Efficiently Reconstructing Dynamic Scenes One 🔪 D4RT at a Time

Chuhan Zhang* Guillaume Le Moing* Skanda Koppula*◇ Ignacio Rocco*
Liliane Momeni* Junyu Xie°[1] Shuyang Sun* Rahul Sukthankar*
Joëlle K. Barral* Raia Hadsell* Zoubin Ghahramani* Andrew Zisserman*°
Junlin Zhang* Mehdi S. M. Sajjadi*[2]

Setting

Dynamic 4D Reconstruction from Video

Input: A monocular RGB video

Output:

1. Video depth
2. 3D point trajectories
3. Dynamic point clouds
4. Camera Intrinsics
5. Camera extrinsics

Main Challenges:

Recover both **scene geometry** and **temporal motion** while disentangling **object motion** from **camera motion.**

Goal:

Build a unified feedforward model for both static and dynamic 4D reconstruction.

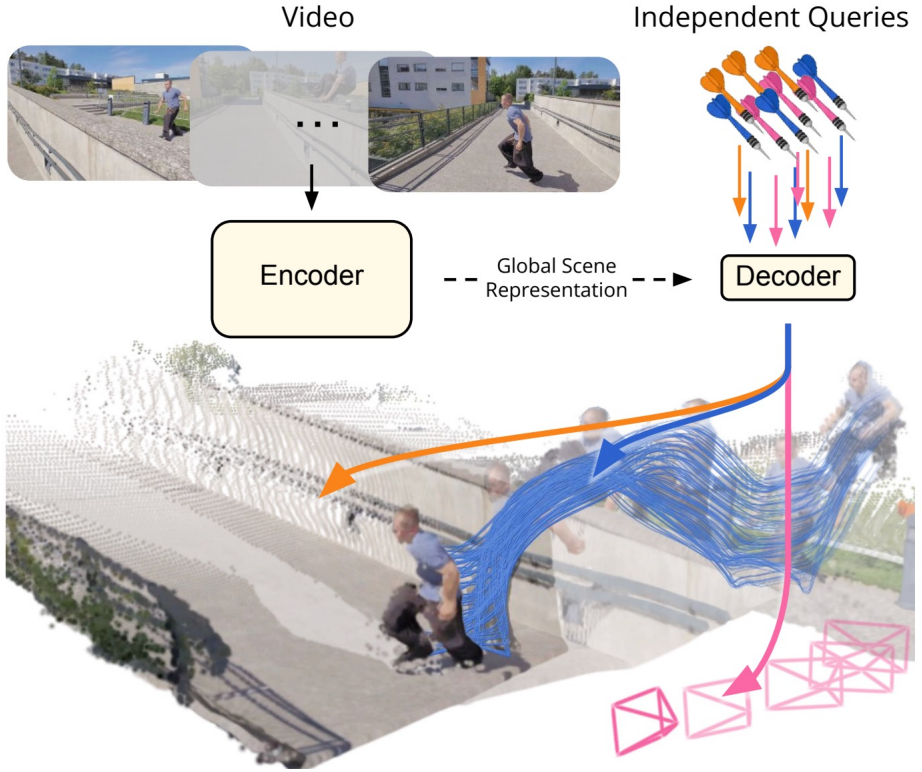


Figure 1. **D4RT** is a unified, efficient, feedforward method for ***Dynamic 4D Reconstruction and Tracking***, unlocking a variety of outputs including point cloud (), point tracks (), camera parameters () through a single interface.

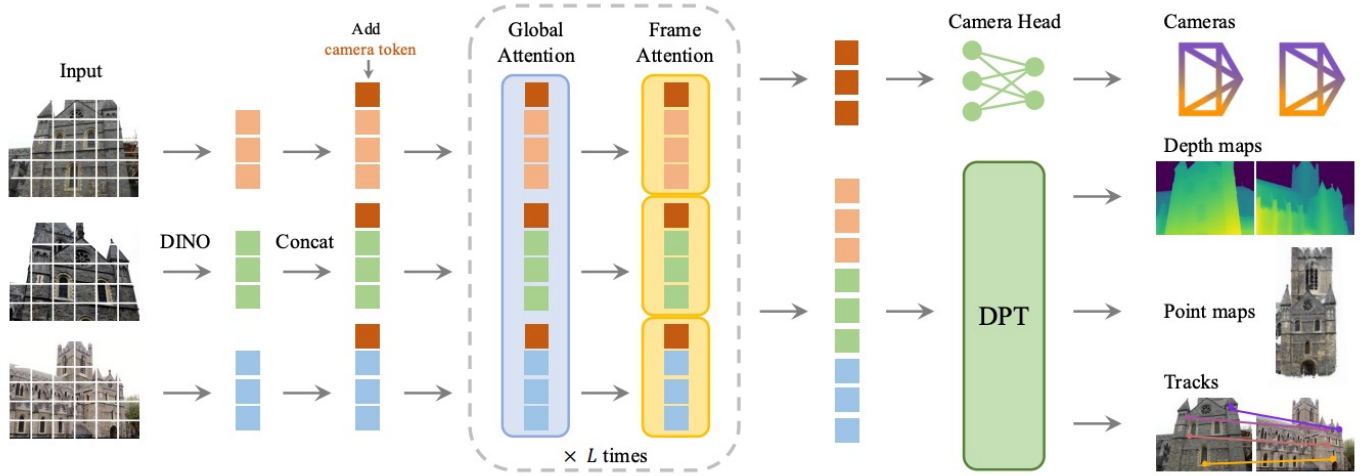
Motivation

Limitation of Existing methods

1. Reconstruction methods often assume a static scene.
2. Dynamic objects cause duplicated or missing geometry.
3. Optimization-based methods require expensive test-time refinement.
4. Feedforward methods usually use separate decoders for depth, pose, and point maps.
5. Tracking methods often rely on iterative refinement or only reconstruct sparse trajectories.

Model	Task		Functionality		Architecture	
	3D Reconstruction	Dynamic Correspondence	Flexible Ref. Frames	Sparse Decoding	Global Context	Single Decoder
MegaSaM [29]	✓	✗	✗	✗	✗	✗
DUSt3R[51]	✓	✗	✗	✗	✗	✗
VGGT [48]	✓	✗	✗	✗	✓	✗
π^3 [54]	✓	✗	✓	✗	✓	✗
St4RTrack [11]	✓	✓	✗	✗	✗	✗
STv2 [56]	✓	✓	✗	✓	✓	✗
D4RT (Ours)	✓	✓	✓	✓	✓	✓

Table 2. **Model capabilities** – We highlight both the tasks our model executes but also its comprehensive functionality and simple model architecture.



VGGT pipeline

Core Idea

From Dense Decoding to On-Demand Querying

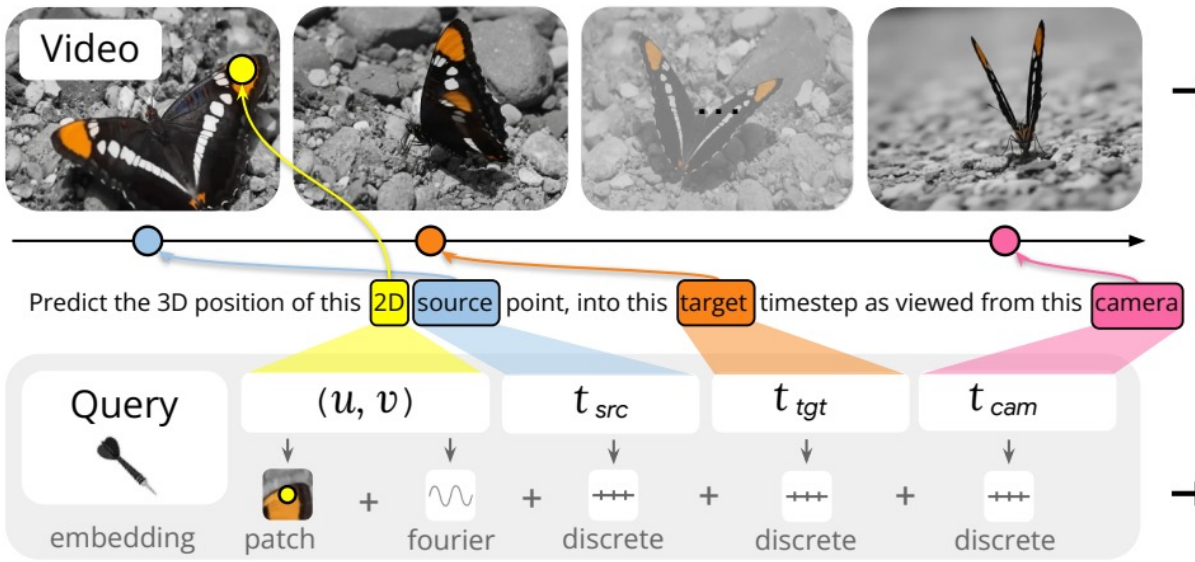
Instead of predicting all outputs with task-specific dense decoders, D4RT asks:

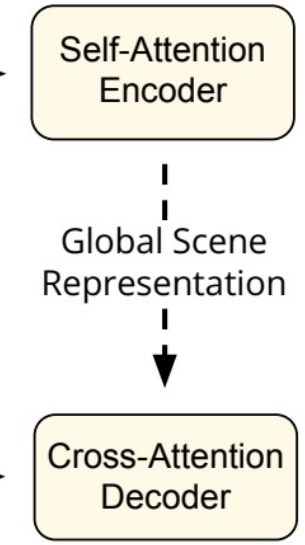
Where is a source pixel located in 3D at a target time, expressed in a selected camera coordinate system?

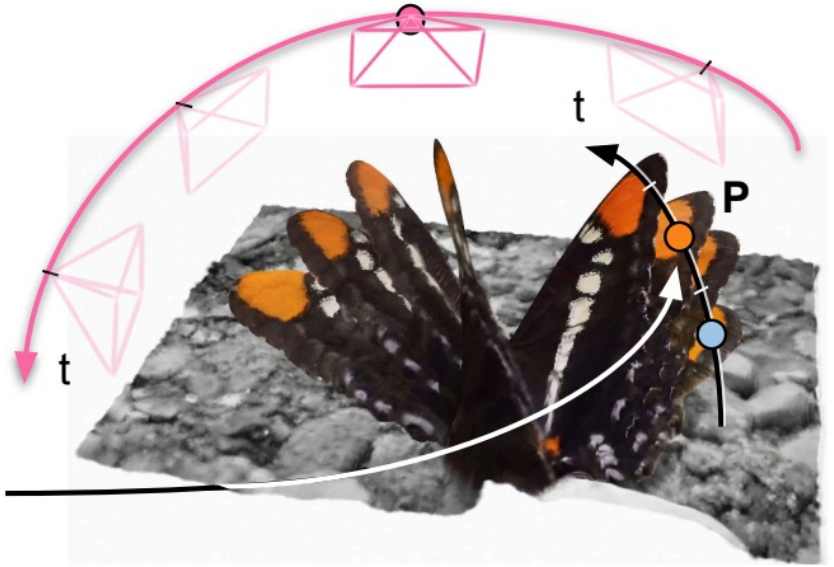
Query-based reconstruction:

$$q = (u, v, t_{src}, t_{tgt}, t_{cam})$$

$$P = D(q, F) \in \mathbb{R}^3$$







Core Idea

Multi-task with the query interface

Task	Query				
	u	v	t_{src}	t_{tgt}	t_{cam}
Point Track	Fixed	Fixed	Fixed	$1 \dots T$	
Point Cloud	$1 \dots W$	$1 \dots H$	$1 \dots T$		Fixed
Depth Map	$1 \dots W$	$1 \dots H$	$1 \dots T$		
Extrinsics	$1 \dots h$	$1 \dots w$	Fixed		$1 \dots T$
Intrinsics	$1 \dots h$	$1 \dots w$	$1 \dots T$		

Table 1. **Unified decoding** – A diverse set of geometry-related tasks can be inferred by querying the Cartesian product of the respective entries. Note that for intrinsics and extrinsics, we only query a coarse (h, w) grid for faster inference.

Methods

Overall Encoder-decoder Structure

ViT-based encoder:

Scene Representation Feature

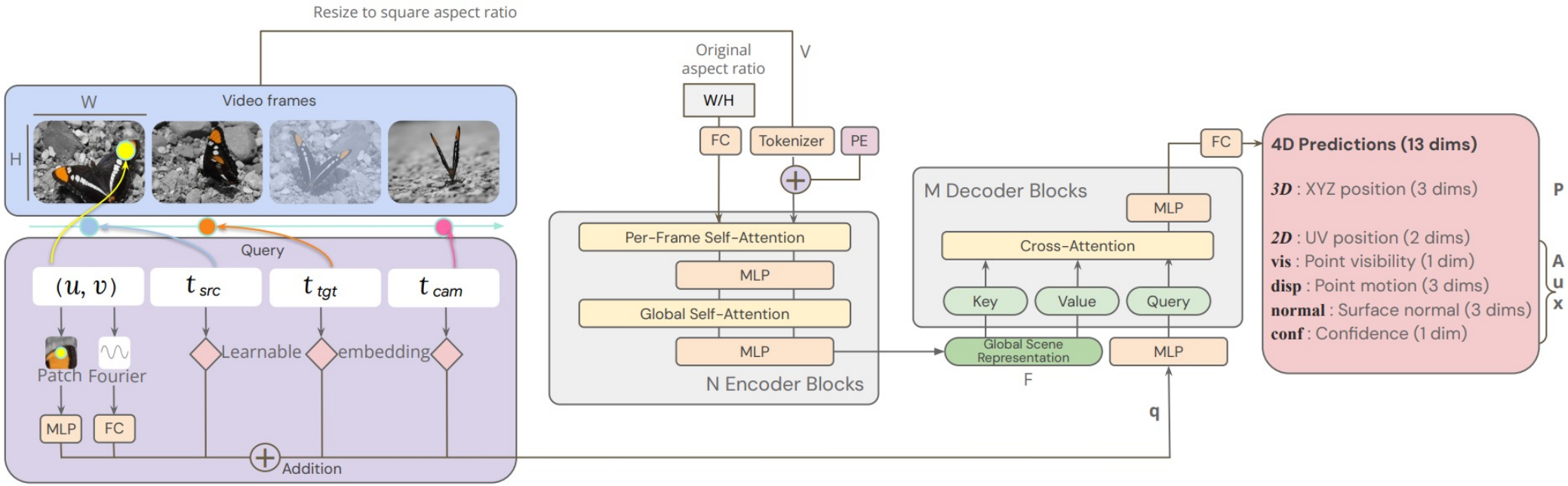
$$F = \mathcal{E}(V) \in \mathbb{R}^{N \times C}$$

Point-wise decoder:

Cross-attn based decoder

Each query is decoded independently

$$\mathbf{P} = \mathcal{D}(\mathbf{q}, F) \in \mathbb{R}^3.$$



Methods

Query-formulation

$$q = (u, v, t\_src, t\_tgt, t\_cam)$$

(u,v): normalized pixel coordinates in the source frame
t_src: source time defining the physical point
t_tgt: target time defining the point state
t_cam: camera frame defining the output coordinate system

Which physical point?

At what time?

In which coordinate system?

Query-Embedding

$$e\_q = e\_uv + e\_src + e\_tgt + e\_cam + e\_patch$$

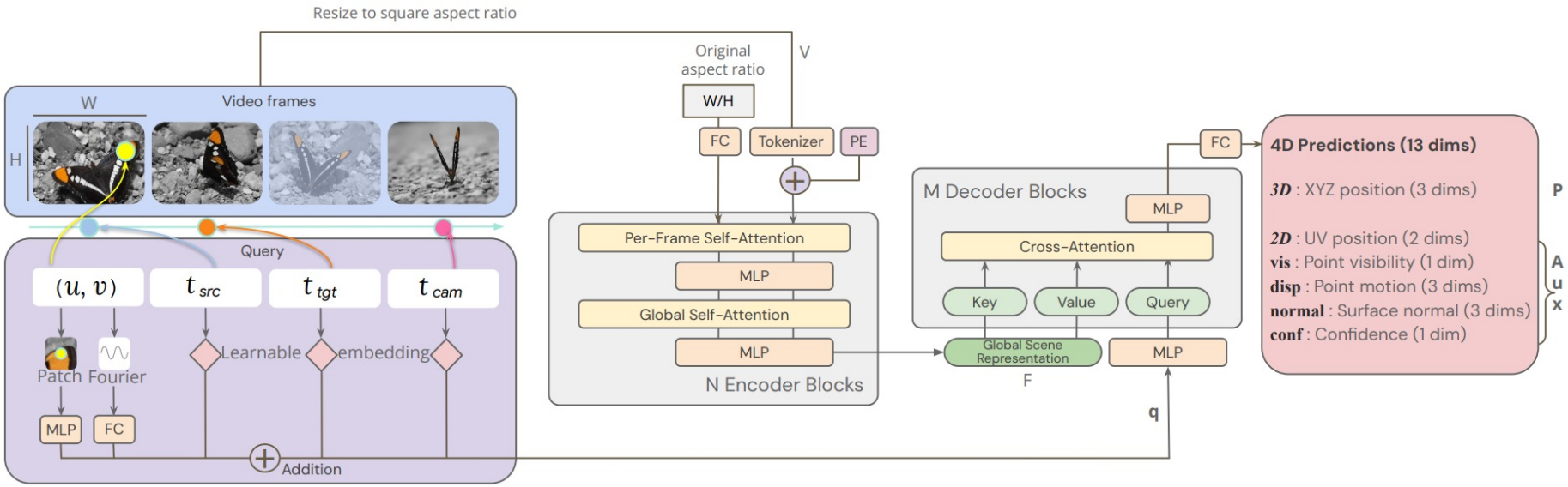
e_uv: Fourier feature embedding of (u,v)
e_src: Learnable discrete embedding
e_tgt: Learnable discrete embedding
e_cam: Learnable discrete embedding
e_patch: Embedding encoded with 9x9 pixel patch by MLP(maybe)

Methods

Point-wise Decoder

(P_3d, P_2d, n, d, v, c) = CrossAttn(q, F)

P_3d: target 3D point
P_2d: target 2D point
n: local surface normal
d: 3D point motion
v: whether the point is visible
c: prediction confidence



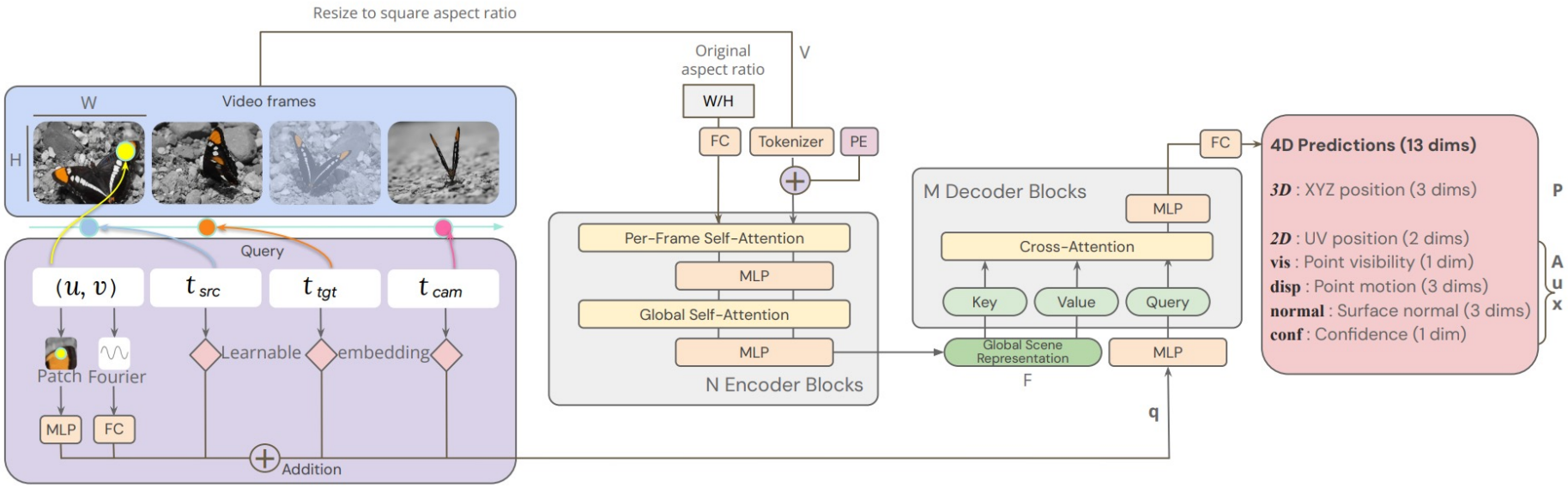
Methods

Training Objective

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(c\lambda_{3D}\mathcal{L}_{3D} - \lambda_{\text{conf}}\log c + \lambda_{2D}\mathcal{L}_{2D} + \lambda_{\text{vis}}\mathcal{L}_{\text{vis}} + \lambda_{\text{disp}}\mathcal{L}_{\text{disp}} + \lambda_{\text{conf}}\mathcal{L}_{\text{conf}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}}\right)_i$$

Query sampling

Only sample 2048 queries each clip with 48 frames



Experiments

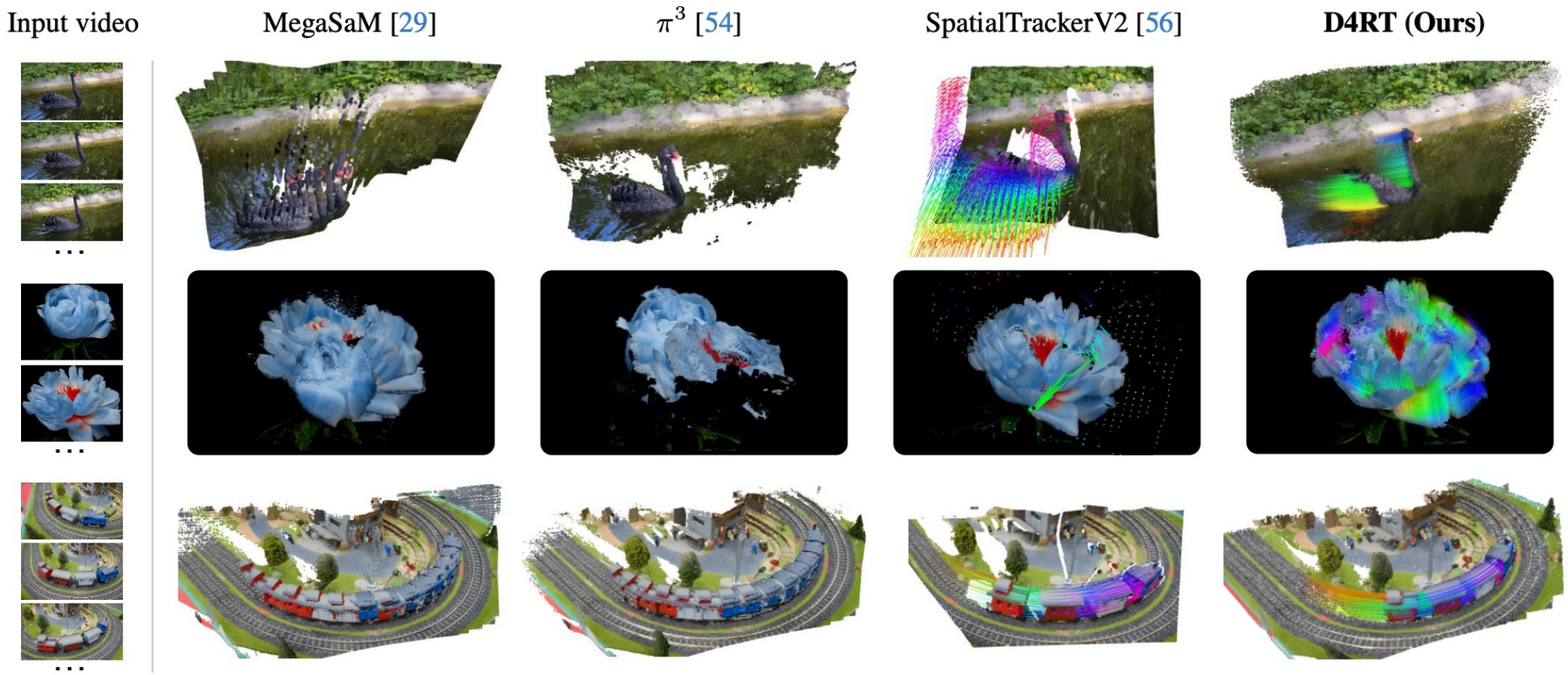


Figure 4. **Reconstruction results across methods** – Pure reconstruction methods (MegaSaM and π^3) are only able to accumulate point clouds of all pixels; exhibiting clear failure cases in dynamic scenes. For example, the swan is repeated in MegaSaM's reconstruction, and π^3 is failing entirely to reconstruct the flower. SpatialTrackerV2, a state-of-the-art tracking method, successfully captures dynamics, however its design only allows tracking points from *one* frame, leaving gaps in the reconstruction (behind the swan and train). D4RT is the only method that successfully reconstructs a full 4D representation of the scene including *all* pixels of the video.

Experiments

4D reconstruction and tracking

w/ GT intrin.	Method	TAPVid-3D											
		Camera Coordinate 3D tracking									World Coordinate 3D tracking		
		DriveTrack			ADT			PStudio			DriveTrack		ADT	
		AJ ↑	APD_{3D} ↑	OA ↑	AJ ↑	APD_{3D} ↑	OA ↑	AJ ↑	APD_{3D} ↑	OA ↑	APD_{3D} ↑	L1 ↓	APD_{3D} ↑	L1 ↓
✗	St4RTrack [11]	-	-	-	-	-	-	-	-	-	0.020	0.499	0.062	0.839
	CoTracker3 [23] + UniDepthV2 [36]	0.038	0.062	0.856	0.102	0.146	0.937	0.119	0.176	0.862	-	-	-	-
	CoTracker3 [23] + VGGT [48]	0.129	0.189	0.856	0.132	0.190	0.937	0.045	0.070	0.862	0.205	0.170	0.192	0.736
	SpatialTrackerV2 [56]	0.064	0.100	0.865	0.260	0.342	0.936	0.097	0.152	0.805	0.101	0.139	0.336	0.238
	D4RT (Ours)	0.257	0.345	0.875	0.240	0.319	0.926	0.138	0.186	0.897	0.373	0.020	0.319	0.096
✓	CoTracker3 [23] + UniDepthV2 [36]	0.147	0.225	0.856	0.173	0.254	0.937	0.205	0.311	0.862	-	-	-	-
	CoTracker3 [23] + VGGT [48]	0.245	0.342	0.856	0.175	0.250	0.937	0.215	0.318	0.862	0.292	0.160	0.234	0.755
	DELTA [32]	0.170	0.238	0.874	0.199	0.258	0.879	0.175	0.261	0.760	-	-	-	-
	SpatialTrackerV2 [56]	0.195	0.275	0.865	0.303	0.404	0.936	0.175	0.270	0.805	0.201	0.117	0.378	0.263
	D4RT (Ours)	0.304	0.410	0.875	0.307	0.408	0.926	0.372	0.498	0.897	0.470	0.017	0.398	0.093

Table 4. **4D reconstruction and tracking** – We evaluate 3D tracking capability on dynamic videos, with tracks predicted in both local camera coordinates (left) and world coordinates (right). Our model achieves superior performance compared to the prior state-of-the-art.

Experiments

Video depth and point cloud estimation

Method	Point Cloud		Video Depth							
	Sintel	Scannet	Sintel		ScanNet		KITTI		Bonn	
	L1 ↓	L1 ↓	AbsRel (S) ↓	AbsRel (SS) ↓	AbsRel (S) ↓	AbsRel (SS) ↓	AbsRel (S) ↓	AbsRel (SS) ↓	AbsRel (S) ↓	AbsRel (SS) ↓
MegaSaM [29]	1.531	0.072	0.342	0.249	0.050	0.047	0.109	0.101	0.056	0.056
VGGT [48]	1.582	0.063	0.318	0.247	0.044	0.040	0.094	0.067	0.055	0.051
MapAnything [24]	1.718	0.064	0.397	0.306	0.043	0.035	0.096	0.090	0.076	0.049
SpatialTrackerV2 [56]	1.375	0.036	0.209	0.175	0.027	0.025	0.075	0.064	0.042	0.978
π^3 [54]	1.139	0.030	0.241	0.163	0.021	0.019	0.055	0.053	0.033	0.032
D4RT (Ours)	0.768	0.028	0.171	0.148	0.020	0.018	0.055	0.051	0.036	0.036

Table 5. **Video depth and point map estimation** – Quantitative results for both video depth and point map estimation across four benchmarks. D4RT achieves top-tier performance on the depth estimation task under both scale-only (S) and scale-and-shift (SS) alignments.

Camera pose estimation

Method	Sintel			ScanNet			Re10K
	ATE ↓	RPE-T ↓	RPE-R ↓	ATE ↓	RPE-T ↓	RPE-R ↓	Pose AUC ↑
MegaSaM [29]	0.074	0.030	0.126	0.029	0.016	0.839	71.0
VGGT [48]	0.168	0.056	0.428	0.016	0.012	0.316	70.2
MapAnything [24]	0.202	0.089	2.383	0.023	0.016	0.438	68.7
SpatialTrackerV2 [56]	0.126	0.053	1.052	0.018	0.012	0.324	75.7
π^3 [54]	0.086	0.039	0.248	0.015	0.010	0.291	78.7
D4RT (Ours)	0.065	0.024	0.126	0.014	0.010	0.302	83.5

Table 6. **Camera pose estimation** – We evaluate D4RT against state-of-the-art methods on static indoor scenes (ScanNet, Re10K) and dynamic outdoor scenes (Sintel).

Experiments

Ablation-local rgb patch

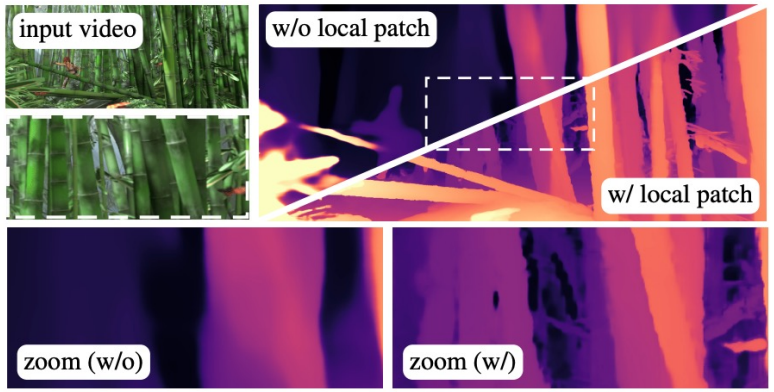


Figure 6. **Preservation of low-level details** – We ablate the effect of including local patch information into the model's queries, visualizing the resulting depth maps on an example from the Sintel dataset. We find that the local RGB patches help preserve fine-grained details and produce sharper object boundaries.

w/ local appear. patch	Video Depth		Camera Pose		
	AbsRel (S) ↓	AbsRel (SS) ↓	ATE ↓	RPE-T ↓	RPE-R ↓
✗	0.366	0.306	0.173	0.031	0.262
✓	0.302	0.257	0.091	0.028	0.245

Table 7. **Local RGB patch** – We evaluate a ViT-L model, trained with and without the appearance patch as input to the decoder, on Sintel video depth and camera pose estimation.

Ablation-Auxiliary loss

Losses	Video Depth		Camera Pose		
	AbsRel(S) ↓	AbsRel(SS) ↓	ATE ↓	RPE-T ↓	RPE-R ↓
D4RT	0.302	0.257	0.091	0.028	0.245
w/o 2D position	+0.071	+0.063	+0.002	+0.002	-0.028
w/o normal	+0.043	+0.026	+0.003	+0.003	-0.022
w/o displacement	+0.011	+0.007	+0.011	+0.003	+0.007
w/o visibility	-0.003	-0.025	+0.012	+0.011	+0.022
w/o confidence	+0.002	-0.025	+0.126	+0.061	+0.115

Table 8. **Auxiliary losses** – We remove auxiliary losses individually to evaluate their impact on model performance. A slight trade-off between depth and camera estimation pose is observed, though we find that all auxiliary losses improve overall performance.