Point4D: Long-range 4D Motion Reconstruction

Anonymous Author(s)
Affiliation
Address
email

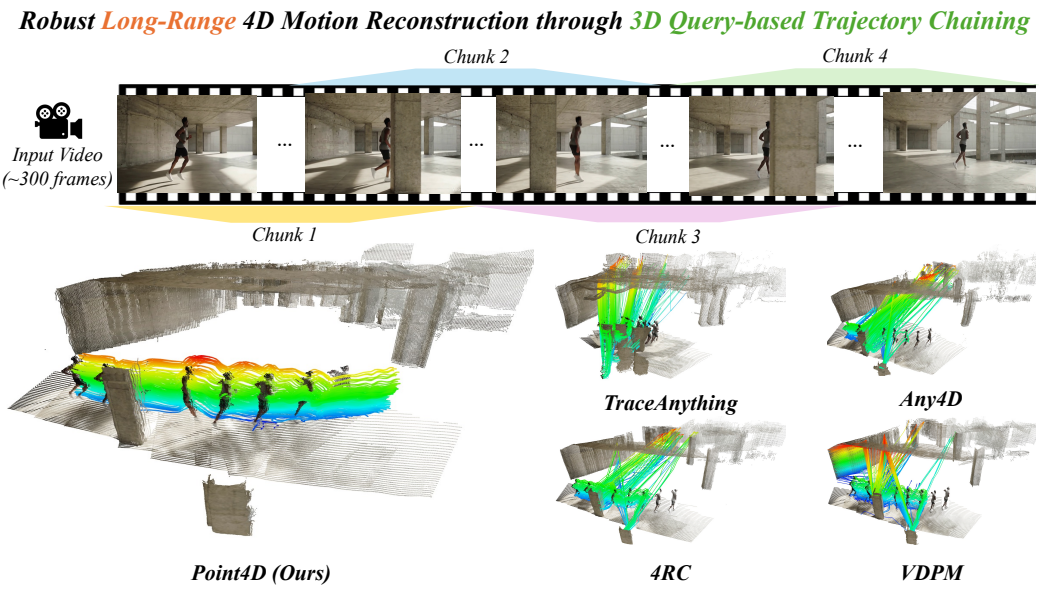

Figure 1: **Point4D enables long-range 4D reconstruction via autoregressively chaining motion reconstruction across overlapping chunks**. Existing 4D reconstruction methods (e.g., Any4D [13], 4RC [18]) predict 4D motion for query 2D pixels, and cannot be easily chained under occlusion as the tracked points may not be visible in overlapping frames. Point4D decodes motion for query 3D points instead of 2D pixels, allowing direct chaining for long-range motion reconstruction under occlusions.

Abstract

We introduce Point4D, a feed-forward model for 4D reconstruction of long-range video sequences. Point4D is able to reliably infer dense per-point 3D trajectories across multi-hundred video sequences, unlike existing 4D methods that are limited to short input windows of at most a few dozen frames. A key innovation that enables this is our flexible 3D query-based motion decoder that decouples trajectory prediction from image-plane visibility. The predicted 3D endpoints are then directly re-queried in the next chunk without re-projection or matching. Furthermore, we show that extracting and reusing a visual across an arbitrary frame where the point is visible leads to superior performance than purely relying on the source patch. Overall, Point4D achieves state-of-the-art performance across diverse long-video tracking benchmarks spanning over 200 frames and substantially outperforms previous feed-forward 4D methods.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

1 Introduction

Consider the runner in Fig. 1: a person jogging down a long corridor, vanishing behind structural columns and reappearing moments later. We humans can effortlessly follow the runner's continuing trajectory over this clip, understanding their long-range motion despite occlusions, and even reasoning about where they may be when not directly visible. In this work, we seek to build a computational system that can similarly perform *long-range 4D motion reconstruction* from a monocular video — recovering per-point 3D trajectories across hundreds of frames.

While initial approaches [31, 32, 41, 36] for such '4D reconstruction' leveraged slow and expensive iterative optimization, there has been a marked shift towards feed-forward methods [13, 42, 18, 22, 17, 25] which, following the successes in feed-forward 3D reconstruction [34, 30], have extended such multi-view models to additionally perform motion prediction. Specifically, by adding scene flow prediction 'heads' to multi-view models, these approaches allow decoding each pixel's 3D position at any queried timestep, thereby inferring the 4D motion of the scene in an efficient feed-forward manner. While these methods deliver impressive results, they are designed to operate over short input windows of a handful of frames. Long videos that span across hundreds of frames and more expose two challenges that short windows do not face. **Firstly**, jointly processing the full frame set is computationally infeasible for the underlying transformer encoders, and **secondly**, the points to be tracked routinely become occluded or leave the field of view between observations.

The first challenge is not unique to long-range 4D reconstruction. Indeed, approaches for long-range static reconstruction are similarly bottlenecked by the computational complexity, and a common solution is to *chain* short-range predictions across overlapping chunks [5]. Whereas chaining static reconstruction merely requires aligning the coordinate systems from independent per-chunk reconstruction, *chaining for 4D reconstruction requires autoregressive motion prediction* – the motion for a point tracked from an earlier chunk needs to be queried again from a later chunk to continue the point track. Existing 4D methods that compute 3D track predictions for query *pixels* are not well suited for such chaining, either requiring reprojection from 3D to 2D for re-querying or a computationally expensive 'matching' of independently predicted 4D trajectories. However, these strategies break down when a point is occluded or outside the field of view at the chunk boundary which are the very situations that are routine in long videos. Our key insight is to **query points, not pixels** – a motion reconstruction decoder with 3D points as queries allows the predicted endpoint of one chunk to be re-queried in the next directly, with no reprojection or matching (see Fig. 2). More generally, 3D queries also encourage the decoder to focus on motion reconstruction instead of geometry estimation (which already captured in input) and can also help resolve ambiguous 2D pixel queries e.g., on object boundaries where it maybe unclear if the query is a foreground or background pixel.

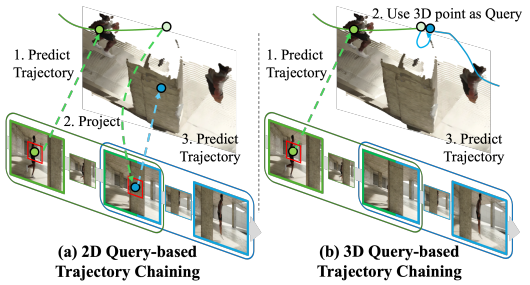

Figure 2: **Point4D uses 3D query representations for chaining motion prediction.** A 2D-query-based motion decoder requires image reprojection and is not robust to chaining under occlusions. Our insight is to leverage 3D queries, allowing direct chaining without reprojection.

We operationalize this insight in Point4D , a feed-forward model that decodes 3D motion from a 3D-coordinate query. A shared ViT encoder produces a scene representation along with per-frame depth and camera poses; a lightweight cross-attention decoder takes a 3D query, target time and camera indices, and a visual descriptor extracted once from any prior frame where the point is visible, and predicts the queried point's 3D position at the target time. To process long videos, we partition the video into overlapping chunks, autoregressively re-querying each predicted 3D endpoint in the next chunk based on the decoded trajectory from the previous one, while reusing the same visual descriptor throughout. We validate our approach across multiple datasets and show that Point4D substantially improves over the prior state of the art on long-video 4D tracking while also improving over its 2D-query counterpart for short-range motion reconstruction.

In summary, our contributions are:

- We build a feed-forward pipeline for *long-range* 4D motion reconstruction, recovering per-point motion across hundreds of frames via trajectory chaining across chunks.
- We introduce a 3D-query-based motion decoding that decouples trajectory prediction from image-plane visibility. This also allows predicted 3D endpoints to directly be propagated across chunks.
- We achieve state-of-the-art long-video 4D tracking, with substantial gains over chaining-based feed-forward 4D methods and online 3D point trackers.

2 Related Work

3D Tracking. To model scene motion, point tracking and optical flow [26] methods estimate pixel-level correspondences across frames. Subsequent work extended tracking to longer temporal ranges through correlation-based matching and iterative updates [6, 7, 12], yet these methods operate purely in 2D. The TAPVid-3D benchmark [15] motivated lifting tracks into 3D: DELTA [19] and SpatialTracker [36] combine 2D trackers with monocular depth to track points in camera coordinates. More recently, TAPIP3D [41] lifts video features into a camera-stabilized 3D point cloud and iteratively refines trajectories in 3D space, while SpatialTrackerV2 [37] learns geometry and point motion jointly in an end-to-end architecture, eliminating separate depth estimation. However, these methods only track sparse query points, often rely on off-the-shelf depth modules, and require iterative refinement that limits speed. Point4D can predict dense 3D trajectories in a single forward pass via query-based decoding, while remaining competitive with iterative trackers on sparse benchmarks.

4D Reconstruction of Dynamic Scenes. 4D reconstruction aims to recover both the 3D structure of a scene and how it changes over time. Early methods rely on per-scene optimization [32], achieving high fidelity but at prohibitive cost. Recent feed-forward approaches extend 3D reconstruction architectures to the temporal dimension, predicting dense geometry and motion jointly from multiple frames [8, 17, 13, 25, 18]. While methods differ in motion representation – scene flow from a canonical view [13], continuous trajectory fields [17], or dense point maps at queried timesteps [18, 25], all decode motion for every pixel through DPT-style heads or similar dense decoders. [42] introduces a query-based 4D decoder that predicts the 3D position of a 2D query point at a target timestep and coordinate frame. However, all methods are constrained to short input windows of fewer than 100 frames. To this end, we adopt the flexible query-based decoding from [42] and extend it to work for 3D queries. This decouples tracking from visibility and enables longer video 4D reconstruction to over 300 frames through direct re-querying of predicted 3D positions.

Long-range 3D reconstruction. Feed-forward 3D reconstruction methods [34, 30, 16, 14] have shown impressive results in multi-view 3D reconstruction. However, these approaches only work for a limited number of frames, due to high memory requirements, and quickly run out of memory for longer sequence inputs. CUT3R [33], InfiniteVGGT [40], StreamingVGGT [47] proposed memory mechanism for long-sequence reconstructions, while VGGT-Long [5] proposes a chunking and loop closure mechanism. More recently, Loger [44] proposed a test-time training mechanism for long sequence 3D reconstruction. These works demonstrate that 3D reconstruction models trained for small inputs can be effectively chained to handle longer sequences. Point4D extends this insight to the 4D setting. Rather than simply chaining predictions for static geometry, Point4D leverages 3D point queries to enable long-range motion prediction. To our knowledge, Point4D is the first method to recover dense scene flow and long-range 3D tracks in this scalable, feed-forward manner.

3 Method

Our goal is to recover dense 3D trajectories from long monocular video sequences. To achieve this, we propose **Point4D** , a feed-forward model that uses *3D coordinate queries* as input instead of inferring motion from 2D pixels. By querying directly in 3D space, we decouple trajectory prediction from image-plane visibility. Building upon established visual geometry prediction models [30, 16, 18] and D4RT [42], we first encode the video into a feature representation while predicting depth and camera poses to support query decoding (Sec. 3.1). We then reformulate the query mechanism to use 3D points (x, y, z) in the source frame, paired with a visual descriptor extracted from any timestep where

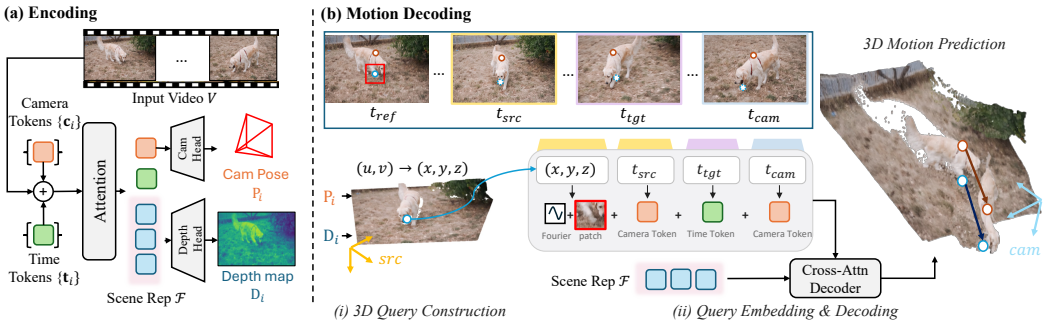

Figure 3: **Overview of Point4D .** (a) Given a monocular video V, a ViT encoder with alternating frame-wise and global attention produces a scene representation $\mathcal{F}$ with per-frame camera and time tokens, depth maps, and camera poses. (b) We then decode the motion of each query points. First, a 3D query $\mathbf{p} = (x, y, z)$ is constructed as the point's position at source time t_{src} in frame t_{src}'s camera coordinates. Since the visual patch can be extracted from a different frame t_{ref} where the point is visible, the query point may be occluded or out of frame at t_{src}. (ii) The query embedding $\mathbf{q}$ is formed by summing Fourier-encoded coordinates with camera and time tokens and cross-attends to $\mathcal{F}$ to predict the point's 3D position at the target timestep.

the point is visible (Sec. 3.2). This formulation allows us to predict a point's 3D position at any target time regardless of its image-plane visibility. Finally, we present a simple chaining framework for long-range 3D trajectory inference (Sec. 3.3). Since 3D queries remain well-defined under occlusion, we can propagate trajectories across overlapping video chunks by re-querying predicted 3D endpoints, naturally surviving occlusions and field-of-view exits. An overview of our method is shown in Fig. 3.

3.1 Preliminaries

Feed-forward visual geometry prediction. Recent feed-forward methods [16, 30, 43, 34, 3] process multi-view images through a ViT backbone $\mathcal{E}$ with alternating frame-wise and global self-attention layers, producing per-frame patch tokens $\mathbf{Z}_i$ and camera token $\mathbf{c}_i$ for each frame $i \in \{1, \ldots, T\}$. Dedicated heads decode camera poses $\mathbf{P}_i$ from $\mathbf{c}_i$ and depth maps $\mathbf{D}_i$ from $\mathbf{Z}_i$ via a DPT decoder [23]. To handle dynamic scenes, recent works is augmented with a learnable time token $\mathbf{t}_i$ per frame [18], initialized from a sinusoidal encoding of the normalized timestep $\in [0, 1]$ and refined through the same attention layers. This yields a scene representation $\mathcal{F}$ along with per-frame tokens and predictions:

$$\mathcal{F}, \{\mathbf{c}_i, \mathbf{t}_i, \mathbf{P}_i, \mathbf{D}_i\}_{i=1}^{T} = \mathcal{E}(V) \tag{1}$$

Query-based 4D decoding. While standard 4D reconstruction methods typically utilize DPT-style heads for dense geometry, D4RT introduces on-demand query-based decoding. This mechanism defines a query as $(u, v, t_{\text{src}}, t_{\text{tgt}}, t_{\text{cam}}, S)$, where (u, v) represents a 2D pixel in frame t_{src} and S provides visual context via a local RGB patch. The decoder $\mathcal{D}$ then predicts the 3D position of that pixel at a target timestep t_{tgt}, expressed in camera t_{cam}'s coordinate system. Each query is mapped to an embedding $\mathbf{q}$ and independently cross-attends to the scene representation $\mathcal{F}$ to produce a predicted point:

$$\hat{\mathbf{p}} = \mathcal{D}(\mathbf{q}, \mathcal{F}) \in \mathbb{R}^3. \tag{2}$$

This formulation enables flexible decoding of arbitrary pixel sets across varying timesteps and coordinate frames. However, like all prior 4D methods, D4RT relies on 2D pixel coordinates, requiring the queried point to be visible in the source frame for initialization.

3.2 3D Query-based Decoder

Existing 4D reconstruction methods infer motion via predicting scene flow for 2D pixels and require the tracked 3D point to be visible in the source frame. This breaks down when the point is occluded or outside the field of view, and across video chunks where the predicted 3D position at the end of one chunk may have no visible pixel to re-query in the next. We address this with two key design

choices. **First**, we replace 2D pixels with *3D points* as queries, which decouples point identity from image-plane visibility, and allows the decoder to focus on predicting motion rather than jointly recovering geometry. **Second**, we supply appearance context via a local image patch drawn from *any* frame where the point is visible, rather than tying visual context to the query frame. This enables cross-chunk trajectory chaining as the patch can be sourced from a different chunk entirely, even when the point is occluded or out of view in the current one.

3D query construction. To specify a query, we first select a pixel (u, v) that is visible in some frame t_{ref} and extract a local image patch S around it as a visual descriptor. We then obtain the query coordinate $\mathbf{p} = (x, y, z)$ by taking this point's 3D position at a source time t_{src}, expressed in frame t_{src}'s camera coordinate system. When $t_{\text{ref}} \neq t_{\text{src}}$, the point may have moved and may be occluded or outside the field of view at t_{src}. However, the 3D coordinate remains well-defined since it does not depend on the pixel projection. The full query is $(\mathbf{p},\, t_{\text{src}},\, t_{\text{tgt}},\, t_{\text{cam}}, S)$: *where is the 3D point at position* $\mathbf{p}$ *(in frame* t_{src}*'s coordinate frame) with visual descriptor* S *at timestep* t_{tgt}*, expressed in camera* t_{cam}*'s coordinate frame?*

Query embedding and decoding. To build the query embedding $\mathbf{q}$, the 3D spatial coordinates $\mathbf{p}$ are first encoded with Fourier features [28]. The temporal and camera indices are encoded by passing the corresponding tokens from the encoder through dedicated MLPs: t_{src} and t_{cam} use camera tokens $C_{t_{\text{src}}}$ and $C_{t_{\text{cam}}}$, while t_{tgt} uses time token $T_{t_{\text{tgt}}}$. These encodings are summed with S to form the query embedding $\mathbf{q}$. The decoder passes q through cross-attention layers attending to $\mathcal{F}$, producing the predicted 3D position $\hat{\mathbf{p}} = \mathcal{D}(\mathbf{q}, \mathcal{F}) \in \mathbb{R}^3$. Following D4RT, the decoder uses only cross-attention with no self-attention between queries, so each query is decoded independently, enabling flexible batching of arbitrary query sets at inference.

Loss. The primary loss $\mathcal{L}_{\text{point}}$ is an L1 loss on the predicted 3D position, where both prediction and target are passed through a signed log-transform $\phi(x) = \text{sign}(x)\log(1+|x|)$ to dampen the influence of far-away points. The confidence loss $\mathcal{L}_{\text{conf}}$ modulates $\mathcal{L}_{\text{point}}$ by a per-query confidence score, so that uncertain predictions are penalized less heavily. The auxiliary reprojection loss $\mathcal{L}_{\text{2d}}$ is an L1 loss on the predicted 2D projection of $\hat{P}$ into frame t_{cam}, enforcing consistency between the predicted 3D position and camera geometry. The auxiliary visibility loss $\mathcal{L}_{\text{vis}}$ is a binary cross-entropy loss on a per-query logit predicting whether the point is visible at t_{tgt}. The total loss is:

$$\mathcal{L} = \mathcal{L}_{\text{point}} + \mathcal{L}_{\text{conf}} + \mathcal{L}_{\text{2d}} + \mathcal{L}_{\text{vis}}. \tag{3}$$

3.3 Trajectory Chaining

Long videos cannot be processed in a single forward pass due to memory constraints. We therefore partition the video into short, overlapping chunks and encode each chunk independently. To produce *coherent long-range trajectories* from these chunk-level predictions, requires (1) aligning each chunk's independent 3D coordinate frame into a single global frame, and (2) propagating point identity across chunk boundaries so that trajectories stitch together correctly.

Algorithm 1 Trajectory Chaining for Long Video

Require: Video V of T frames; query pixels $\{(u_i, v_i)\}$ visible in frame 0
Ensure: 4D trajectories $\{P_i(t)\}_{t=0}^{T-1}$ in a global coordinate frame

Partition V into K overlapping chunks $\{C_k\}_{k=0}^{K-1}$
$p_i \leftarrow \textsc{Unproject}(u_i,\, v_i,\, D_0) \quad \forall i$ ▷ lift pixel query to 3D at t=0
$S_i \leftarrow \textsc{ExtractPatch}(\text{frame}_0,\, u_i,\, v_i) \quad \forall i$ ▷ visual descriptor, extracted once
for $k = 0, \ldots, K-1$ **do**
  $F_k \leftarrow \textsc{Encode}(C_k)$ ▷ 4D scene representation for chunk k
  $P_i(t) \leftarrow \mathcal{D}(p_i,\, S_i,\, F_k,\, \pi_t) \quad \forall i,\, \forall t \in C_k$ ▷ decode 3D position at each frame
  if $k < K-1$ **then**
    $(c, \mathbf{R}, \mathbf{t}) \leftarrow \textsc{EstimateSim3}(C_k \cap C_{k+1})$ ▷ align chunks via overlap depth
    $p_i \leftarrow c\,\mathbf{R}\,P_i(t_h) + \mathbf{t} \quad \forall i$ ▷ propagate 3D query into C_{k+1}'s frame; reuse S_i
  end if
end for
return $\{P_i(t)\}_{t=0}^{T-1}$ transformed to global frame-0 coordinates

Alignment of chunk-level 4D reconstruction Since each chunk's geometry predictions are up to an arbitrary scale and coordinate frame, we align adjacent chunks via a Sim(3) transformation [27] estimated from dense depth predictions on the shared overlap frames. Composing these pairwise Sim(3) transforms places every chunk's predictions into a single global coordinate frame. While this procedure suffices for static 3D reconstruction, we also need to propagate the *temporal point identity* across chunks for 4D scene reconstruction.

Chaining trajectories with 3D queries. Our 3D query formulation reduces this correspondence problem to a single coordinate transform. Within each chunk, we decode every query point's trajectory by sweeping the target time t_{tgt} across the chunk's frames. At the first overlapping frame t_h between chunk k and chunk $k{+}1$, we take the predicted 3D position $P_i(t_h)$ and apply the already-estimated Sim(3) to express it in chunk $k{+}1$'s local coordinates, yielding the re-query coordinate $\mathbf{p}_i$ for the next chunk. The visual descriptor S_0, extracted once from the first frame where the point is visible, is reused across all subsequent chunks without re-extraction. This procedure chains trajectories regardless of whether the point is visible in the overlap: the 3D coordinate is always well-defined, so occluded and out-of-frame points propagate without failure. In contrast, 2D-query methods must project the predicted 3D position back to pixel coordinates to re-query, which is undefined for occluded points and compounds camera prediction error for visible ones.

3.4 Implementation Details

We initialize the encoder and geometry heads from Depth Anything 3 [16] pretrained weights, while the decoder is trained from scratch. We train on a mixture of dynamic datasets with ground-truth trajectories (PointOdyssey [45], Dynamic Replica [11], Kubric Movi-F [10], Cotracker-Kubric [12], Waymo [1]) and static datasets (ScanNet [4], ScanNet++ [39], BlendedMVS [38], Co3Dv2 [24], WildRGBD [35]), where static points are treated as stationary trajectories. Training proceeds in two phases: the first uses image width 252 with [8, 18] frames per sample, and the second increases to width 420 with [4, 7] frames. Further details are in Appendix A.

4 Experiments

4.1 Experimental Setting

Baselines We compare against two categories of methods namely, (1) Feed-forward 4D reconstruction methods such as TraceAnything [17], Any4D [13], 4RC [18], VDPM [25] which predict dense per-pixel 3D positions at queried timesteps and are most directly comparable to ours. (2) 3D point trackers such as SpatialTrackerV2 [37] and TAPIP3D [41] which represent the current state of the art in per-point 3D tracking but do not support dense per-pixel queries.

Setup. We evaluate Point4D on (A) long-video tracking, which requires trajectory chaining across chunks, and a simpler (B) single-chunk tracking. For long-video tracking, we use PointOdyssey, Dynamic Replica sequences of 200 frames and PStudio sequences of 150 frames, partitioned into chunks of 48 frames with 8-frame overlap. These are then aligned via Umeyama alignment [27] on overlapping point maps (Sec. 3.3). Feed-forward 4D methods are evaluated with selection-based reprojection chaining, which picks the overlapping frame with best depth agreement for each point (Appendix C). SpatialTrackerv2 and TAPIP3D use their online inference modes. For single-chunk tracking, we evaluate on TAPVid3D Parallel Studio, LSFOdyssey, and Dynamic Replica with up to 64 frames per sequence.

Metrics. Following the benchmarking protocol of recent works [8, 13], we report end-point error (EPE) and average percentage of points within distance thresholds $\{0.1, 0.3, 0.5, 1.0\}$ m (APD), after median scale alignment of ground truth and predicted trajectories.

$$\text{EPE}_{i,t} = \|P_{i,t} - \tilde{P}_{i,t}\| \tag{4}$$

$$\text{APD} = \sum_{i,t} \mathbb{1} \cdot (\text{EPE}_{i,t} < \delta_{\text{3D}}) \tag{5}$$

where $\delta_{3D} \in \{0.1, 0.3, 0.5, 1.0\}m$

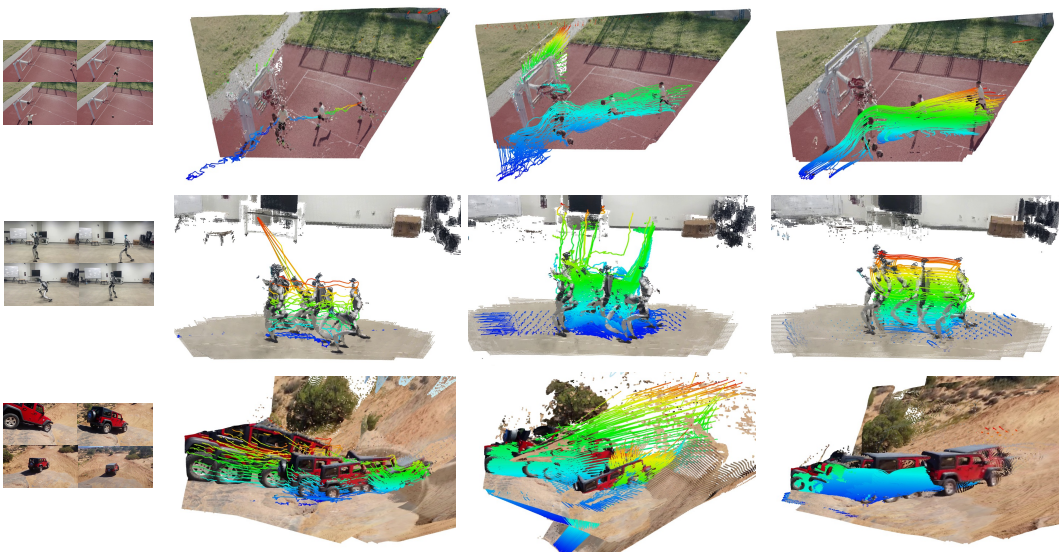

Figure 4: **Point4D produces reliable and consistent 4D tracking over 200-frame sequences via trajectory chaining compared to existing methods**. 4RC struggles to maintain correspondence and its tracks are wrongly chained with background points. SpatialTrackerV2 only can predict a sparser trajectory due to memory constraints.

	Method	PointOdyssey [45]			Dynamic Replica [11]			PStudio [15]		
		EPE ↓	APD ↑	Survival ↑	EPE ↓	APD ↑	Survival ↑	EPE ↓	APD ↑	Survival ↑
Iter.	TAPIP3D	0.529	0.544	0.415	0.311	0.654	0.540	0.310	0.638	0.426
	SpatialTrackV2	0.529	0.588	0.452	0.256	0.750	0.671	0.131	0.874	0.748
Feed-Forward	TraceAnything	2.059	0.159	0.073	0.779	0.466	0.351	0.643	0.519	0.324
	Any4D	0.939	0.468	0.291	0.582	0.548	0.396	0.471	0.572	0.405
	4RC	0.688	0.597	0.457	0.409	0.712	0.586	0.383	0.641	0.542
	V-DPM	0.644	0.651	0.478	0.363	0.713	0.555	0.263	0.759	0.640
	Point4D	0.526	0.598	0.535	0.256	0.759	0.669	0.246	0.717	0.643

Table 1: **Long-Video 4D Tracking via Trajectory Chaining.** Point4D achieves state-of-the-art performance across all three metrics while providing dense motion predictions. We report end-point error (EPE ↓), average percentage of points within distance thresholds (APD ↑), and survival rate (↑) on sequences of 200 frames, partitioned into chunks of 48 frames with 8-frame overlap. Red, Orange, and Yellow indicate the top three results.

For long-video, we additionally report Survival rate [45]: the average fraction of video length before tracking failure. A point i is considered failed at frame t if $\|\hat{\mathbf{p}}_i^t - \mathbf{p}_i^t\|_2 > \delta_{3D}$, and we report:

$$\text{Survival}(\delta_{3D}) = \frac{1}{N}\sum_{i=1}^{N}\frac{t_i^{\text{fail}} - 1}{T} \tag{6}$$

where t_i^{fail} is the first failure frame. We average over $\delta_{3D} \in \{0.1, 0.3, 0.5, 1.0\}$ m.

4.2 4D Tracking

Long-video trajectory chaining. Table 1 compares all methods on 200-frame sequences that require chaining across multiple chunks. Feed-forward 4D baselines use reprojection-based chaining, which must project predicted 3D points back to 2D to re-query in the next chunk; this is undefined for occluded points and compounds camera prediction error even for visible ones, causing trajectories to drift or break at chunk boundaries. Iterative 3D trackers avoid explicit chaining by processing

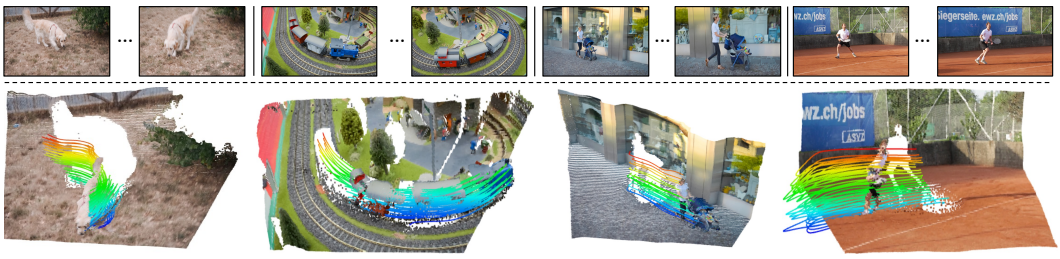

Figure 5: **Single-chunk 4D tracking on DAVIS [21].** Point4D produces consistent 3D trajectories across challenging real-world sequences.

	Method	LSFOdyssey [29]		Dynamic Replica [11]		PStudio [15]	
		EPE ↓	APD ↑	EPE ↓	APD ↑	EPE ↓	APD ↑
2D Tracker	MonST3R + CoTracker3	0.61	0.51	0.81	0.43	0.51	0.52
	MASt3R + CoTracker3	0.83	0.46	0.40	0.58	0.43	0.54
	VGGT + CoTracker3	0.47	0.59	0.26	0.69	0.26	0.69
	MapAnything + CoTracker3	0.63	0.35	0.25	0.71	0.63	0.51
Iter.	TAPIP3D	0.35	0.66	0.87	0.50	0.30	0.65
	SpatialTrackv2	0.34	0.68	0.69	0.62	0.21	0.75
Feed-Forward	St4RTrack	0.56	0.48	0.17	0.81	0.41	0.53
	Trace Anything	0.76	0.42	0.34	0.66	0.25	0.72
	Any4D	0.27	0.72	0.07	0.93	0.28	0.66
	VDPM	0.14	0.85	0.14	0.84	0.17	0.79
	4RC	0.16	0.87	0.07	0.95	0.29	0.66
	Point4D	0.27	0.74	0.07	0.94	0.25	0.70

Table 2: **Single-chunk 4D tracking.** Point4D performs comparably to existing methods on short sequences that do not require trajectory chaining, which confirms that the long-video gains in Table 1 stem from the proposed 3D query formulation, not a stronger single-chunk decoder. We report EPE (↓) and APD (↑). Red, Orange, and Yellow indicate the top three results.

frames online, but do not support dense tracking due to expensive iterative optimization. Point4D chains trajectories by directly re-querying predicted 3D coordinates in each subsequent chunk and overcomes failure modes highlighted in (Sec. 3.3) and outperforming baselines in most sequences.

Figure 4 visualizes results on 200-frame sequences: we sample a regular grid of query points on the first frame and decode their trajectories across chunks (chunk size 48, overlap 16). Point4D maintains continuous trajectories through occlusion and field-of-view exits, while 2D-query methods lose track at chunk boundaries. SpatialTrackerV2 uses a sparser query grid due to GPU memory constraints, resulting in visibly sparser trajectories.

Single-chunk tracking. Table 2 evaluates all methods on sequences short enough to fit within a single chunk, isolating decoder accuracy from chaining. Point4D performs comparably to other feed-forward 4D methods while outperforming both 2D tracker + 3D reconstruction pipelines and iterative optimization-based methods. Combined with the long-video results, this shows that our advantage on longer sequences comes from reliable 3D-query chaining rather than a gap in single-chunk decoding. Figure 5 shows representative results on DAVIS sequences with up to 64 frames.

4.3 Ablations and Analysis

Ablation on query formulation. We ablate the two key design choices in Point4D's query formulation: (1) using 3D coordinates instead of 2D pixels, and (2) training with visual descriptors from arbitrary visible frames rather than only the source frame. Table 3 compares three variants: **2D** queries with pixel coordinates (u, v) as in D4RT [42]; **3D (source patch)**, which uses 3D coordinates but always extracts the visual descriptor from the source frame, so the model never sees occluded or out-of-frame queries during training; and **3D** (Point4D), which combines 3D coordinates with descriptors from arbitrary visible frames.

Query Formulation	Long-Video Tracking				Single-Chunk Tracking			
	PointOdyssey [45]		Dynamic Replica [11]		LSFOdyssey [29]		Dynamic Replica [11]	
	EPE ↓	SR ↑	EPE ↓	SR ↑	EPE ↓	APD ↑	EPE ↓	APD ↑
2D	1.022	0.245	0.505	0.479	0.511	0.604	0.248	0.749
3D (source patch)	0.803	0.388	0.925	0.247	0.442	0.620	0.121	0.800
3D (Point4D)	**0.526**	**0.535**	**0.256**	**0.669**	**0.259**	**0.744**	**0.070**	**0.936**

Table 3: **Point4D 's proposed 3D query formulation is optimal for reliable decoding.** We ablate this choice on both single-chunk and long-video chaining benchmarks. EPE (lower is better), APD and Survival Rate (higher is better).

Switching from 2D to 3D queries improves single-chunk accuracy across all benchmarks (Table 3). The 3D coordinate provides geometry directly as input, letting the decoder focus on predicting motion rather than jointly recovering depth. The 3D source patch variant already improves over 2D on single-chunk tracking, but it cannot chain trajectories because the model expects the visual descriptor from the source frame — which may not contain the point at chunk boundaries. Training with descriptors from arbitrary visible frames further improves single-chunk accuracy and trajectory chaining.

Robustness of tracking across chunks.

To verify that our 3D query formulation enables reliable trajectory chaining across video chunks, we measure per-chunk APD and EPE on TAPVid3D Parallel Studio sequences. For the entire sequence of 150 frames, we partitioned it into 48-frame chunks with 8-frame overlap, and compared how each method's accuracy evolves as chaining proceeds.

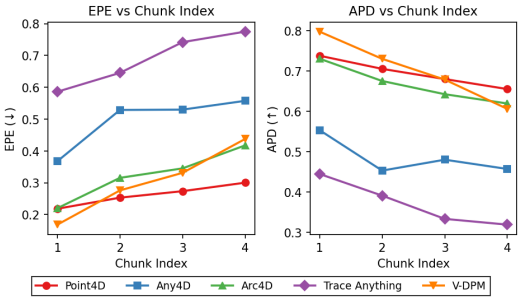

Figure 6: **Per-chunk tracking accuracy on PStudio.** Point4D (red) degrades slowly across chunks, while others worsen faster as chaining proceeds.

Figure 6 shows that Point4D degrades slowly across chunks in both APD and EPE, while baselines deteriorate faster as chaining proceeds. The gap widens because 2D-query methods compound error at every chunk boundary through reprojection, whereas our 3D re-querying avoids this entirely (Sec 3.3). Notably, some baselines like VDPM start with higher first-chunk accuracy than Point4D , yet Point4D surpasses them within a few chunks. This confirms that the advantage of 3D queries lies not in single-chunk decoding but in sustaining accuracy across many chunks, which matters for long-video 4D reconstruction.

5 Discussion

We presented Point4D , a feed-forward model for 4D reconstruction of long-range video sequences. Point4D infers motion using 3D coordinate queries compared to prior feed-forward methods which inferred motion from 2D pixels. This provides two advantages as it decouples trajectory prediction from image-plane visibility, and allows for direct requerying of 3D points across chunks even when occluded or out of field of view. Moreover, we show that our visual descriptor extracted from arbitrary frames improves greatly over the source patch descriptor alone. We show these design choices enable Point4D to outperform existing methods on long-range videos. Ultimately, we believe Point4D will serve as a foundation step towards achieving reliable 4D reconstruction on in the wild videos of arbitrary length and serve for diverse applications in generative AI, AR/VR and robotics.

Limitations. Trajectory chaining requires that each point is visible in at least one frame of the chunk being decoded. Since the scene representation encodes only what the encoder has observed, a point absent from all frames in a chunk cannot be tracked within it. Additionally, inference-based 3D queries are based on predicted depth, and this depth error can compound across chunks.

References

[1] 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*, pages 22488–22497, 2024.

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[3] Zhongxiao Cong, Qitao Zhao, Minsik Jeon, and Shubham Tulsiani. Flow3r: Factored flow prediction for scalable visual geometry learning. *arXiv preprint arXiv:2602.20157*, 2026.

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

[5] Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. Vggt-long: Chunk it, loop it, align it–pushing vggt's limits on kilometer-scale long rgb sequences. *arXiv preprint arXiv:2507.16443*, 2025.

[6] 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.

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

[8] Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J Black, Trevor Darrell, and Angjoo Kanazawa. St4rtrack: Simultaneous 4d reconstruction and tracking in the world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8503–8513, 2025.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013.

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

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

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

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

[20] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7855–7862. IEEE, 2019.

[21] 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*, pages 724–732, 2016.

[22] Shenhan Qian, Ganlin Zhang, Shangzhe Wu, and Daniel Cremers. Flow4r: Unifying 4d reconstruction and tracking with scene flow. *arXiv preprint arXiv:2602.14021*, 2026.

[23] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021.

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

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

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

[27] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4):376–380, 2002.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[30] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[31] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *International Conference on Computer Vision*, 2023.

[32] 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*, pages 9660–9672, 2025.

[33] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025.

[34] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20697–20709, 2024.

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

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

[37] 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*, pages 6726–6737, 2025.

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

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

[40] Shuai Yuan, Yantai Yang, Xiaotian Yang, Xupeng Zhang, Zhonghao Zhao, Lingming Zhang, and Zhipeng Zhang. Infinitevggt: Visual geometry grounded transformer for endless streams. *arXiv preprint arXiv:2601.02281*, 2026.

[41] Bowei Zhang, Lei Ke, Adam W Harley, and Katerina Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. *arXiv preprint arXiv:2504.14717*, 2025.

[42] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K Barral, Raia Hadsell, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. *arXiv preprint arXiv:2512.08924*, 2025.

[43] 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. *arXiv preprint arXiv:2410.03825*, 2024.

[44] Junyi Zhang, Charles Herrmann, Junhwa Hur, Chen Sun, Ming-Hsuan Yang, Forrester Cole, Trevor Darrell, and Deqing Sun. Loger: Long-context geometric reconstruction with hybrid memory. *arXiv preprint arXiv:2603.03269*, 2026.

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

[46] Yang Zhou, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Haoyu Guo, Zizun Li, Kaijing Ma, Xinyue Li, Yating Wang, Haoyi Zhu, et al. Omniworld: A multi-domain and multi-modal dataset for 4d world modeling. *arXiv preprint arXiv:2509.12201*, 2025.

[47] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

A Training Details

Table 4 summarizes the training datasets and their sampling ratios. OmniWorld [46] contains dynamic content but lacks trajectory ground truth; we restrict its queries to $t_{\text{src}} = t_{\text{tgt}}$, reducing the task to depth and relative-pose estimation without requiring cross-frame trajectory labels.

For each frame, we sample $N = 750$ query pixels, with 40% drawn from edge regions detected by Sobel filtering to encourage coverage of object boundaries. The query indices t_{src}, t_{tgt}, and t_{cam} are sampled uniformly from the frame indices, with 40% of queries constrained to $t_{\text{cam}} = t_{\text{tgt}}$ so the model frequently predicts points in the target frame's own coordinate system.

We use AdamW with a peak learning rate of 1×10^{-4} for 150 epochs, linearly warmed up till 10 epoch and then cosine-decayed. During training, the learning rate for models initialized from DepthAnything3 is scaled by 0.1. Training is done on 8 H100 GPUs, and the loss for dynamic points is upweighted relative to static points. We apply color jittering, Gaussian blurring, random rescaling, and aspect-ratio augmentation throughout the training.

Table 4: **Training datasets.** Sampling ratio denotes the proportion of samples drawn per epoch. Dynamic datasets provide ground-truth trajectories; static datasets treat all points as stationary.

Dataset	Dynamic	Sampling Ratio
PointOdyssey [45]	✓	24.7%
Dynamic Replica [11]	✓	24.7%
Kubric CoTracker [12]	✓	14.8%
Kubric Movi-F [10]	✓	14.8%
Waymo Drivetrack [1]	✓	7.4%
OmniWorld [46]	✓†	2.5%
ScanNet [4]	✗	2.5%
ScanNet++ [39]	✗	2.5%
BlendedMVS [38]	✗	2.5%
Co3Dv2 [24]	✗	2.5%
WildRGBD [35]	✗	1.2%

† Dynamic content but no trajectory GT; queries restricted to $t_{\text{src}} = t_{\text{tgt}}$.

B Video depth and camera pose estimation.

We evaluate video depth and camera pose estimation on Sintel [2], Bonn [20], and KITTI [9]. Depth accuracy is measured by absolute relative error (AbsRel, ↓) and the inlier ratio $\delta < 1.25$ (↑). Camera pose is assessed using Absolute Trajectory Error (ATE), translational Relative Pose Error (RPE_t), and rotational Relative Pose Error (RPE_r), computed after global SE(3) alignment. As shown in Table 5, Point4D performs on par with DA3, while outperforming other feed-forward 4D reconstruction methods across both tasks. This indicates that Point4D , built on the DA3 backbone, maintains accurate scene geometry understanding — important for ensuring that the 3D queries used in motion reconstruction are reliably initialized from predicted depth.

C Analysis of Trajectory Chaining by Visibility Regime

To understand how trajectory chaining performs under different visibility conditions, we break down EPE on PointOdyssey long-video sequences by the visibility regime at the handoff frame: *Visible* (the point projects into the frame and is unoccluded), *Occluded* (the point projects into the frame but is occluded by another object), and *OOF* (the point projects outside the frame boundary).

Chaining strategies for 2D-query baselines. Since 2D-query methods require projecting predicted 3D points to pixel coordinates for re-querying, the choice of handoff strategy significantly affects chaining quality. Table 6a ablates four strategies using 4RC as the base method. *First-frame* projects all points to the first overlapping frame and clips out-of-frame projections to the image boundary. *Stationary* and *Linear* apply the same projection but extrapolate out-of-frame points as stationary or with constant velocity, respectively. *Selection* projects each point at every overlapping frame, selects

	Video Depth Estimation						Camera Pose Estimation					
	KITTI [9]		Bonn [20]		Sintel [2]		Bonn [20]			Sintel [2]		
Method	AbsRel	$\delta < 1.25$	AbsRel	$\delta < 1.25$	AbsRel	$\delta < 1.25$	ATE	RPE_t	RPE_r	ATE	RPE_t	RPE_r
DA3	0.053	0.976	**0.069**	**0.967**	0.238	0.655	0.029	0.011	0.667	0.124	0.053	0.479
TraceAnything	0.1055	0.903	6.964	0.462	0.506	0.409	0.040	0.015	43.280	0.499	0.322	10.801
Any4D	0.090	0.939	0.414	0.600	0.670	0.384	0.059	0.023	0.632	0.619	0.241	0.741
4RC	0.051	0.960	0.076	0.910	0.483	0.581	0.030	**0.009**	0.645	0.397	0.190	0.654
VDPM	0.068	0.945	0.086	0.925	0.239	0.651	**0.028**	**0.009**	0.664	0.133	0.063	0.508
Point4D	**0.051**	**0.978**	0.072	0.934	**0.202**	**0.707**	0.029	0.010	**0.588**	0.114	**0.049**	**0.470**

Table 5: **Video depth and camera pose estimation.** Point4D achieves state-of-the-art accuracy among feed-forward 4D reconstruction methods. We report AbsRel (↓) and $\delta < 1.25$ (↑) for depth, and ATE (↓), RPE_t (↓), RPE_r (↓) for pose on KITTI, Bonn, and Sintel.

Table 6: **EPE by visibility regime on PointOdyssey long-video sequences.** (a) Ablation of chaining strategies for 2D-query methods using 4RC. Selection-based chaining, which picks the overlapping frame with best depth agreement, outperforms simpler strategies across all regimes. (b) Method comparison using Selection chaining for all baselines. Point4D reduces EPE on Visible and Occluded points through 3D re-querying; the higher OOF error reflects queries for points absent from the next chunk's scene representation.

(a) Chaining strategy ablation (4RC)

Chaining	Visible	Occluded	OOF	All
First-frame	0.763	0.967	0.575	0.867
Stationary	0.756	0.952	0.513	0.823
Linear	0.754	0.957	0.575	0.837
Selection	**0.614**	**0.755**	**0.501**	**0.688**

(b) Method comparison

Method	Visible	Occluded	OOF	All
TraceAnything	2.045	2.099	2.015	2.059
Any4D	0.857	1.034	0.909	0.939
4RC	0.614	0.755	**0.501**	0.688
VDPM	0.640	0.731	0.557	0.644
Point4D	**0.529**	**0.588**	0.675	**0.526**

the frame where the projected depth best matches the depth head's prediction, and re-queries at that frame, which reduces occlusion-related errors by choosing the frame where the point is most likely visible. Selection outperforms all alternatives across regimes (Table 6a), so we use it as the chaining strategy for all 2D-query baselines in our experiments.

Comparison across methods. Table 6b compares Point4D against 2D-query baselines (all using Selection chaining) broken down by visibility regime. Point4D achieves lower EPE on Visible and Occluded points, confirming that 3D re-querying successfully chains both unoccluded and occluded trajectories without the projection error that degrades 2D methods. However, Point4D shows higher EPE on OOF points. This is expected: if an out-of-frame point is never observed in the next chunk, the scene representation lacks information about it and prediction quality degrades. By contrast, 2D methods clip OOF projections to the image boundary and re-query *some* point—the resulting trajectory is incorrectly chained to a different surface point, but the numerical error can be lower than predicting from an unobserved query.

D Additional Long Video Tracking Results

We provide additional tracking results for long videos with up to 200 frames in Figure 7. Each chunk has size 48 with 16 overlapping frames. Point4D show robust trajectory prediction across long videos, demonstrating the effectiveness of 3D query based trajectory chaining.

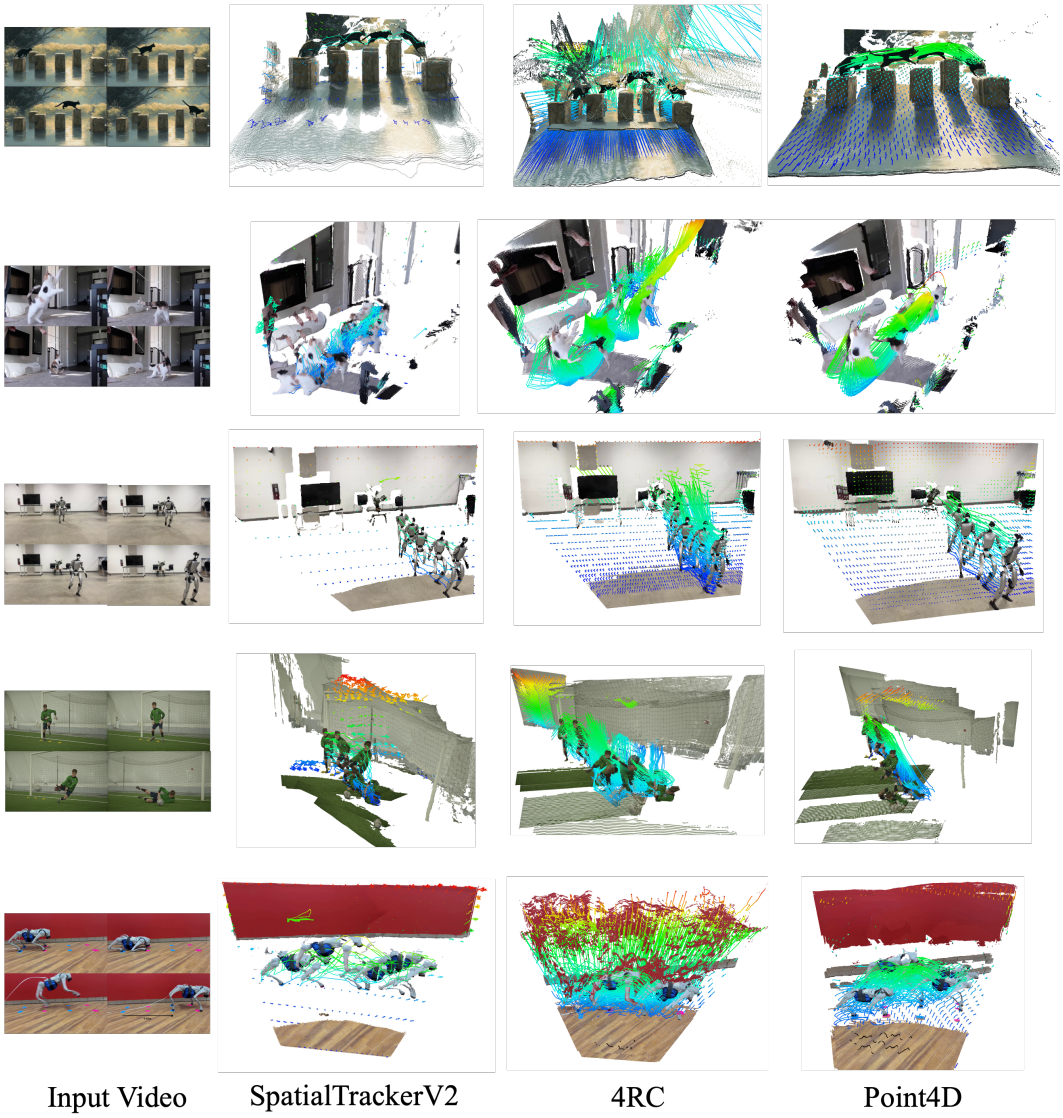

Figure 7: **Additional results for long-video tracking via trajectory chaining.** Point4D produces consistent 3D trajectories across videos with up to 200 frames.