601.09499*, 2026. URL https://arxiv.org/abs/2601.09499.

Jonathan Taylor, Allan D. Jepson, and Kiriakos N. Kutulakos. Non-rigid structure from locally-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2761–2768, 2010.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. URL https://arxiv.org/abs/1703.06907.

Narek Tumanyan, Assaf Singer, Shai Bagon, and Tali Dekel. DINO-Tracker: Taming DINO for self-supervised point tracking in a single video. *arXiv preprint arXiv:2403.14548*, 2024.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5294–5306, 2025a.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-

\

omega. *arXiv preprint arXiv:2605.15195*, 2026.

Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025b. URL https://arxiv.org/abs/2407.13764.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme 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.

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

David Yifan Yao, Albert J. Zhai, and Shenlong Wang. Uni4D: Unifying visual foundation models for 4D modeling from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1116–1126, 2025. URL https://arxiv.org/abs/2503.21761.

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. *arXiv preprint arXiv:2512.08924*, 2025a. URL https://arxiv.org/abs/2512.08924.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A simple approach for estimating geometry in the presence of motion. In *International Conference on Learning Representations (ICLR)*, 2025b.

Zhoutong Zhang, Forrester Cole, Zhengqi Li, Michael Rubinstein, Noah Snavely, and William T. Freeman. Structure and motion from casual videos. In *European Conference on Computer Vision*, pp. 20–37. Springer, 2022.

Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2307.15055.

Ruijie Zhu, Jiahao Lu, Wenbo Hu, Xiaoguang Han, Jianfei Cai, Ying Shan, and Chuanxia Zheng. MotionCrafter: Dense geometry and motion reconstruction with a 4D VAE. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2602.08961.

A METHOD DETAILS

A.1 OBSERVATION PREPROCESSING

We use AllTracker (Harley et al., 2025) or CoWTracker (Lai et al., 2026) for 2D tracks, with depth, intrinsics and cameras from Depth Anything 3 (Lin et al., 2025), VGGT-Ω (Wang et al., 2026), or 4RC (Luo et al., 2026). One corrector checkpoint serves all evaluated combinations. Depth lookup uses the nearest pixel, while the viewing ray uses the sub-pixel track position. The observation mask in Section 3.1 uses $\gamma = 0.85$ for AllTracker and $\gamma = 0.65$ for CoWTracker, calibrated to match AllTracker's false-visible rate on 24 PointOdyssey training windows. We use visibility scores, not the trackers' confidence outputs. Depth lookup is clamped to the image border. These tracker-specific thresholds calibrate input availability; the corrector weights are shared across providers, and output visibility scores remain unchanged.

A.2 CORRECTOR IMPLEMENTATION

Input and output scaling. The scene scale s normalizes geometry as in Section 3.2. A separate numerical factor $S = 10$ multiplies the position, lift and camera-centre entries of Eq. equation 2; the gap feature is $\min(g_{ti}, 40)/40$. If $\mathbf{z}_{ti}$ denotes the raw prediction-head output, the residual used in Eq. equation 4 is $\mathbf{\Delta}_{ti} = \mathbf{z}_{ti}/S$. Computing the training loss in these scaled coordinates gives $S^2\mathcal{L}$.

Position embeddings. For an even embedding width d, define

$$\mathrm{PE}(x; d, \omega) = \left[\sin(x\,\omega^{-k/(d/2)}),\ \cos(x\,\omega^{-k/(d/2)})\right]_{k=0}^{d/2-1}. \tag{7}$$

The anchor embedding $\boldsymbol{\psi}(\mathbf{a}_i)$ concatenates $\mathrm{PE}(8a_{i,c}; 32, 10^4)$ over the three coordinates c, giving 96 features, then applies $\mathrm{Linear}(96 \to 384)$, GELU and $\mathrm{Linear}(384 \to 384)$. The temporal embedding is $\mathbf{p}_n = \mathrm{PE}(n; 384, 10^3)$, using the chunk index n.

Architecture. Each block applies pre-normalised residual updates to the token grid H:

$$\begin{aligned} H &\leftarrow H + \mathrm{Att}_{\mathrm{time}}(\mathrm{LN}\,H), \\ H &\leftarrow H + \mathrm{Att}_{\mathrm{track}}(\mathrm{LN}\,H), \\ H &\leftarrow H + \mathrm{MLP}(\mathrm{LN}\,H). \end{aligned} \tag{8}$$

There is no track-index or pixel embedding, so reordering tracks reorders the outputs correspondingly. Predictions can nevertheless depend on which other tracks are included in the group. Table 3 lists the network dimensions. A final layer normalization and a zero-initialized linear head produce $C \times 3$ residual coordinates per tracklet token, so zero initialization reproduces the initialized trajectories. The corrector has 19.1 M parameters.

Setting	Value
Frames per tracklet C	4
Feature width D	384
Factorized attention blocks	8
Attention heads	6, with width 64 each
Feed-forward hidden width / activation	1536 / GELU
Normalization	Pre-normalized residual blocks; final layer norm
Prediction head	Zero-initialized linear map to $3C$ coordinates

Table 3: Corrector architecture settings.

A.3 TRAINING DETAILS

Training data. Training uses the PointOdyssey annotations described in Section 3.4. A track is marked as moving, $\mu_i = 1$, if its true position departs from its reference position by more than 5 cm; otherwise $\mu_i = 0$. Windows are cut from the ground-truth annotations of the PointOdyssey training split: 84 frames at temporal strides 1 and 2, starting every 42 strides. A window keeps the points

that are visible, inside the image and in front of the camera in its reference frame, and is discarded if fewer than 1,024 remain. The cache targets 2,048 tracks per window, at most half with $\mu_i = 1$ unless too few static points remain. This gives 100 training scenes with 6,137 windows, and 8 validation scenes. A training sample draws 512 of the stored tracks, keeps the first $T \in \{48, 68, 84\}$ frames, and is mirrored in x with probability one half.

Corruption processes. The simulated lift in Eq. equation 5 uses the log-range perturbation

$$\varepsilon_{ti} = f_t + \mathbf{b}_t^\top \boldsymbol{\delta}_{ti} + \beta_{ti} + \eta_{ti} + j_{ti}. \tag{9}$$

We clip ε_{ti} to $[-1, 1]$ before computing the range. Here f_t is flicker shared by a frame, $\mathbf{b}_t \in \mathbb{R}^2$ is a tilt acting on $\boldsymbol{\delta}_{ti}$, the first two coordinates of the true ray in the reference camera coordinate system, β_{ti} is a slowly drifting per-track bias, and η_{ti} is white noise. The first three terms are first-order autoregressive in time, with unit stationary variance before scaling: $z_t = \kappa z_{t-1} + \sqrt{1-\kappa^2}\,\xi_t$, where $\xi_t \sim \mathcal{N}(0, 1)$, initialized from the stationary distribution. The outlier term j_{ti} moves a sparse set of cells far along their rays for one or two frames. Angular noise projects a draw from $\mathcal{N}(\mathbf{0}, \sigma^2 I_3)$ orthogonally to the true ray, adds it to that ray, and renormalizes, defining the rotation Q_{ti} in Eq. equation 5; the corrupted observation uses the unperturbed ground-truth camera centre $\mathbf{o}_t$. These perturbations approximate depth errors and small correspondence errors; they do not explicitly simulate camera-pose errors or persistent switches to another object's trajectory.

The observation mask m_{ti} starts from ground-truth visibility $v^\star_{ti}$. Up to two synthetic occluders, slowly drifting rectangles in the reference image plane, are placed in front of a sampled track, preferentially a moving one. They hide cells whose projections lie inside the rectangles and whose depths lie behind them. The image-plane coordinates are those of the reference camera, $(x/z,\ y/z)$. Tracker dropouts remove contiguous runs of frames. Cells hidden only by an added occluder and adjacent to an observed frame can instead be retained as observed at the occluder's depth, modelling a tracker following a point into occlusion. Table 4 specifies the distributions and probabilities. Corruption levels are drawn once per sample from broad ranges, set once and not fitted to any evaluation sequence. After all corruptions, $m_{0i} = 1$ is enforced, and the reference observation never takes an occluder's depth.

Table 4: Distributions of the simulated lift. $\mathcal{U}$ is uniform and every level is drawn once per sample.

Term	Distribution
Flicker f_t	std $\mathcal{U}(0, 0.03)$, autoregressive with $\kappa \sim \mathcal{U}(0.3, 0.95)$
Tilt $\mathbf{b}_t$	std $\mathcal{U}(0, 0.03)$, autoregressive with the flicker's κ
Bias β_{ti}	std $\mathcal{U}(0, 0.02)$, autoregressive with $\kappa = 0.9$
White noise η_{ti}	std $\mathcal{U}(0.002, 0.02)$
Outliers j_{ti}	rate $\mathcal{U}(0, 0.05)$ per cell, next-frame persistence probability 0.5, jump $\pm\mathcal{U}(0.1, 0.5)$
Angular ray noise	std $\mathcal{U}(0.0005, 0.005)$ rad
Occluders	up to two, with independent probabilities 0.6 and 0.2; target track drawn in proportion to $\mu_i + 10^{-3}$
placement	centred on the target at frame $t_s \sim \mathcal{U}\{\lfloor T/4 \rfloor, \ldots, T-1\}$, half-size $\mathcal{U}(0.03, 0.25)$ per axis
motion, depth	constant velocity drawn once per occluder, $\mathcal{N}(0, 0.002^2)$ per axis, in image-plane units per frame; depth $\mathcal{U}(0.5, 0.9)$ times the target's
Dropouts	10% of tracks, start $\mathcal{U}\{1, \ldots, T-1\}$, length $\mathcal{U}\{2, \ldots, 20\}$
Occluder depth on hidden cells	probability 0.3, for cells hidden only by an occluder and adjacent in time to an observed cell

Loss settings. For the normalization and displacement loss in Section 3.4, we use $\rho_{\min} = 0.3$ and $\lambda = 4$. The loss weights are $w_{ti} = (1 + 3\mu_i)(2 - m_{ti})$, where μ_i marks a moving track as defined above and m_{ti} is the input availability mask. Displacements are measured between consecutive sampled frames, including for windows extracted with temporal stride 2.

Training optimization. All corrector training runs, including the ablations, use 150,000 steps. AdamW with learning rate $3 \cdot 10^{-4}$, weight decay 0.01 and $\beta = (0.9, 0.95)$; linear warm-up over 1,000 steps, then cosine decay to 3% of the peak. Batches hold 16 samples; the forward pass runs

in bfloat16 and gradients are clipped to norm 1. We evaluate an exponential moving average of the weights with decay $\min(0.999, (1+k)/(10+k))$ at step k, taking the checkpoint after 150,000 steps. Training uses one RTX PRO 6000 GPU.

Ablation controls. The simulated-versus-provider-generated comparison in Section 4.4 trains the same network for 150,000 updates per run, with step-indexed cosine decay. The simulated and DA3-lift runs share 3,890 PointOdyssey training windows with usable DA3 alignment; the VGGT-Ω-lift run uses the 3,837 of these windows with usable VGGT-Ω alignment. Provider-generated inputs come from AllTracker paired with DA3 or VGGT-Ω on these synthetic videos, and each model is evaluated with both providers. Thus training steps are matched, while the VGGT-Ω training subset is slightly smaller. Provider inference for data preparation is outside this training-step budget.

This comparison establishes a difference in downstream accuracy under the tested setup. It does not isolate the effects of extreme input errors, domain-specific error patterns, and the training objective. We base our conclusions on the DA3-lift run. The model trained on VGGT-Ω-generated lifts lowers accuracy on every dataset, including PointOdyssey tested with VGGT-Ω, the closest match to its training data. A network should not degrade its own training distribution this strongly, so this suggests a problem in that run's training data, possibly the alignment of VGGT-Ω geometry to the ground-truth targets, which also rejected more windows. We therefore draw no conclusions from this run.

The corruption-family study retrains the corrector after removing each family, or retaining only white noise among the added perturbations. Natural PointOdyssey occlusions remain in all variants, so removing synthetic gaps does not remove completion supervision. Each removal measures the family's contribution in the presence of the remaining training components.

A.4 Dense Inference

We randomly partition tracks using a shuffle seeded with 0 into groups of $P = 1024$, twice the 512 tracks of a training sample (Appendix B.6 varies P at inference), and process up to 8 groups per batch. A short final group is filled by cyclically repeating pixels from its batch. Repeated tracks participate in attention without masking, and their predictions are written to their original pixel indices. Each group retains the full temporal sequence, as in Section 3.3. The depth scale s is computed once over all reference pixels and shared across groups; during training it is computed over the sampled tracks. No scale is shared between clips. Clips whose length is not a multiple of C are extended by at most $C-1$ unobserved frames, with the camera held at its last position. Predictions for these padding frames are discarded.

B Additional Quantitative Results

B.1 Evaluation Protocol

The scored points of a clip are the ground-truth points visible in the first frame whose projection falls inside the image. One ADT clip has no such point and is skipped by every method, leaving 199 clips. Both alignments follow 4RC's protocol (Luo et al., 2026), with the RANSAC seeded per clip and alignment so that every method is scored on the same draws. Dyn selects complete tracks whose cumulative ground-truth motion over the evaluated frames, $\sum_{t=0}^{T-2} \|\mathbf{x}_{t+1,i} - \mathbf{x}_{ti}\|$, exceeds 0.01: metres on WorldTrack and scene units on the 24-frame Kubric clips. For Dyn APD, Sim(3) is re-fitted using only these tracks. AJ and OA follow TAP-Vid (Doersch et al., 2022): a point-frame is predicted visible when the method's visibility exceeds 0.5, the Jaccard index over the aligned trajectories is averaged over the distance thresholds of APD, and OA is the fraction of point-frames whose predicted visibility matches the ground truth. Every metric is averaged over the clips of a dataset, then over the four datasets.

B.2 Dense Tracking on Kubric

This evaluation tests the decomposition on dense ground truth, complementing WorldTrack's sparse evaluation locations, and examines how the result depends on the input geometry. Table 5 scores the dense evaluation set of TrackCraft3R (Nam et al., 2026), 50 Kubric (Greff et al., 2022) clips

with ground truth for every pixel of the first frame, with the alignments of App. B.1; every baseline runs with its WorldTrack settings on the 24-frame clips (App. B.9). Under RANSAC Sim(3), 4RC leads on all points, while V-DPM leads on dynamic points but has substantially higher EPE. Our DA3 and VGGT-Ω configurations trail 4RC, despite outperforming the other methods using the same providers on all-point APD. With 4RC geometry, our results approach 4RC's all-point accuracy and exceed its dynamic-point accuracy with either 2D tracker. Under global-scale alignment, CoWTracker with 4RC geometry leads all three reported metrics. The strong dependence on the geometry source illustrates a boundary of trajectory repair: transfer of the corrector across providers does not eliminate dependence on the quality of their observations.

Table 5: **Dense tracking on Kubric, query frame 0.** TrackCraft3R's 50 Kubric clips, 24 frames, every frame-0 pixel a query, 4RC's dense metric (App. B.2): APD×100 on all and dynamic tracks and EPE (scene units) under RANSAC Sim(3) and under global scale; bold best, underline second best.

	RANSAC Sim(3)			Global scale		
Method	All	Dyn	EPE	All	Dyn	EPE
4RC	**85.33**	78.81	**0.299**	<u>82.40</u>	73.68	<u>0.342</u>
Any4D	65.31	55.51	0.854	42.69	39.77	1.107
MotionCrafter	52.98	47.89	1.765	–	–	–
OpenD4RT	38.40	37.92	1.432	24.32	26.92	1.801
St4RTrack	59.45	54.24	3.640	56.56	51.77	3.585
Track4World	53.55	55.53	2.074	37.37	38.82	2.552
TraceAnything	59.29	50.40	1.023	30.46	28.02	1.252
V-DPM	75.19	**81.03**	3.369	63.70	75.18	3.405
DELTAv2 + DA3	63.27	62.05	1.659	44.73	43.37	2.028
DELTAv2 + VGGT-Ω	62.35	59.19	5.048	41.19	38.52	5.888
TrackCraft3R + DA3	62.76	55.72	1.539	41.42	38.08	2.088
TrackCraft3R + VGGT-Ω	59.27	50.48	4.059	34.81	31.85	6.352
Ours: the corrector on the lift of						
AllTracker + DA3	67.10	61.32	1.794	46.72	44.01	2.144
AllTracker + VGGT-Ω	65.89	59.66	4.999	43.45	39.37	5.813
AllTracker + 4RC	84.28	79.79	0.336	81.99	<u>76.23</u>	0.347
CoWTracker + DA3	67.29	61.40	1.773	47.07	44.33	2.128
CoWTracker + VGGT-Ω	66.57	59.76	4.863	43.54	39.82	5.659
CoWTracker + 4RC	<u>84.88</u>	<u>80.66</u>	<u>0.324</u>	**83.04**	**77.20**	**0.335**

B.3 Visibility Metrics and End-Point Error

Table 6 gives AJ and OA for the methods that predict visibility, each from its own visibility output, and Table 7 the end-point error of every method. EPE complements thresholded APD by measuring the magnitude of position errors. AJ evaluates localization together with visibility; for our method, OA is inherited from the 2D tracker because its visibility scores are returned unchanged (Sec. 3.3); correction changes the 3D positions used by AJ and EPE.

B.4 Global-Scale Alignment

Table 8 repeats Table 1 under global-scale alignment, which keeps the first camera's rotation and translation and fits only a scale. This tests whether the main conclusions depend on the rotational and translational freedom of Sim(3) alignment. Mean accuracy drops for every method, with substantial losses on Panoptic Studio: Depth Anything 3's focal length departs from the true one by more than 10% on 41 of the 50 clips, by 15% in the median, and the rows that read a geometry model's intrinsics lose 18 to 21 points there, 4RC 17. Our configurations using 4RC geometry still lead both benchmark means, but the per-dataset ranking changes: V-DPM leads on Panoptic Studio. In the benchmark means, V-DPM moves ahead of 4RC and within 0.7 points of our VGGT-Ω rows. Correction still improves all six lifts, with gains similar to those under Sim(3): $+1.4$ to $+2.6$ points on all points and $+2.6$ to $+3.8$ on moving points. The mean-accuracy conclusion therefore persists, while the real-world per-dataset claims remain specific to the alignment being reported.

Table 6: **AJ and OA on WorldTrack.** $\times 100$, for the methods that predict visibility, from their own visibility at a threshold of 0.5 under the alignment of Table 1; TrackCraft3R's paper reports them at the best of a threshold sweep.

	ADT		PS		PO		DR		Mean	
Method	AJ	OA	AJ	OA	AJ	OA	AJ	OA	AJ	OA
OpenD4RT	58.82	91.04	69.01	87.34	45.52	**78.63**	63.04	90.03	59.10	86.76
Track4World	72.88	98.06	64.76	86.51	47.53	76.25	67.46	92.79	63.16	88.40
DELTAv2 + DA3	84.18	96.94	55.18	71.57	44.90	74.35	62.74	87.36	61.75	82.56
DELTAv2 + VGGT-Ω	83.83	95.09	57.91	71.32	45.61	74.21	68.54	87.94	63.97	82.14
TrackCraft3R + DA3	82.95	94.26	70.18	88.13	50.29	75.59	66.79	92.69	67.55	87.67
TrackCraft3R + VGGT-Ω	83.48	93.48	72.40	88.29	49.72	75.64	72.68	92.92	69.57	87.58
Ours: the corrector on the lift of										
AllTracker + DA3	89.18	97.82	70.88	86.56	51.64	74.49	67.33	93.02	69.76	87.97
AllTracker + VGGT-Ω	89.75	97.82	76.17	86.56	52.21	74.49	74.27	93.02	73.10	87.97
AllTracker + 4RC	85.76	97.82	74.60	86.56	58.79	74.49	78.17	93.02	74.33	87.97
CoWTracker + DA3	89.99	**98.51**	74.34	**90.03**	54.10	77.06	67.42	**93.03**	71.46	**89.66**
CoWTracker + VGGT-Ω	**90.54**	**98.51**	**80.13**	**90.03**	54.53	77.06	74.42	**93.03**	74.91	**89.66**
CoWTracker + 4RC	86.73	**98.51**	77.93	**90.03**	**61.58**	77.06	**78.34**	**93.03**	**76.14**	**89.66**

Table 7: **End-point error on WorldTrack.** Mean distance to the ground truth in metres after the alignment of Table 1, lower is better. ‡Scored only on the 63.6% of ADT queries inside MotionCrafter's crop; on all queries: 0.610 (App. B.9).

Method	ADT	PS	PO	DR	Mean
4RC	0.153	0.138	0.249	0.149	0.172
Any4D	0.286	0.216	0.335	0.191	0.257
MotionCrafter	0.474‡	0.358	0.820	0.278	0.482
OpenD4RT	0.317	0.161	0.390	0.252	0.280
St4RTrack	0.242	0.258	0.311	0.236	0.262
Track4World	0.269	0.184	0.366	0.219	0.260
TraceAnything	0.198	0.284	0.735	0.378	0.399
V-DPM	0.141	0.116	**0.179**	0.189	0.156
DELTAv2 + DA3	0.330	0.160	0.516	0.270	0.319
DELTAv2 + VGGT-Ω	0.333	0.125	0.595	0.203	0.314
TrackCraft3R + DA3	0.106	0.161	0.348	0.245	0.215
TrackCraft3R + VGGT-Ω	0.100	0.139	0.412	0.181	0.208
Ours: the corrector on the lift of					
AllTracker + DA3	0.089	0.151	0.344	0.242	0.207
AllTracker + VGGT-Ω	0.087	0.109	0.372	0.169	0.184
AllTracker + 4RC	0.114	0.119	0.266	**0.145**	0.161
CoWTracker + DA3	0.087	0.139	0.311	0.241	0.195
CoWTracker + VGGT-Ω	**0.085**	**0.095**	0.347	0.168	0.174
CoWTracker + 4RC	0.110	0.110	0.231	**0.145**	**0.149**

B.5 Ray Masking

Table 9 compares our masked-ray input with a separately trained variant that retains rays when observations are unavailable. Both use AllTracker and $C = 4$. Retaining these rays reduces ADT accuracy by 2.76 and 2.81 percentage points on all points, and by 8.05 and 8.09 on occluded points, with DA3 and VGGT-Ω, respectively. Differences on the other datasets are below one percentage point in either direction.

B.6 Chunk Size and Group Size

We evaluate temporal compression and the amount of across-track context separately. For chunk size, Table 10 compares correctors trained with $C \in \{1, 4, 8\}$, each for 150,000 steps. The $C = 4$

Table 8: **Dense 3D point tracking on WorldTrack.** APD×100 on all (All) and moving (Dyn) points under 4RC's protocol, 84 frames, global-scale alignment (App. B.1), per dataset and averaged; bold best, underline second best. MotionCrafter's coordinate system differs from the first camera's coordinate system, so a scale-only alignment cannot fit it and its row is degenerate. ‡Scored only on the 63.6% of ADT queries inside MotionCrafter's crop; on all queries: 2.83 (All), 5.81 (Dyn) (App. B.9).

	ADT		PS		PO		DR		Mean	
Method	All	Dyn	All	Dyn	All	Dyn	All	Dyn	All	Dyn
4RC	83.09	68.24	69.15	69.15	80.23	76.44	82.95	73.13	78.86	71.74
Any4D	57.60	65.80	63.39	63.39	66.16	66.89	78.15	68.16	66.32	66.06
MotionCrafter	2.12‡	6.91‡	0.00	0.00	0.11	0.02	1.59	1.39	0.96	2.08
OpenD4RT	69.28	67.96	76.87	76.87	65.19	70.39	74.80	76.25	71.53	72.87
St4RTrack	74.33	71.13	69.12	69.12	67.86	68.34	74.10	67.31	71.35	68.97
Track4World	75.63	85.43	71.39	71.39	63.93	65.61	70.74	68.12	70.42	72.64
TraceAnything	73.64	67.34	67.60	67.60	38.44	44.02	60.85	58.59	60.13	59.39
V-DPM	84.41	69.61	**78.40**	**78.40**	81.43	**82.19**	75.56	**78.13**	79.95	77.08
DELTAv2 + DA3	78.63	85.85	65.05	65.05	54.83	56.57	69.41	57.51	66.98	66.25
DELTAv2 + VGGT-Ω	79.43	85.12	70.50	70.50	58.61	60.48	76.19	68.11	71.18	71.05
TrackCraft3R + DA3	89.12	84.04	63.91	63.91	65.51	62.25	71.26	60.14	72.45	67.59
TrackCraft3R + VGGT-Ω	90.46	83.55	68.13	68.13	67.72	64.21	77.69	70.43	76.00	71.58
Ours: the corrector on the lift of										
AllTracker + DA3	90.71	91.40	64.80	64.80	70.22	70.71	72.49	61.92	74.55	72.21
AllTracker + VGGT-Ω	91.93	88.14	71.38	71.38	72.65	73.10	81.15	74.68	79.28	76.82
AllTracker + 4RC	89.22	90.71	71.31	71.31	80.42	76.78	**84.63**	75.18	81.40	78.49
CoWTracker + DA3	90.73	**91.90**	65.93	65.93	72.12	72.93	72.33	62.17	75.28	73.23
CoWTracker + VGGT-Ω	**92.32**	88.82	72.49	72.49	74.15	74.62	80.82	74.10	79.94	77.51
CoWTracker + 4RC	89.08	91.30	72.96	72.96	**82.27**	78.82	84.12	75.80	**82.11**	**79.72**

Table 9: **Ray masking.** APD3D×100 under Sim(3) alignment with AllTracker and $C = 4$. Separately trained correctors with or without masking unavailable rays. Occ denotes ground-truth-occluded point–frames.

		ADT		PS		PO		DR	
Geometry	Subset	With masking	Without masking	With masking	Without masking	With masking	Without masking	With masking	Without masking
DA3	All	93.69	90.93	85.46	85.44	76.04	76.03	79.50	79.52
DA3	Occ	90.27	82.22	76.08	76.30	73.19	73.03	63.17	64.05
VGGT-Ω	All	94.04	91.23	90.40	90.36	76.10	76.10	85.74	85.75
VGGT-Ω	Occ	90.12	82.03	82.04	82.54	72.49	72.23	71.26	71.85

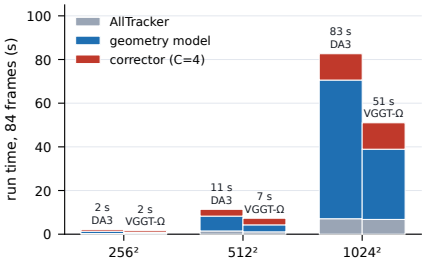
model has the highest reported accuracy; $C = 8$ reduces correction time by about a third with a loss of at most 0.3 APD3D points. The $C = 1$ model is slower and less accurate under the reported training setup.

Group size varies only at inference, keeping the checkpoint fixed (Fig. 6). Reducing P from 1024 to 64 lowers dynamic-point accuracy by 1.3 and 1.9 points with DA3 and VGGT-Ω, respectively, versus about 0.8 on all points. Increasing P from 1024 to 4096 changes every score by at most 0.2 points. We therefore use $P = 1024$.

For M total tracks and N temporal chunks, across-track attention costs $O(NP^2)$ per group (Sec. 3.3) and $O(MNP)$ across M/P groups. At fixed M and N, increasing P from 1024 to 4096 therefore multiplies total across-track attention cost by four. This factor applies to that attention component; total pipeline runtime also includes temporal attention, other corrector operations and the two providers.

Table 10: **Chunk-size accuracy and runtime breakdown.** Left: separately trained chunk-size variants with AllTracker; APD3D×100 averaged over four datasets under RANSAC Sim(3) alignment. Times measure correction of 84 frames at 512^2 on one RTX PRO 6000. Right: runtime by pipeline stage at three resolutions, with $C = 4$.

	DA3 input		VGGT-Ω input		Corrector time (s)	
C	All	Dyn	All	Dyn	DA3	VGGT-Ω
1	82.93	81.08	85.83	83.79	10.0	9.7
4	83.68	81.74	86.57	84.92	3.1	3.0
8	83.59	81.48	86.48	84.75	2.0	1.9



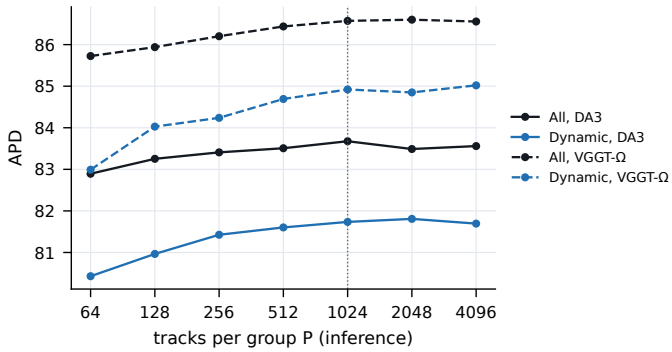


Figure 6: **Inference group size.** Mean WorldTrack APD3D×100 with AllTracker and fixed corrector weights ($C = 4$). The dotted line marks our setting, $P = 1024$.

B.7 Run Time and Memory

We time complete pipelines on 84 frames at 512^2 pixels on one RTX PRO 6000 GPU (Table 1). Our VGGT-Ω configurations achieve comparable or higher average accuracy than 4RC and V-DPM while running 2.7–4.4× faster than 4RC and at least 50× faster than V-DPM; faster baselines are substantially less accurate (Fig. 1, right). At 1024^2, our AllTracker configurations with DA3 or VGGT-Ω remain feasible where 4RC, V-DPM and DELTAv2 exceed GPU memory.

Table 11 gives the measurements behind the times of Table 1 at three input resolutions, and the right panel of Table 10 splits our pipeline into its stages. Timing starts with the 84 decoded frames in host memory and ends with the dense trajectories of every reference pixel in host memory, so it excludes video decoding and model loading and includes every model a method runs: for ours, the 2D tracker, the geometry model and the corrector; for TrackCraft3R and DELTAv2, the geometry model and the tracker; for the self-contained trackers, their own networks. Each method processes the target resolution rounded to its patch size, or its fixed input size where its code fixes one. The stage breakdown tests the practical cost of the grouped corrector in Section 3.3. With AllTracker, geometry estimation accounts for a growing share of runtime as resolution increases. The complete pipeline therefore need not inherit the corrector's linear scaling in pixel count.

B.8 Image-Plane Accuracy

Table 12 complements the 3D evaluation with the Average Percentage of 2D Points within Delta (APD2D). It uses the same 84-frame clips, initial query selection and All/Dyn definitions as Appendix B.1, with the validity and coverage restrictions described below. APD2D averages the fraction of eligible point–frames within $\{1, 2, 4, 8, 16\}$ pixels of the ground-truth image position after rescaling coordinates to 256×256.

Table 11: **Run time and peak memory of the complete pipelines.** 84 frames, dense output, one RTX PRO 6000, each method's released code, at three input resolutions (1024^2: a single run); OOM: exceeds the 96 GB card; –: not run; † resolution fixed by the method's code; OpenD4RT runs at 256^2 only. Ours with $C = 4$.

	Run time (s)			Peak memory (GB)		
Method	256^2	512^2	1024^2	256^2	512^2	1024^2
Ours: the corrector on the lift of						
AllTracker + DA3	2.1	11.4	82.6	10.1	21.3	63.7
AllTracker + VGGT-Ω	1.5	7.3	51.0	7.9	12.5	43.9
CoWTracker + DA3	2.8	15.3	OOM	10.6	36.5	OOM
CoWTracker + VGGT-Ω	2.2	11.7	OOM	10.6	36.5	OOM
4RC	4.3	31.8	OOM	11.0	26.8	OOM
V-DPM†	77.3	590.8	OOM	26.9	85.7	OOM
TrackCraft3R + DA3 / VGGT-Ω	5.0 / 4.4	22.1 / 18.5	162.7 / 132.3	9.8 / 7.6	21.0 / 12.2	63.4 / 30.9
DELTAv2 + DA3 / VGGT-Ω	7.8 / 7.2	33.4 / 29.8	OOM / OOM	9.8 / 8.3	32.5 / 32.5	OOM / OOM
Any4D†	0.6	3.2	OOM	13.7	45.2	OOM
TraceAnything†	1.9	5.2	20.4	3.0	4.8	11.8
Point4D	268.5	2768.4	OOM	24.6	72.1	OOM
Track4World	1.5	8.4	85.7	10.1	21.1	70.2
St4RTrack	0.6	2.9	OOM	6.8	20.8	OOM
MotionCrafter†	4.8	20.4	OOM	10.9	29.1	OOM
OpenD4RT (256^2 only)	284.2	–	–	7.8	–	–

Projection protocol. We project predicted trajectories through their associated cameras without fitting an alignment to ground truth. APD2D therefore does not use the Sim(3) fit of Table 1. Re-anchoring to the first camera is a coordinate conversion applied consistently to trajectories and cameras. Projections are mapped to original image coordinates, accounting for resizing and crop offsets, before the common 256×256 rescaling. Camera sources and recovery procedures are specified in Appendix B.9 and Table 13.

Scored point–frames. Ground-truth point–frames behind the ground-truth camera are excluded from each APD2D scoring set. Within the remaining set, a prediction behind its camera or with a non-finite projection counts as a miss at every threshold, rather than being removed from the denominator. Thus the initial query selection is shared with APD3D, but the scored point–frames can differ.

Native AllTracker and CoWTracker predictions are read at the same query pixels and evaluated directly in 2D using the same coordinate scaling and ground-truth masks. They provide references without a 3D prediction or camera projection. MotionCrafter's restricted ADT query coverage, detailed in Appendix B.9, affects both its ADT scores and its benchmark means.

Among 3D methods, CoWTracker with 4RC geometry leads mean All accuracy, while Track4World leads mean Dyn and our best configuration is second. Switching from AllTracker to CoWTracker improves every dataset and geometry pairing, raising the benchmark mean by 1.86–1.92 points on All and 2.32–2.36 on Dyn. Across geometry providers, the mean varies by 0.15 on All and 0.56–0.57 on Dyn. Tracker choice therefore has a larger effect on APD2D than geometry choice within these configurations, supporting the role of pretrained correspondences in our decomposition (Section 4.2).

Native CoWTracker exceeds every projected 3D method in each reported column. All six of our configurations fall below their own native tracker in both benchmark means. For CoWTracker with 4RC geometry, the gaps are 1.48 points on All and 3.82 on Dyn. These gaps measure the complete lifting, initialization and correction pipeline; the table does not isolate the corrector's effect on 2D accuracy. The APD3D gains over initialized lifts in Figure 3 establish improved 3D estimation, while these references reveal remaining room to preserve the input's image-plane accuracy.

APD2D measures the image-plane accuracy of trajectories and cameras jointly; it does not isolate correspondence quality or capture errors purely along a viewing ray. For cameras fitted by the evaluator, scores also depend on that fitting procedure. The relatively small variation across geometry

Table 12: **Dense 3D tracking on WorldTrack — image-plane accuracy.** APD2D (%) on all (All) and dynamic (Dyn) point-frames: the fraction within $\{1, 2, 4, 8, 16\}$ px of the ground-truth 2D position after rescaling each frame to 256^2, averaged over the five thresholds. Each method's trajectories are projected through its associated cameras. Bold and underlined values indicate the best and second-best results. ‡ MotionCrafter's ADT cells are computed on 63.6% of the ADT queries, not directly comparable to the full-support cells of the other methods; full-support values 12.33 (All) and 5.98 (Dyn), reason in App. B.9. Every Panoptic Studio query moves more than 1 cm, so its All and Dyn columns coincide by construction (dynamic fraction 1.00; 0.30 on ADT, 0.50 on PO, 0.32 on DR). MotionCrafter, OpenD4RT, St4RTrack and TraceAnything predict no cameras; theirs are derived from each method's own outputs of the same forward pass (App. B.9), never from ground truth. Track4World and DELTAv2 are reported with their input capped to our grid (App. B.9); ADT is uncapped, where the cap removes no information. The last block is each 2D tracker's own pixel predictions, with no 3D lift and no cameras; these rows are shown for reference and are excluded from the bold/underline ranking, which covers the 3D methods only.

	ADT		PS		PO		DR		Mean	
Method	All	Dyn	All	Dyn	All	Dyn	All	Dyn	All	Dyn
4RC	62.06	27.36	51.79	51.79	71.50	60.91	79.15	65.26	66.13	51.33
Any4D	40.90	23.60	31.46	31.46	56.99	44.39	76.93	50.10	51.57	37.39
MotionCrafter	17.47‡	10.34‡	15.11	15.11	21.88	13.70	40.69	30.58	23.79‡	17.43‡
OpenD4RT	45.94	21.77	46.90	46.90	56.13	49.43	72.58	57.41	55.39	43.88
St4RTrack	59.53	28.47	39.12	39.12	63.95	47.47	79.77	50.73	60.59	41.45
Track4World	76.31	**66.42**	63.67	63.67	<u>76.82</u>	**68.95**	**89.73**	**74.30**	76.63	**68.34**
TraceAnything	23.13	11.51	32.30	32.30	35.29	28.14	59.93	41.65	37.66	28.40
V-DPM	50.15	22.88	55.04	55.04	**77.00**	64.66	85.10	64.42	66.82	51.75
DELTAv2 + DA3	76.27	<u>65.59</u>	66.20	66.20	72.94	61.94	87.28	66.65	75.68	65.09
DELTAv2 + VGGT-Ω	74.75	62.12	66.15	66.15	72.52	62.32	87.52	67.33	75.23	64.48
TrackCraft3R + DA3	48.89	20.04	42.44	42.44	63.29	45.66	75.85	54.15	57.62	40.57
TrackCraft3R + VGGT-Ω	52.14	22.21	41.59	41.59	61.10	44.52	77.27	55.37	58.02	40.92
Ours (2D tracker + geometry provider)										
AllTracker + DA3	76.59	60.36	63.41	63.41	74.51	66.05	86.35	70.14	75.22	64.99
AllTracker + VGGT-Ω	76.47	59.19	63.56	63.56	74.33	65.92	86.67	70.88	75.26	64.89
AllTracker + 4RC	76.45	60.84	63.66	63.66	74.68	66.29	86.69	71.00	75.37	65.45
CoWTracker + DA3	**76.98**	62.64	67.18	67.18	76.31	67.41	88.08	72.15	<u>77.14</u>	67.35
CoWTracker + VGGT-Ω	76.78	61.43	<u>67.23</u>	<u>67.23</u>	76.11	67.28	88.35	72.88	77.12	67.21
CoWTracker + 4RC	<u>76.91</u>	63.23	**67.35**	**67.35**	76.41	<u>67.62</u>	<u>88.39</u>	<u>72.93</u>	**77.27**	<u>67.78</u>
Native 2D references (no 3D prediction)										
AllTracker (2D tracker alone)	77.18	67.90	64.29	64.29	75.00	68.68	88.31	73.55	76.19	68.60
CoWTracker (2D tracker alone)	78.32	71.22	68.59	68.59	77.44	70.58	90.67	76.00	78.75	71.60

providers therefore does not imply that geometry is unimportant for 3D tracking. The table contains no visibility split, so it does not establish where the native-to-projected gap arises relative to occlusion.

B.9 Baseline Settings

We use each method's released code, with the input grids detailed below. The native frame size is 512×512 for ADT and 360×640, 720×1280 and 540×960 for Panoptic Studio, Dynamic Replica and PointOdyssey, the last three at an aspect ratio of exactly 16:9. We give each method's input grid as $H \times W$ on these three datasets, with the ADT grid in parentheses where it differs: 4RC (Luo et al., 2026) 280×504 (504^2); V-DPM (Sucar et al., 2026) and Any4D (Karhade et al., 2025) 294×518 (518^2); St4RTrack (Feng et al., 2025) and TraceAnything (Liu et al., 2025) 288×512 (512^2); MotionCrafter (Zhu et al., 2026) 320×640 with its centre crop; OpenD4RT (RHOS Team, 2026) a resize to 256^2 without a crop, a change of aspect ratio of $1.78\times$ on the 16:9 datasets, whose dense output is read at every second pixel, an effective 128×128 grid; TrackCraft3R (Nam et al., 2026) a resize to 480×832, run over interleaved 12-frame clips at stride 8; and Track4World (Lu et al., 2026) and DELTAv2 (Ngo et al., 2025a), the latter in its dense mode, 280×504 (512×512 for ADT). Every method is read out at the query pixels scaled to its grid, and where a crop drops a query pixel the nearest prediction is used, except on ADT for MotionCrafter. Its 320×640 window is a 1:2 landscape shape, which the three widescreen datasets fit with about 11% of their rows

Table 13: **Camera sources for APD2D.** Cameras are predicted, supplied by a geometry provider, or recovered from model outputs; no ground-truth camera parameters are used.

Methods	Projection cameras
4RC, Any4D, Track4World, V-DPM	Predicted by the method
TrackCraft3R, DELTAv2	Input geometry cameras
Our configurations	Predicted by the geometry provider
MotionCrafter	Fitted by the evaluator to predicted point maps
St4RTrack, OpenD4RT, TraceAnything	Recovered from each method's own outputs

trimmed; ADT is square, so the model's own resize scales it to 640×640 and keeps only the central 320 rows, leaving 36.4% of the ADT query points outside anything the model saw. Rather than extrapolate them, we exclude those queries and report MotionCrafter's ADT cells on the remaining 63.6%, marked as such wherever they appear. Feeding the whole frame to the model instead, either squashed onto 320×640 or at a uniform 448^2, scores lower than the crop on the queries all variants predict, so no full-coverage mapping replaces it. OpenD4RT's and MotionCrafter's outputs are not metric, and MotionCrafter's are not expressed in the first camera's coordinate system; the Sim(3) alignment absorbs both. On the 24-frame Kubric clips of App. B.2 the same settings run with fewer windows.

Cameras for APD2D. Table 13 distinguishes cameras predicted by a method, supplied with its input geometry, and recovered by the evaluator. Our method, TrackCraft3R and DELTAv2 use their geometry provider's predicted intrinsics and extrinsics, following the shared estimated-geometry setup in Section 4.1. For methods without camera outputs, cameras are recovered from their own predictions without ground-truth camera parameters, as in Table 12.

For MotionCrafter, we fit camera pose and focal length jointly to pixel–point correspondences from a subsample of its predicted point maps, initialized from the previous frame and refined under a robust loss. The principal point is fixed at the image centre; no ground-truth camera parameters enter this recovery.

MotionCrafter's restricted ADT coverage is marked in the APD2D table, including its means; scored on all ADT queries, its ADT scores are 12.33 (All) and 5.98 (Dyn).

B.10 Additional Qualitative Results

Visualization protocol for Figures 4 and 7. Queries form a grid spaced 16 pixels apart at native resolution inside the eroded first-frame object mask; 4RC and V-DPM outputs are read at the nearest pixels of their grids. TrackCraft3R uses its interleaved benchmark mode, with queries lifted using VGGT-Ω depth. Each row shows the third-to-last frame of its clip. Track colors identify queries in raster order; trails show the current and preceding 20 frames, with dots marking current positions. A novel camera in VGGT-Ω coordinates is mapped into each method's reconstruction by a similarity fitted to the first-frame point maps, with 20% outlier trimming over two rounds. Backdrops combine confident background points from all frames, outside the dilated object mask, with object points from the displayed frame.

Additional method comparisons. Figure 7 adds two DAVIS (Perazzi et al., 2016) examples. In *soccerball*, which includes occlusion by a tree trunk, our trajectories form cleaner arcs ending on the ball, illustrating the temporal coherence targeted by trajectory repair. In *lady-running*, ours follow the body in an ordered band, while TrackCraft3R's scatter below it; 4RC's and V-DPM's remain compact. The same corrector weights serve different tracker–geometry combinations in the two rows, illustrating the modular use of learned motion integration.

Isolating trajectory repair. Figure 8 isolates the effect of the corrector (Section 3.3): the tracks, geometry, point cloud and viewpoint are fixed between the lift and corrected panels. The corrector receives only geometric trajectory features, without images. Missing observations in the lift are initialized by interpolation, or held at the last observation for terminal gaps (Section 3.2). The

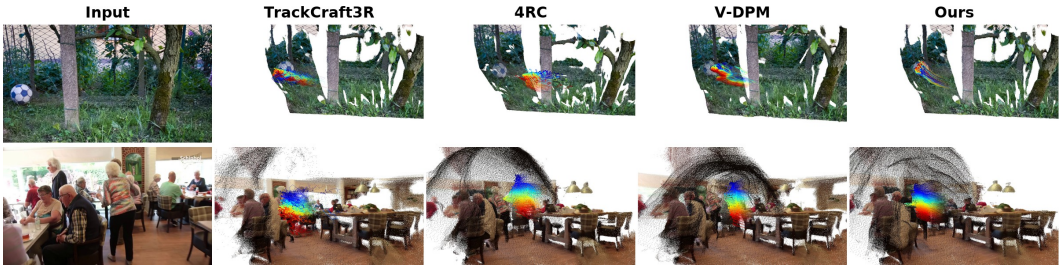


Figure 7: **Additional qualitative comparisons.** Top: DAVIS *soccerball*; bottom: *lady-running*. Ours uses CoWTracker with VGGT-Ω (top) and AllTracker with DA3 (bottom), with the same $C = 4$ corrector. TrackCraft3R uses VGGT-Ω in both rows; 4RC and V-DPM use their own reconstructions. Thus ours and TrackCraft3R use different geometry on *lady-running*. Methods track the same reference pixels, shown on their respective reconstructions from matched novel viewpoints. Colors identify tracks.

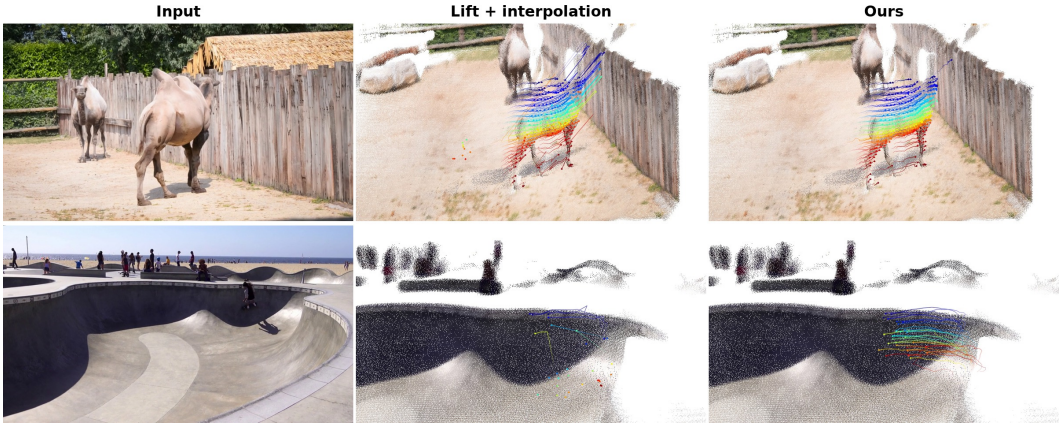


Figure 8: **Effect of trajectory correction.** Input, lift with interpolated gaps, and the same lift after correction ($C = 4$). Both trajectory panels use CoWTracker tracks and DA3 geometry, drawn on the same DA3 point cloud from the same novel viewpoint. Top: DAVIS *camel*; correction reduces stray ground points and produces a more ordered band along the body. Bottom: DAVIS *skate-park*; scattered strands become a smoother band along the bowl's lip.

reduced scatter and more coherent motion illustrate learned trajectory repair, complementing the quantitative gains over the initialized lift in Figure 3.

Query sampling, colors, trails and backdrop construction follow the protocol above.