# CoWTracker: Tracking by Warping instead of Correlation

Zihang Lai[1,2] Eldar Insafutdinov[1] Edgar Sucar[1] Andrea Vedaldi[1,2]
[1]Visual Geometry Group, University of Oxford [2]Meta AI

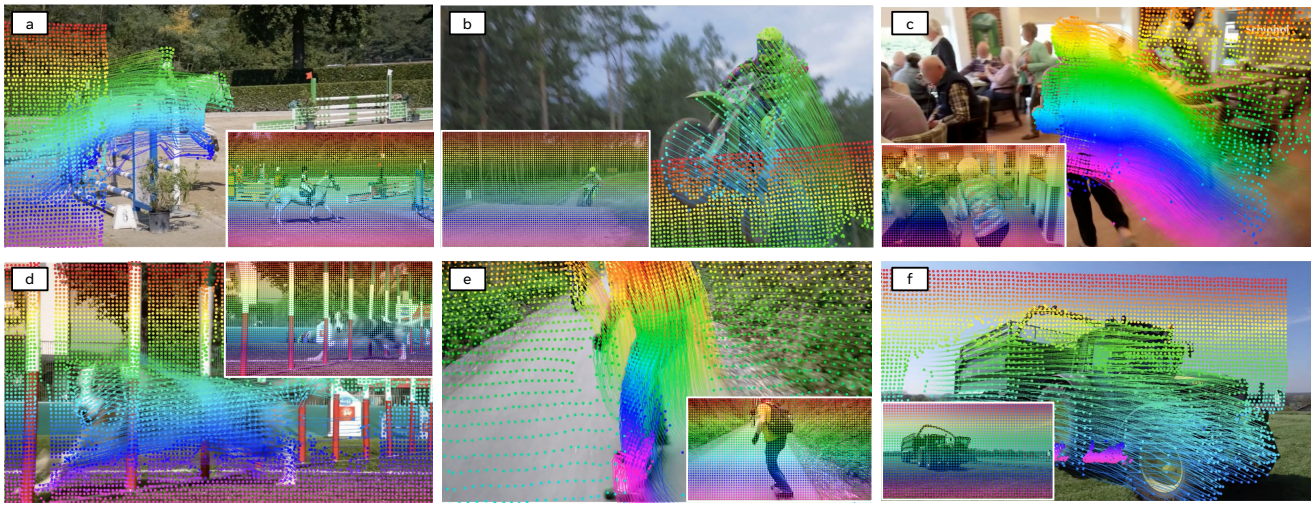


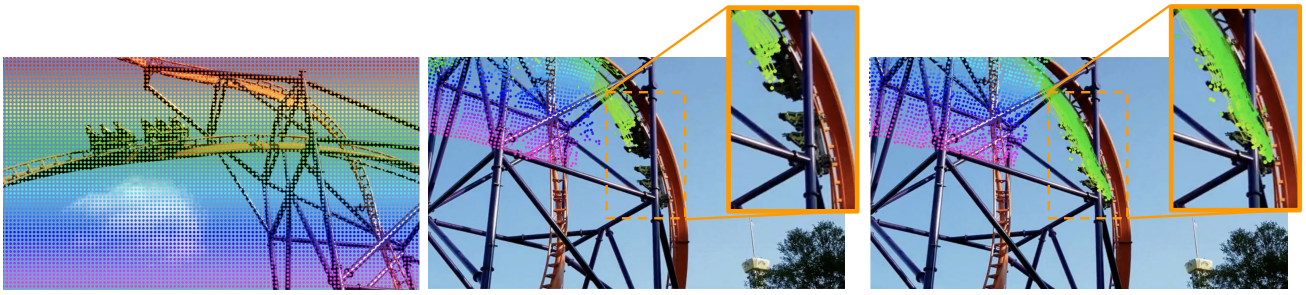

Figure 1. **CoWTracker results.** Our tracker produces dense, long-range point tracks in diverse real-world scenes. It reliably follows humans undergoing rapid motion **(a,b,c,e)**, animals under repeated occlusions and background clutter **(d)**, and vehicles in challenging outdoor settings **(b,f)**. Corner images show the query points from frame 0. Results are subsampled by a factor of 8 (showing only 1/64 of the predicted points).

## Abstract

*Dense point tracking is a fundamental problem in computer vision, with applications ranging from video analysis to robotic manipulation. State-of-the-art trackers typically rely on cost volumes to match features across frames, but this approach incurs quadratic complexity in spatial resolution, limiting scalability and efficiency. In this paper, we propose CoWTracker, a novel dense point tracker that eschews cost volumes in favor of warping. Inspired by recent advances in optical flow, our approach iteratively refines track estimates by warping features from the target frame to the query frame based on the current estimate. Combined with a transformer architecture that performs joint spatiotemporal reasoning across all tracks, our design establishes long-range correspondences without computing feature correlations. Our model is simple and achieves state-of-the-art performance on standard dense point tracking benchmarks, including TAP-Vid-DAVIS, TAP-Vid-Kinetics, and Robo-TAP. Remarkably, the model also excels at optical flow, sometimes outperforming specialized methods on the Sintel, KITTI, and Spring benchmarks. These results suggest that warping-based architectures can unify dense point tracking and optical flow estimation. Project website:* `cowtracker.github.io`.

## 1. Introduction

Point tracking [34] is as old as computer vision and yet remains an important tool in video analysis today. While early trackers were based on first principles and handcrafted designs, deep learning has transformed the field, starting with the introduction of correlation filters [38]. Most recent progress in point tracking has focused on the *track-any-point* (TAP) formulation of [7, 13], where a neural network is tasked with tracking arbitrary points in videos. The PIPs model introduced by [13] has had a strong influence on subsequent work. A key aspect of its design is the use of a transformer network to reason about tracks over time, treating each point track as a token sequence. Another key aspect, borrowed from the optical flow literature, is the use of cost volumes [10, 35, 37], which measure similarities



(a) Grid of query points (8x subsampled) (b) AllTracker (c) WarpTracker

Figure 2. **Dense point tracking on a challenging roller-coaster scene.** (a) Initial grid of query points (8× subsampled). (b) AllTracker [14] struggles with the thin strcutures, large viewpoint changes, and occlusions, and fails to track the front half of the coaster (see zoom-in). (c) CoWTracker accurately follows the front segment and maintains accurate tracks along the coaster tracks, even near boundaries and through occlusions.

between image features across frames and facilitate point matching.

Correlating features and computing cost volumes are staples of optical flow and tracking. Even so, they can be a bottleneck: matching each feature in one image to each feature in the other scales quadratically with image resolution. Recently, however, some optical flow works have questioned this design choice. For example, [20] shows that an optical flow predictor can gradually reduce its dependence on cost volumes during training, discarding them at test time. More relevant to our work, WAFT [43] shows that cost volumes can be replaced by a warping-based mechanism that is simpler and more efficient.

In this paper, we ask whether these benefits can be transferred from optical flow to dense point tracking. To answer this question, we propose CoWTracker, a novel model for dense point tracking that dispenses with cost volumes and instead uses a warping mechanism similar to WAFT.

WAFT borrows from early optical flow approaches such as [2, 25, 27], and is also related to the Spatial Transformer [16].[1] In a nutshell, WAFT iteratively refines the correspondences between two images. At each iteration, dense features in the second (target) image are warped back to the first (source) image based on the current estimate of the forward flow. After warping the target features, they are concatenated channel-wise with the corresponding source features and processed by a network to update the flow estimate. Unlike cost volumes, which explicitly compare each source location to a neighborhood of candidate locations in the target image, this approach evaluates only a single pairing, as specified by the current flow estimate.

At first glance, this design may seem unlikely to establish matches effectively, since each source location is paired with only a single target location at each iteration. However, another insight from WAFT is that, if the paired features are then processed by a transformer via self-attention, the model can still reason globally about correspondences.

Beyond optical flow, the idea of processing matches globally via self-attention was also proposed in tracking, for example, by CoTracker [18]. Building further on this observation, we draw a connection between joint tracking and warp-based matching. In fact, joint tracking in CoTracker is obtained by introducing self-attention layers that operate across tracks, which is conceptually equivalent to the self-attention layers used in WAFT to reason globally about correspondences.

Building on this connection, we propose a tracking-by-warping technique. We estimate tracks densely for every pixel in a reference image. Based on the current track estimates, we warp features from *all* other frames to the reference frame. Then, we use a transformer with separate self-attention along the spatial and temporal dimensions to refine the tracks. This is inspired by the design of PIPs and CoTracker, and it also subsumes the spatial attention in WAFT.

The resulting design is extremely simple yet effective. When combined with powerful feature encoders—especially VGGT [42]—CoWTracker achieves state-of-the-art performance on several dense point tracking benchmarks, including TAP-Vid-DAVIS [7], TAP-Vid-Kinetics [13], and Robo-TAP [39]. Moreover, the *same model* also performs strongly on optical flow, sometimes outperforming the state of the art on benchmarks such as Sintel [3], KITTI [12], and Spring [26].

In summary, our contributions are: (i) we show that warp-based matching works well for optical flow and tracking, and leverage this observation to build CoWTracker, a unified model that solves both while eschewing cost volume computation; (ii) we demonstrate that CoWTracker achieves state-of-the-art performance on several dense point tracking benchmarks; and (iii) we show that the *same* model is also highly competitive for optical flow, sometimes outperforming the state of the art on standard benchmarks for this task. We believe this design can serve as a stepping stone for future research in optical flow, point tracking, and other matching tasks.

[1] Not to be confused with the unrelated concept of transformer networks.

## 2. Related work

While 2D tracking has a long history in computer vision [34], we focus on the so-called *tracking any point* (TAP) problem, popularized by recent works [7, 13] and most relevant to our setting. The former also revisits Particle Video [32] using deep learning; the resulting PIPs tracker introduces several design elements that influenced numerous follow-ups and extensions. PIPs, inspired by the optical flow method RAFT [37], computes correlation maps between image features (a *correlation volume*) and refines track estimates using a transformer network.

TAP-Vid [7] refines the TAP problem statement, proposes multiple benchmarks, and introduces TAP-Net, a lightweight model for point tracking. TAPIR [8] fuses TAP-Net-style global matching with PIPs-style local refinement, yielding substantial accuracy gains. Zheng et al. [48] contribute the PointOdyssey synthetic benchmark and present PIPs++, an improved PIPs variant for longer-term tracking.

CoTracker [18, 19] exploits correlations between multiple simultaneous tracks to improve robustness under occlusion and out-of-frame motion; CoTracker3 simplifies the architecture and proposes a data-efficient self-training regime. DOT [22] builds on CoTracker by densifying its outputs, while VGGSfM [41] adopts a coarse-to-fine design that validates tracks via 3D reconstruction (but targets static scenes only). BootsTAPIR [9] further improves TAPIR through large-scale self-training on millions of videos.

Inspired by DETR [4], TAPTR [24] formulates point tracking as an end-to-end transformer, where points are decoder queries. TAPTRv2 [23] introduces a deformable-attention mechanism that eliminates cost volumes for sparse point tracking, but is difficult to extend to dense tracking because each query outputs only a single displacement. LocoTrack [6] extends 2D correlation features to 4D correlation volumes, simplifying the pipeline and improving efficiency. TAPVid-3D [21] is a 3D-aware variant of TAP-Vid that incorporates reconstruction or multi-view cues. TAPIP3D [47] further extends TAP methods to 3D/depth-aware tracking. TAPNext [49] focuses on improved matching and temporal modeling for higher accuracy and speed. ReTracker [36] boosts accuracy via pretraining on large-scale image-matching datasets. DELTA [29] develops dense embedding and association learning for long-term robustness. SceneTracker [40] leverages scene-level geometry to validate and recover tracks, and AllTracker [14] proposes a unified model for tracking all points or objects in a video. In this work, we depart from the cost-volume-based paradigm adopted by most prior TAP approaches, opting for a purely warping-based tracker that is simple, efficient, and accurate.

## 3. Method

In this section, we introduce CoWTracker, a new warping-based dense tracker. The full model, shown in Fig. 3, consists of three parts: (1) a backbone that produces low-resolution features for each frame, (2) a DPT upsampler that lifts those features to the target (higher) resolution for tracking, and (3) a tracker that computes dense tracks by warping.

### 3.1. Problem Definition

Let a video be $\{I_t\}_{t=0}^{T}$, consisting of a sequence of $T+1$ images $I_t \in \mathbb{R}^{3\times H\times W}$, $t \in \mathcal{T} = \{0, 1, \ldots, T\}$. $I_0$ is the *query frame* and $I_1, \ldots, I_T$ are the *target frames*. Let $p \in \mathbb{R}^2$ denote a 2D pixel location in image coordinates and let $\mathcal{P} \subset \mathbb{R}^2$ be a subset of locations that we use as *queries*. The goal of the tracker is to determine the location $x_t(p) \in \mathbb{R}^2$ of each query pixel $p \in \mathcal{P}$ in each target frame $I_t$, $t = 1, \ldots, T$, under the assumption that the track starts at the query locations at time $t = 0$, i.e., $x_0(p) = p$.

For convenience, we express tracks in terms of the displacement field $u : \mathcal{T} \times \mathcal{P} \to \mathbb{R}^2$, defined by the equation

$$x_t(p) = p + u_t(p), \quad p \in \mathcal{P}, \quad t \in \mathcal{T}. \tag{1}$$

We additionally predict a visibility probability $v_t(p) \in [0, 1]$ indicating whether the point is observable in $I_t$, and a confidence score $\tau_t(p) \in [0, 1]$.

### 3.2. CoWTracker Overview

Unlike previous trackers in the vein of PIPs and follow-up works, our architecture eschews the calculation of cost volumes, utilizing a warp-based formulation instead.

The tracker takes the input video $I$ and extracts dense features $F$ using an off-the-shelf feature extractor and an upsampling layer to obtain a spatial resolution sufficient for accurate tracking. The tracker $\Phi$ then starts by assuming a null displacement field $u = \mathbf{0}$ (corresponding to assuming that points are stationary) and iteratively updates it, captured by the recurrence $u' = \Phi(u \mid F)$. Key to the tracker is a warping operation that, given the current estimate of the displacement field $u$, samples features from each target frame at locations specified by $u$ and brings them into correspondence with the query frame. This results in a *warped feature field* $G = \mathcal{W}(F, u, p)$, from which the updated displacement field is computed. The latter uses a transformer network which, crucially, alternates spatial and temporal attention layers. As noted in CoTracker, tracks are highly correlated. This correlation is crucial to compensating for the fact that there is no cost volume to explicitly search for matches [43].

See the Appendix for pseudocode describing this process. Next, we explain each step in detail.

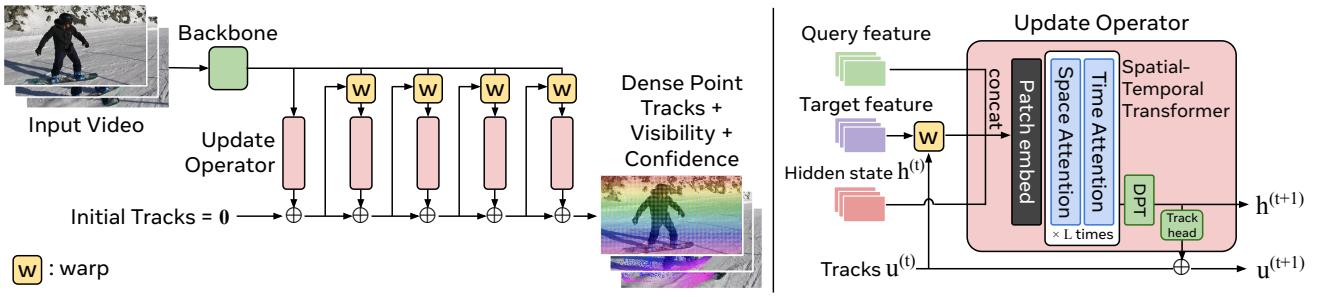



Figure 3. **Left: CoWTracker Pipeline.** The backbone extracts video features from the input video, and a lightweight update operator (see right for details) iteratively warps and refines tracks to yield dense trajectories, visibility, and confidence. **Right: Update operator.** Warped query/target features, hidden states, and current track estimates are fused by a spatial-temporal transformer to predict residual motion and update hidden states.

### 3.3. Features

**Feature Extractor.** To compute dense features from the input video $I$, we use a network $\Psi$ for dense feature extraction. In practice, we consider strong pretrained models like VGGT. Because the feature extractor can process all frames jointly (e.g., VGGT), we write this as a map $F = \Psi(I)$ assigning the tensor $I \in \mathbb{R}^{T\times 3\times H\times W}$ of all video frames to a tensor $F \in \mathbb{R}^{T\times H'\times W'\times C}$ of corresponding features with (typically) lower spatial resolution $H' \times W'$ and $C_b$ channels. The resolution is determined by the stride or patch size $s$ of the feature extractor (typically 14 or 16), so that $\hat{H} = \lfloor H/s \rfloor$ and $\hat{W} = \lfloor W/s \rfloor$.

Unless otherwise noted, we freeze early stages and optionally fine-tune the last blocks, and do not otherwise change the architecture.

**Upsampler.** Methods that use cost volumes typically operate at low spatial resolution to keep memory consumption manageable. In contrast, our warping-only design can efficiently handle higher spatial resolution, as it does not require computing and storing large cost volumes. We therefore upsample the features $\hat{F}$ to a higher spatial resolution $H' \times W'$ using a DPT upsampler [31], resulting in features $F = \mathrm{DPT}(\Psi(I))$ with stride $s' = 2 \ll s$. This uses skip connections from the backbone and a lightweight convolutional decoder. Following WAFT, we also use a small U-Net that directly takes raw images as input and concatenates its output with the upsampled backbone features.

Bringing the resolution of the features close to the input video resolution improves tracking of thin structures and near boundaries. No additional correlation tensor or pyramid is constructed.

### 3.4. Warping-only Tracker

The high-resolution feature maps produced by the backbone and DPT upsampler are consumed by our *warping-only* tracker to compute dense tracks from the query frame to every target frame. Figure 3 depicts one iteration of the update operator inside the tracker head.

The tracker takes as input the feature tensor $F$ computed in Sec. 3.3. It also maintains two states. The first is the current estimate of the displacement fields $u^{(k)} : \mathcal{T} \times \mathcal{P} \to \mathbb{R}^2$, i.e., a $|\mathcal{T}| \times |\mathcal{P}| \times 2$ tensor. The second is a *hidden state* $h^{(k)} : \mathcal{T} \times \mathcal{P} \to \mathbb{R}^{D_h}$, where $D_h$ is the hidden dimension.

The index $k \in \{0, 1, \ldots, K\}$ denotes the update iteration. As noted above, we initialize the displacement field by setting it to zero, i.e., $u^{(0)} = 0$. The hidden state $h^{(0)}$ is initialized from the features $F$ by setting $h_t^{(0)} = \phi(F_0 \oplus F_t)$, where $F_0 \oplus F_t$ is the channel-wise concatenation of the features for frames 0 and $t$, and $\phi : \mathbb{R}^{2C} \to \mathbb{R}^{D_h}$ is a small network implemented as a $1 \times 1$ convolution followed by layer normalization to reduce the number of channels from $2C$ to $D_h$.

The model incrementally *updates* $u$ and $h$ via $K$ iterations of a learned update operator (Fig. 3):

$$(u^{(k+1)}, h^{(k+1)}) = \Phi(u^{(k)}, h^{(k)} \mid F), \quad k = 0, \ldots, K-1.$$

After $K$ iterations, the final displacement field $u = u^{(K)}$ is obtained. Track visibility and confidence are output by readout heads on the final hidden state $h^{(K)}$ via linear layers followed by a sigmoid:

$$v_t = \sigma\left(h^{(K)} W_v\right), \quad \tau_t = \sigma\left(h^{(K)} W_\tau\right).$$

Next, we describe the design of the update operator $\Phi$. This begins by *warping* the features $F$ based on the current displacement estimate $G = \mathcal{W}(F, u, p)$, given by

$$G_t(p) = \mathrm{sample}(F_t,\, p + u_t(p)). \tag{2}$$

Here, $\mathrm{sample}(\cdot)$ denotes bilinear sampling. This aligns the feature vectors in each target frame $t$ to the query frame at location $p$ at time 0. Thus, $G$ is a $|\mathcal{T}| \times |\mathcal{P}| \times C$ tensor of features aligned across time.

The $k$-th update takes the warped features $G^{(k)} = \mathcal{W}(F, u^{(k)}, p)$, the query features $F_0$, the current displacement $u^{(k)}$ and hidden state $h^{(k)}$, and concatenates them into a tensor $z \in \mathbb{R}^{|\mathcal{T}|\times H'\times W'\times(2C+D_h+2)}$ given by:

$$z_t = G_t{}^{(k)} \oplus F_0 \oplus u_t^{(k)} \oplus h_t^{(k)}, \quad t \in \mathcal{T}.$$

| Method | Train data | DAVIS | | | RGB-S | | | RoboTAP | | | Kinetics | | | Mean | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AJ↑ | $\delta_{avg}$ ↑ | OA↑ | AJ↑ | $\delta_{avg}$ ↑ | OA↑ | AJ↑ | $\delta_{avg}$ ↑ | OA↑ | AJ↑ | $\delta_{avg}$ ↑ | OA↑ | AJ↑ | $\delta_{avg}$ ↑ | OA↑ |
| *Sparse Trackers* | | | | | | | | | | | | | | | | |
| PIPs++ [48] | PO | — | 73.7 | — | — | 58.5 | — | — | 63.5 | — | — | 63.5 | — | — | 64.8 | — |
| TAPIR [8] | Kub | 56.2 | 70.0 | 86.5 | 55.5 | 69.7 | 88.0 | 49.6 | 64.2 | 85.0 | 49.6 | 64.2 | 85.0 | 52.7 | 67.0 | 86.1 |
| CoTracker [18] | Kub | 61.8 | 76.1 | 88.3 | 67.4 | 78.9 | 85.2 | 49.6 | 64.3 | 83.3 | 49.6 | 64.3 | 83.3 | 57.1 | 70.9 | 85.0 |
| TAPTR [24] | Kub | 63.0 | 76.1 | 91.1 | 60.8 | 76.2 | 87.0 | 49.0 | 64.4 | 85.2 | 49.0 | 64.4 | 85.2 | 55.5 | 70.3 | 87.1 |
| LocoTrack [6] | Kub | 62.9 | 75.3 | 87.2 | 69.7 | 83.2 | 89.5 | 52.9 | 66.8 | 85.3 | 52.9 | 66.8 | 85.3 | 59.6 | 73.0 | 86.8 |
| BootsTAPIR [9] | Kub+15**M** | 61.4 | 73.6 | 88.7 | 70.8 | 83.0 | 89.9 | 54.6 | 68.4 | 86.5 | 54.6 | 68.4 | 86.5 | 60.4 | 73.4 | 87.9 |
| CoTracker3 (onl.) [19] | Kub+15k | 63.8 | 76.3 | 90.2 | 71.7 | 83.6 | 91.1 | 55.8 | 68.5 | 88.3 | 55.8 | 68.5 | 88.3 | 61.8 | 74.2 | 89.5 |
| CoTracker3 (offl.) [19] | Kub+15k | 64.4 | 76.9 | 91.2 | 74.3 | 85.2 | 92.4 | 54.7 | 67.8 | 87.4 | 54.7 | 67.8 | 87.4 | 62.0 | 74.4 | 89.6 |
| *Dense Trackers* | | | | | | | | | | | | | | | | |
| DOT [28] | Kub | 60.1 | 74.5 | 89.0 | 77.1 | 87.7 | 93.3 | — | — | — | 48.4 | 63.8 | 85.2 | — | — | — |
| DELTA [29] | Kub | 60.8 | 75.2 | 87.6 | 72.2 | 83.0 | 91.4 | 60.2 | 73.4 | 85.6 | 51.0 | 66.6 | 84.0 | 61.1 | 74.6 | 87.2 |
| AllTracker [14] | Kub | 61.9 | 75.4 | 87.8 | 80.7 | 90.1 | 91.7 | 70.1 | 81.5 | 92.2 | 59.3 | 71.3 | 89.3 | 68.0 | 79.6 | 90.3 |
| AllTracker [14] | Kub+Mix | 63.3 | 76.3 | 90.1 | 81.1 | 90.0 | 92.8 | 71.9 | **83.4** | 92.8 | 59.1 | 72.3 | 90.3 | 68.9 | 80.5 | 91.5 |
| CoWTracker | Kub | **65.5** | **78.0** | **92.1** | **85.4** | **92.8** | **94.9** | **73.2** | **83.4** | **94.7** | **60.9** | **73.1** | **91.5** | **71.3** | **81.8** | **93.3** |

Table 1. **TAP-Vid [7] (DAVIS, RGB-Stacking, Kinetics) and RoboTAP [39] benchmarks.** We report Average Jaccard (AJ), $\delta_{avg}$, and Occlusion Accuracy (OA; ↑higher is better). Trained only on Kubric data, our dense CoWTracker achieves the best mean performance across all four datasets and all three metrics, outperforming prior dense methods such as AllTracker as well as strong sparse TAP baselines.

The features $z$ are interpreted as a $|\mathcal{T}| \times |\mathcal{P}|$ matrix of tokens organized by time and space. We augment the features with spatial and temporal embeddings and process these tokens using a Vision Transformer (ViT) [11] adapted to video [1], where spatial and temporal attentions interleave: every two spatial self-attention blocks, we insert one temporal attention block. Each spatial self-attention layer runs independently over $\mathcal{P}$ for each fixed temporal slice $t$, and each temporal attention layer runs over $\mathcal{T}$ for each fixed spatial position $p$. The architecture otherwise follows ViT, with MLP blocks, residual connections, and LayerNorm throughout. Finally, a linear head predicts the residual displacement:

$$\Delta u^{(k+1)} = h^{(k+1)} W_u, \quad u^{(k+1)} = u^{(k)} + \Delta u^{(k+1)}. \quad (3)$$

Note that there is no correlation volume computation in the head; instead, the only place where cross-frame features are paired is the warping operator in Eq. (2). Hence, the cost of running the head scales linearly with the number of targets $T$, the feature resolution $|\mathcal{P}|$, and the number of iterations $K$, making it possible to use a relatively large spatial resolution.

# 4. Experiments

We show that CoWTracker demonstrates competitive performance on point tracking (Sec. 4.2) and optical flow (Sec. 4.3), and we ablate our design choices (Sec. 4.4).

## 4.1. Implementation details

**Architecture.** Unless noted otherwise, we use a pretrained VGGT [42] as our backbone and a DPT-style [31] upsampler to lift low-resolution features to high resolution. We freeze the patch-embedding layers in VGGT and finetune the other parts of the backbone, together with the upsampler (initialized from scratch) and the tracker. The tracker uses $K = 5$ iterations and operates on features with stride $s' = 2$.

**Training.** We train CoWTracker on Kubric data with Huber losses for both visible and occluded tracks, with exponentially increasing weights for each iteration, following [19]. Confidence and visibility are supervised using a binary cross-entropy (BCE) loss at each iterative update. The ground-truth confidence is given by an indicator function that checks whether the predicted track falls within 12 pixels of the ground-truth track. We use the AdamW optimizer with learning rate $5 \times 10^{-4}$ and cosine decay, a batch size of 32 videos with up to 16 frames of size $336 \times 560$, and train for 50k iterations. We use random frame rate and random video length for data augmentation. Mixed precision and gradient checkpointing are enabled.

## 4.2. Point Tracking

We evaluate the point-tracking performance of CoWTracker on the TAP-Vid benchmark suite [7]—which includes the Kinetics, DAVIS, and RGB-Stacking subsets—and on RoboTAP [39]. TAP-Vid Kinetics contains 1,144 videos from the Kinetics validation set [5] featuring complex camera motion; the DAVIS subset provides 30 real-world videos from DAVIS [30]; RGB-Stacking consists of synthetic robotic sequences with large texture-less regions; and RoboTAP includes 265 real-world robotic manipulation videos. We report the standard metrics for TAP-Vid: Occlusion Accuracy (OA), $\delta_{avg}$, and Average Jaccard (AJ).

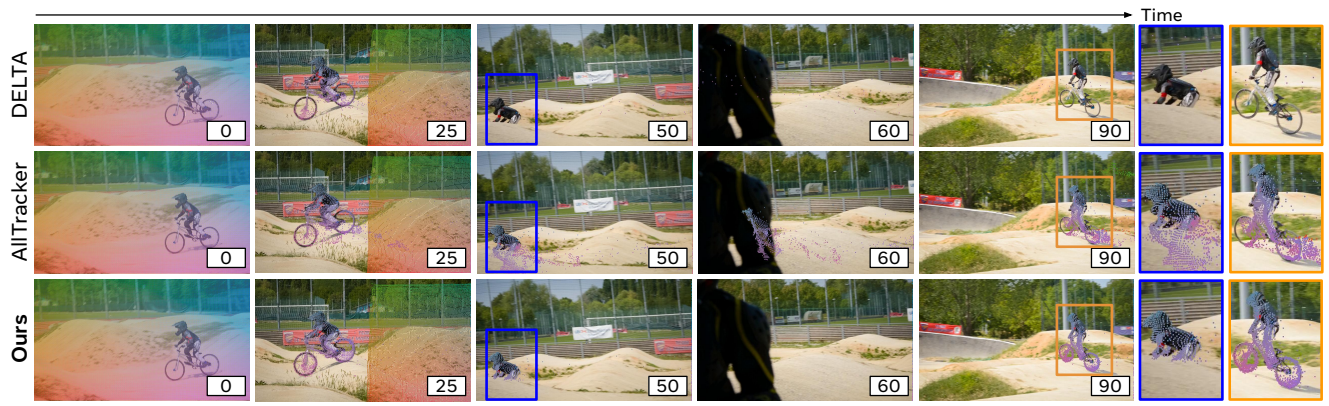



Figure 4. **Tracking through a challenging BMX sequence with a full occlusion in the middle frames.** Rows compare DELTA, AllTracker, and Ours. Our method maintains a consistent track before, during, and after occlusion, whereas DELTA loses the target and fails to recover and AllTracker exhibits noticeable drift and fragmentation. Our warp-based indexing queries features at high resolution, preserving fine details and enabling accurate localization after occlusion. Numbers in lower-right boxes indicate frame numbers.

#### 4.2.1. Quantitative Results

Table 1 reports results on the TAP-Vid benchmark and RoboTAP. Our model surpasses the strongest dense prior, AllTracker, on all four datasets and across all three metrics. Averaged over the datasets, CoWTracker improves AJ by 2.4, $\delta_{\text{avg}}$ by 1.3, and OA by 1.8. Per dataset, the gains are also consistent. These improvements are achieved without cost volumes or multi-resolution pyramids (used in AllTracker), indicating that a warping-only head with a lightweight video-based transformer can enable accurate point tracking.

This gap widens further when training data is controlled: against AllTracker (Kub), the mean improvements of CoWTracker increase to 3.3 / 2.2 / 3.0 in AJ/$\delta_{\text{avg}}$/OA, respectively. We attribute robustness across heterogeneous domains—from *in-the-wild* Kinetics to robotic-centric RGB-Stacking—to the high-resolution pixel alignment enabled by our warping-based architecture.

Beyond point-tracking accuracy, CoWTracker also delivers superior occlusion classification. On average, we improve OA by 3.0 over AllTracker (Kub) and by 1.8 over AllTracker (Kub+Mix), with the largest gap on DAVIS (+4.3). We hypothesize that this advantage arises because our head operates directly on image features rather than compressing appearance into correlation scores when building cost volumes: dot-product similarity can discard channel-wise cues that are predictive of visibility at boundaries, whereas warping-indexed features preserve such information for the occlusion head.

#### 4.2.2. Qualitative Results

Figure 4 visualizes a challenging BMX sequence with a long, full occlusion in the middle frames. We compare DELTA, AllTracker, and our method under the same evaluation protocol. Before the occlusion, all three methods roughly follow the rider; however, as the cyclist passes behind the foreground object, DELTA rapidly loses the target and fails to re-acquire it afterward, leading to large swaths of missing or mislocalized tracks. AllTracker retains partial correspondence but exhibits visible drift and fragmentation: trajectories spread into the background and then snap back with an offset, producing inconsistent motion on the rider's limbs and the bicycle outline. In contrast, our tracker maintains a coherent hypothesis throughout the occlusion and cleanly locks back onto the rider once visibility returns. The re-acquired tracks align tightly with object boundaries, including thin structures such as the handlebar and wheel, and remain stable through the landing.

| | Sintel | | KITTI (train) | | Spring (val) | |
|---|---|---|---|---|---|---|
| **Method** | Clean↓ | Final↓ | EPE↓ | Fl-all↓ | EPE↓ | 1px↓ |
| *Specialized Optical Flow model* | | | | | | |
| RAFT [37] | 1.15 | 1.86 | 1.53 | 7.81 | 0.22 | 1.79 |
| SEA-RAFT [44] (M) | 0.97 | 1.96 | 1.60 | 8.26 | 0.21 | 1.44 |
| WAFT [43] (twins-a2) | 0.94 | 2.09 | 1.15 | 5.29 | **0.11** | 0.74 |
| WAFT [43] (Dv3-a2) | 1.01 | 1.86 | 1.35 | 6.41 | 0.13 | **0.70** |
| *Dense Point Tracker* | | | | | | |
| **CoWTracker** | **0.78** | **1.48** | **1.04** | **4.87** | 0.17 | 0.75 |

Table 2. **Zero-shot optical flow results** on three benchmarks. Our predictions are produced by the *same model used for point tracking*, and the model was *not trained* on any optical-flow datasets, including Sintel. Despite this, CoWTracker shows strong zero-shot transfer from tracking to optical flow.

### 4.3. Optical Flow

We now evaluate how well CoWTracker transfers to optical flow estimation by simply treating a frame pair as a two-frame "video". We consider three standard benchmarks: MPI-Sintel [3], KITTI-2015 [12], and Spring [26], which cover cinematic synthetic sequences with complex non-rigid motion (Sintel), real-world driving scenes with strong camera and background motion (KITTI), and high-


Figure 5. **Optical flow predictions on MPI Sintel** using the *same* model as our point-tracking results. The predicted flows closely match the ground truth even in difficult scenarios—*large motion*, *occlusions*, and *background clutter*. Note that the model was *not trained* on any optical-flow datasets, including Sintel.

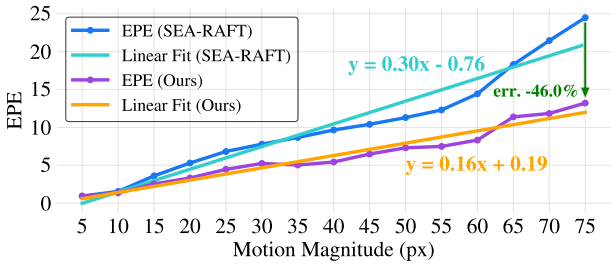



Figure 6. **EPE vs. motion magnitude** for SEA-RAFT and CoWTracker, evaluated on all pixels in Sintel. Our method yields lower error across all motion bins and a flatter increase with motion (slope 0.16 vs. 0.30), with the largest gains at high displacements (46% lower EPE), indicating stronger robustness to large motion.

quality rendered videos with fine structures and realistic motion (Spring). All predictions are produced by the same model used for point tracking, without any additional training on optical-flow data.

For each benchmark, we follow the standard evaluation protocols. All inputs are resized to a resolution of $336 \times 560$. We treat each frame pair as a two-frame video and run CoWTracker with exactly the same configuration as in our video setting, without any flow-specific tuning or changes to hyperparameters. We report End-Point Error (EPE) on all datasets, Fl-all on KITTI-2015, and the 1px error rate on Spring.

#### 4.3.1. Quantitative Results

Table 2 reports zero-shot optical flow accuracy on Sintel, KITTI-2015 (train), and Spring (val), where predictions are produced by the *same* model used for point tracking without any flow-specific training. On Sintel, our warping-only tracker attains 0.78 EPE on the Clean split and 1.48 EPE on the Final split, improving over the next best specialized flow model (0.94/1.86) by 17%/20% relative, respectively. These gains suggest that CoWTracker recovers pixel-accurate motion and preserves thin structures, yielding state-of-the-art zero-shot performance on both Sintel splits.

On KITTI-2015, our model achieves 1.04 EPE and 4.87% Fl-all, outperforming RAFT, SEA-RAFT, and both WAFT variants. Relative to the strongest prior (WAFT twins-a2), this is a 9.6% reduction in EPE and a 7.9% reduction in Fl-all. This suggests an advantage under wide-baseline motions and in texture-poor regions common in driving scenes, where cost-volume methods either lower the feature resolution or restrict the search radius; our warping-based architecture allows the head to index at higher spatial resolution and convert additional input detail directly into accuracy.

On Spring, our tracker remains competitive despite not being trained on flow data, obtaining 0.17 EPE and 0.75% 1px. Specialized WAFT models achieve slightly smaller absolute errors (0.11–0.13 EPE and 0.70–0.74% 1px), but our zero-shot results are within 0.06 EPE and 0.05% 1px of the best entries. We note that Spring operates in a very low-motion regime: the average flow magnitude is only 3.5 pixels, compared to 13.4 pixels on Sintel and 27.8 pixels on KITTI-2015. In this regime, all methods achieve extremely small errors, and the gaps are close to the scale of noise, making it difficult to draw strong conclusions from Spring alone.

**Accuracy vs. motion scales.** To understand how our warping-based head behaves across motion scales, we stratify errors by flow magnitude and report EPE (averaged across all pixels in a flow-magnitude bin) as a function of motion in Fig. 6. Compared to SEA-RAFT, our method not only attains consistently lower EPE across all bins, but also exhibits a substantially flatter linear trend (slope 0.16 vs. 0.30). The relative improvement becomes most pronounced for large displacements: in the highest-motion bin, our EPE is roughly 46% lower than SEA-RAFT, compared to an average improvement of about 22.5%. This analysis suggests that, despite lacking an explicit cost volume, our warping-only design does *not* struggle with large motions; rather, high-resolution local warping appears more robust when two matching features are far apart, whereas correlation volumes over large search regions may produce noisy or ambiguous responses.

#### 4.3.2. Qualitative Results

Figure 5 compares our optical flow predictions with the ground-truth flows. Our method yields smooth, accurate motion with sharp boundaries even in difficult scenarios, including large displacements, occlusions, and cluttered backgrounds.

| Model | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| CoTracker [19] | 62.9 | 77.1 | 68.3 |
| ViT [42] | 75.7 | 89.9 | 80.9 |
| Pi$^3$ [45] | 75.3 | 91.1 | 82.6 |
| VGGT [42] | **78.0** | **92.8** | **83.4** |

(a) **Backbone**: VGGT backbone yields best results across datasets, whereas Pi$^3$ also produce reasonable results.

| Type | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| None | 72.5 | 90.3 | 80.0 |
| Bilinear | 76.8 | 91.9 | 82.0 |
| Loftup [15] | 77.4 | 90.8 | 81.9 |
| DPT [31] | **78.0** | **92.8** | **83.4** |

(b) **Upsampler**: DPT upsampler gives strongest $\delta_{\text{avg}}$, outperforming bilinear and a Loftup-style upsampler for high-resolution indexing.

| Ratio, Patch | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| 1/16, 1x1 | 70.9 | 89.8 | 80.1 |
| 1/8, 1x1 | 74.6 | 91.6 | 82.3 |
| 1/4, 2x2 | 75.0 | 91.6 | 81.8 |
| 1/2, 4x4 | **78.0** | **92.8** | **83.4** |

(c) **Resolution–patch-size tradeoff.** A finer indexing stride (1/2) achieves the best results, outperforming coarser strides.

| Model | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| ViT (img.) | 74.7 | 81.1 | 72.6 |
| Ours (vid.) | **78.0** | **92.8** | **83.4** |
| Δ | +3.3 | +11.7 | +11.2 |

(d) **Tracker network structure**: Adding temporal attention in the head greatly boosts performance on long RGB and RoboTAP videos.

| Iterative? | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| No | 71.4 | 90.0 | 79.4 |
| Yes | **78.0** | **92.8** | **83.4** |
| Δ | +6.6 | +2.8 | +4.0 |

(e) **Iterative refinement**: Multi-step warping updates significantly outperform single-pass refinement on all three benchmarks.

| Head | DAVIS↑ | RGB↑ | Rob.↑ |
|---|---|---|---|
| AllTracker | 72.0 | 89.5 | 80.6 |
| Ours(no warp) | 54.6 | 85.5 | 73.8 |
| Ours | **78.0** | **92.8** | **83.4** |

(f) **Head design**: Our full warping-only head significantly outperforms a non-warping variant and also the AllTracker head.

Table 3. **Ablation study for CoWTracker.** Each table isolates the effect of a specific design choice. Results are reported using $\delta_{\text{avg}}$.

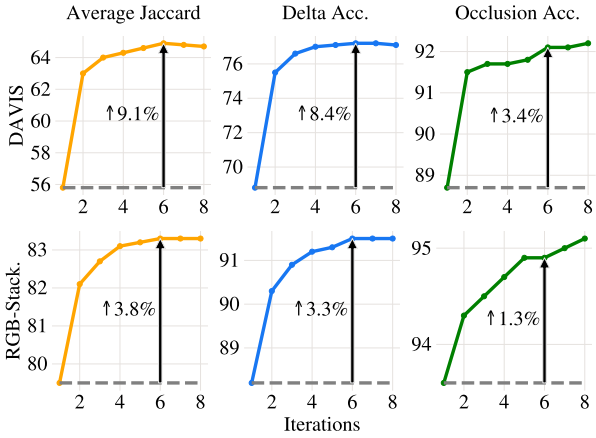



Figure 7. **Ablation on test-time iterations.** We vary the number of refinement steps $K$. Performance improves significantly from $K = 1$ to $K = 2$, then gradually saturates around $K = 5$–6, with OA yielding modest additional gains.

### 4.4. Ablation Studies

In Tab. 3 and Fig. 7, we ablate our main design choices:

**Backbone choice.** Table 3a compares interchangeable backbones under the same warping-only head. The ConvNet-based backbone used by CoTracker performs clearly worse, while ViT (we use the patch embedding network of VGGT) and Pi$^3$ works reasonably well. VGGT achieves the best $\delta_{\text{avg}}$ on all three datasets, improving over Pi$^3$ by +1.7 $\delta_{\text{avg}}$ on average, showing that our head benefits directly from stronger video backbones.

**Upsampling for high-resolution indexing.** Table 3b compares ways to lift low-resolution features for high-resolution warping. Removing the upsampler hurts performance; bilinear upsampling helps; LoftUp-style cross-attention helps slightly more; and the DPT upsampler performs best, with gains of +5.5 $\delta_{\text{avg}}$ on DAVIS and +3.4 $\delta_{\text{avg}}$ on RoboTAP over no upsampler.

**Indexing resolution vs. patch size.** In Table 3c, we vary the indexing resolution (feature stride vs. input) while adjusting the transformer patch size to roughly match compute. Even with compensating patch sizes, higher indexing resolution (1/2 stride) clearly outperforms coarser settings (1/4–1/16), reaching 78.0 $\delta_{\text{avg}}$ on DAVIS and 83.4 on RoboTAP. This supports prioritizing high-resolution indexing.

**Spatial vs. spatio-temporal tracker network.** Table 3d compares a purely spatial image transformer (ViT (img.)) with our spatio-temporal transformer that adds temporal attention. Temporal attention yields large gains on long sequences (RGB-Stacking and RoboTAP, up to 600 frames), improving $\delta_{\text{avg}}$ by +11.7 and +11.2, respectively.

**Iterative refinement.** Table 3e contrasts a single-pass head with our iterative refinement scheme. Multiple refinement steps in the warping head significantly improve performance, by +6.6 $\delta_{\text{avg}}$ on DAVIS and +4.0 $\delta_{\text{avg}}$ on RoboTAP.

**Head design.** Finally, Table 3f evaluates tracker head designs. A non-warping variant that always queries the original features performs dramatically worse ($-23.4$ $\delta_{\text{avg}}$ on DAVIS, $-9.6$ $\delta_{\text{avg}}$ on RoboTAP), highlighting the importance of explicit warping in CoWTracker. CoWTracker also compares favourably to an AllTracker head on the same backbone. We hypothesize that this is because our warping-only head uses backbone features directly and can operate at higher resolution, whereas correlation volumes are expensive and thus confined to coarser grids.

**Refinement steps.** Figure 7 shows that increasing the refinement steps $K$ consistently boosts performance, with the largest gain from $K = 1$ to $K = 2$ and diminishing returns thereafter. Metrics stabilize around $K = 5$, which we adopt as our default.

## 5. Limitations

CoWTracker has several limitations that suggest future research directions. Our method may fail under extreme viewpoint changes, long-range full occlusions, or severe specularities. While the iterative warping head is lightweight, overall throughput is heavily dependent on the chosen backbone, such as VGGT. Nonetheless, the marginal overhead of CoWTracker remains small, and the benefits of higher-resolution feature indexing persist even when using more efficient backbones. Another constraint is the quadratic complexity of the VGGT backbone with respect to video length, which requires processing longer clips in chunks. Furthermore, the current iterative refinement process tends to saturate after five to six steps, indicating a ceiling on the model's self-correction capability. Finally, the model is currently trained exclusively on synthetic Kubric data. Diverse real-world data remains significantly underleveraged; incorporating natural videos could improve robustness to lighting and noise patterns that are difficult to simulate.

## 6. Conclusions

We presented CoWTracker, a dense point tracker that replaces cost volumes with iterative warping-based refinement. This simple, warp-based head scales linearly with the spatial resolution and attains state-of-the-art tracking on TAP-Vid and RoboTAP benchmarks while transferring competitively to optical flow estimation without flow-specific training. We hope these results will encourage revisiting dense correspondence architecture with simple, warp-centric designs that bridge tracking and optical flow.

## References


[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proc. ICCV*, 2021. 5

[2] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*, 2004. 2

[3] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, 2012. 2, 6

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. ECCV*, 2020. 3

[5] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 5

[6] Seokju Cho, Jiahui Huang, Jisu Nam, Honggyu An, Seungryong Kim, and Joon-Young Lee. Local all-pair correspondence for point tracking. *arXiv*, 2407.15420, 2024. 3, 5, 1

[7] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. In *Proc. NeurIPS*, 2022. 1, 2, 3, 5

[8] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: tracking any point with per-frame initialization and temporal refinement. In *Proc. CVPR*, 2023. 3, 5

[9] Carl Doersch, Yi Yang, Dilara Gokay, Pauline Luc, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped training for tracking-any-point. *arXiv*, 2402.00847, 2024. 3, 5, 1

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 1

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16×16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021. 5

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 6, 3

[13] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle videos revisited: Tracking through occlusions using point trajectories. In *Proc. ECCV*, 2022. 1, 2, 3

[14] Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas J. Guibas. AllTracker: efficient dense point tracking at high resolution. *arXiv*, 2506.07310, 2025. 2, 3, 5, 1

[15] Haiwen Huang, Anpei Chen, Volodymyr Havrylov, Andreas Geiger, and Dan Zhang. Loftup: Learning a coordinate-based feature upsampler for vision foundation models. *arXiv preprint arXiv:2504.14032*, 2025. 8

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proc. NeurIPS*, 2015. 2

[17] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *arXiv*, 2024. 1

[18] Nikita Karaev, Ignacio Rocco, Ben Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *Proceedings of the European Conference on Computer Vision (ECCVngo24delta)*, 2024. 2, 3, 5, 1

[19] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-

labelling real videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2025. 3, 5, 8

[20] Simon Kiefhaber, Stefan Roth, and Simone Schaub-Meyer. Removing cost volumes from optical flow estimators. In *Proc. ICCV*, 2025. 2

[21] Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, João Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. TAPVid-3D: a benchmark for tracking any point in 3D. *arXiv*, 2407.05921, 2024. 3

[22] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. In *Proc. CVPR*, 2024. 3

[23] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Feng Li, Bohan Li, Tianhe Ren, and Lei Zhang. Taptrv2: Attention-based position update improves tracking any point. *Advances in Neural Information Processing Systems*, 2024. 3

[24] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, and Lei Zhang. TAPTR: Tracking any point with transformers as detection. In *Proc. ECCV*, 2024. 3, 5

[25] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, 1981. 2

[26] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proc. CVPR*, 2023. 2, 6, 3

[27] É. Mémin and P. Pérez. A multigrid approach for hierarchical motion estimation. In *Proc. ICCV*, 1998. 2

[28] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. *arXiv.cs*, abs/2312.00786, 2023. 5

[29] Tuan Duc Ngo, Peiye Zhuang, Chuang Gan, Evangelos Kalogerakis, Sergey Tulyakov, Hsin-Ying Lee, and Chaoyang Wang. DELTA: Dense efficient long-range 3D tracking for any video. *arXiv*, 2024. 3, 5, 1

[30] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016. 5

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proc. ICCV*, 2021. 4, 5, 8

[32] Peter Sand and Seth J. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80(1), 2008. 3

[33] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Proc. NeurIPS*, 2024. 2

[34] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proc. CVPR*, 1994. 1, 3

[35] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proc. CVPR*, 2018. 1

[36] Dongli Tan, Xingyi He, Sida Peng, Yiqing Gong, Xing Zhu, Jiaming Sun, Ruizhen Hu, Yujun Shen, Hujun Bao, and Xiaowei Zhou. Retracker: Exploring image matching for robust online any point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4306–4316, 2025. 3

[37] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, 2020. 1, 3, 6

[38] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[39] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. 2, 5

[40] Bo Wang, Jian Li, Yang Yu, Li Liu, Zhenping Sun, and Dewen Hu. SceneTracker: long-term scene flow estimation network. *arXiv*, 2403.19924, 2024. 3

[41] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. VGGSfM: visual geometry grounded deep structure from motion. In *Proc. CVPR*, 2024. 3

[42] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 5, 8

[43] Yihan Wang and Jia Deng. WAFT: warping-alone field transforms for optical flow. *arXiv*, 2506.21526, 2025. 2, 3, 6

[44] Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *European Conference on Computer Vision*, 2024. 6, 1

[45] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. $pi^3$: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025. 8

[46] Guangyang Wu, Xiaohong Liu, Kunming Luo, Xi Liu, Qingqing Zheng, Shuaicheng Liu, Xinyang Jiang, Guangtao Zhai, and Wenyi Wang. Accflow: Backward accumulation for long-range optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1

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

[48] 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 *Proc. CVPR*, 2023. 3, 5

[49] Artem Zholus, Carl Doersch, Yi Yang, Skanda Koppula, Viorica Patraucean, Xu Owen He, Ignacio Rocco, Mehdi S. M. Sajjadi, Sarath Chandar, and Ross Goroshin. TAPNext: tracking any point (TAP) as next token prediction. *arXiv*, 2504.05579, 2025. 3

# CoWTracker: Tracking by Warping instead of Correlation

Supplementary Material

## 7. Implementation Details

### 7.1. Pseudo-code of CoWTracker

In Figure 8, we provide a minimal PyTorch-style pseudocode of the proposed WarpTracker update loop. First, we extract features for all video frames with a standard backbone and replicate the anchor-frame features for all queries. We then initialize the track field and the hidden state once. After this setup, the algorithm performs a very simple iterative update: at each of the $K$ iterations, we (i) warp the features according to the current tracks, (ii) concatenate the original features, warped features, tracks, and hidden state, (iii) update the hidden state, and (iv) refine the tracks with a small linear head. Importantly, no correlation or cost volume is constructed at any point in this loop; instead, we operate directly on the raw backbone features of each frame. This design makes the method both easy to implement and straightforward to integrate into existing codebases. Please refer to the main paper for a detailed description of each component.

```
# V [T, H, W, C]   - video frames (T = # of frames)
# K                - number of update iterations

# extract features for all frames
F = feature_encoder(V)            # [T, D, H', W']
F_query = F[0:1].repeat(T)        # [T, D, H', W']

# initialize tracks and hidden state
tracks = zeros_like(F[:, :2])     # [T, 2, H', W']
h = init_hidden(F)                # [T, D, H', W']

for _ in range(K):
   # warp features using current tracks
   F_warp = warp(F,tracks)        # [T, D, H', W']

   # combine features and state  [T, 3D+2, H', W']
   x = concat(F, F_warp,tracks, h)

   # update hidden state
   h = update_hidden(x)           # [T, D, H', W']

   # update tracks with a linear head
   tracks += track_head(h)        # [T, 2, H', W']
```

Figure 8. **Pseudocode for CoWTracker algorithm:** our model iteratively refines track prediction across all frames using repeated warping and updates, and as shown, is also simple to implement.

## 8. Extended Quantitative Results

In this section, we provide expanded quantitative comparisons that complement the results presented in the main paper.

### 8.1. Extended Evaluation on the AllTracker Benchmark Suite

In addition to the datasets considered in the main paper, we further evaluate our method on the extended benchmark suite introduced by AllTracker [14]. This suite augments standard point-tracking benchmarks with three additional datasets (DriveTrack, EgoPoints, and Horse10) covering challenging driving scenes, egocentric videos, and articulated animal motion, respectively. We follow exactly the same evaluation protocol and dataset splits as AllTracker, and report the $\delta_{\text{avg}}$ metric averaged over all sequences.

Table 4 compares CoWTracker with recent sparse trackers, dense trackers, and optical-flow models. Our method achieves the best overall performance on this suite, obtaining an average $\delta_{\text{avg}}$ of 73.6, which improves upon the strongest AllTracker by +2.3 points. CoWTracker is best or tied-best on all individual datasets, including the three newly added benchmarks, demonstrating that CoWTracker generalizes well across diverse motion patterns and scene types.

| Method | Data | Dav. | Dri. | Ego. | Hor. | Kin. | Rgb. | Rob. | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| *Optical Flow Models** | | | | | | | | | |
| AccFlow [46] | Flow | 23.5 | 26.4 | 4.0 | 12.1 | 38.8 | 63.2 | 57.9 | 32.3 |
| RAFT [37] | Flow | 48.5 | 44.8 | 41.0 | 27.8 | 64.3 | 82.8 | 72.2 | 54.5 |
| SEA-RAFT [44] | Flow | 48.7 | 49.4 | 44.0 | 33.1 | 64.3 | 85.7 | 67.6 | 56.1 |
| *Sparse Trackers* | | | | | | | | | |
| PIPs++ [13] | PO | 62.5 | 51.3 | 38.5 | 21.4 | 64.2 | 70.4 | 73.4 | 54.5 |
| CoTracker2 [18] | Kub | 70.9 | 67.8 | 43.2 | 33.9 | 65.8 | 73.4 | 73.0 | 61.1 |
| LocoTrack [6] | Kub | 68.0 | 66.5 | 58.4 | 48.9 | 70.0 | 80.3 | 76.9 | 67.0 |
| BootsTAPIR [9] | Kub+ | 67.9 | 66.9 | 56.8 | 48.8 | 70.6 | 81.0 | 78.2 | 67.2 |
| CoTracker3-Kub [17] | Kub | 77.4 | 69.8 | 58.0 | 47.5 | 70.6 | 83.4 | 77.2 | 69.1 |
| CoTracker3 [17] | Kub+ | 77.1 | 69.8 | 60.4 | 47.1 | 71.8 | 84.2 | 81.6 | 70.3 |
| *Dense Trackers* | | | | | | | | | |
| DELTA [29] | Kub | 75.3 | 67.8 | 40.3 | 41.8 | 66.5 | 83.0 | 74.8 | 64.2 |
| AllTracker-Kub [14] | Kub | 75.2 | 66.1 | 60.3 | 49.0 | 71.3 | 90.1 | 82.2 | 70.6 |
| AllTracker [14] | Kub+ | 76.3 | 65.8 | 62.5 | 49.0 | 72.3 | 90.0 | **83.4** | 71.3 |
| CoWTracker | Kub | **78.0** | **70.7** | **64.4** | **52.7** | **73.1** | **92.8** | **83.4** | **73.6** |

Table 4. **Comparison against state-of-the-art point trackers and optical-flow baselines (incl. extra eval datasets used in AllTracker [14]).** Baseline numbers are taken from [14]. *Optical-flow models are assessed *zero-shot* on the tracking task without any task-specific fine-tuning, and understandably yield lower results.

| **Input Frames** | 2* | 10 | 20 | 40 | 60 | 80 | 100 | 200 |
|---|---|---|---|---|---|---|---|---|
| **All** | 0.07 | 0.27 | 0.57 | 1.37 | 2.42 | 3.75 | 5.35 | 10.9 |
| **- Backbone** *(VGGT)* | 0.02 | 0.10 | 0.25 | 0.76 | 1.51 | 2.54 | 3.85 | 7.89 |
| **- Tracker** | 0.05 | 0.17 | 0.32 | 0.61 | 0.91 | 1.21 | 1.50 | 3.01 |
| **Point/sec** ($\times 10^6$) | 5.37 | 6.96 | 6.60 | 5.49 | 4.66 | 4.01 | 3.51 | 3.45 |
| **Frame/sec** | 28.6 | 37.0 | 35.1 | 29.2 | 24.8 | 21.3 | 18.7 | 18.4 |

Table 5. **CoWTracker runtime vs. video length.** We report *Runtime (in seconds, all and by components, ↓ better)* and *Point / Frame per Second (↑ better)*. *equivalent to optical flow runtime.

## 8.2. Runtime Analysis

We benchmark the runtime of CoWTracker on a single NVIDIA H100 GPU with FlashAttention-3 [33]. For all results we use $K{=}5$ refinement steps, and input resolution $336 \times 560$.

Table 5 reports end-to-end runtime as a function of video length, together with a breakdown into backbone and tracker cost, as well as the resulting point and frame throughput. The tracker itself scales approximately linearly with the number of frames and contributes a smaller fraction of the total runtime for longer sequences. For shorter clips ($\leq 40$ frames) CoWTracker tracks around 6–7 million points per second and runs at over 30 frames per second. For longer videos throughput gradually decreases but remains around 3.5 million points per second and nearly 20 frames per second.

The main runtime growth for long sequences stems from the VGGT backbone, whose complexity scales quadratically with video length. We note that (i) the backbone is essentially orthogonal to our tracker, meaning that any future advances in backbone architecture—including more efficient designs—will directly improve end-to-end speed, and (ii) because the backbone is a general-purpose 3D representation model, WarpTracker effectively acts as a lightweight add-on that can be plugged into existing architectures used for other tasks, delivering the new functionality with very little extra runtime. In practice, one can also process long videos in shorter chunks to avoid the quadratic cost of VGGT, trading only about $1\%$ in $\delta$ accuracy for a substantial runtime reduction.

## 8.3. Memory Analysis

Figure 9 reports the memory required to store either a cost volume or a pair of feature maps (the alternative used by our cost-volume-free warping based head). Cost volumes grow roughly $16\times$ in memory whenever the image resolution doubles, because both the number of spatial locations and the search window increase. As a result, using stride $1/2$ features becomes prohibitively expensive (hundreds of gigabytes in our setting), and even stride $1/4$ is costly. This is why most prior dense trackers operate at stride $1/8$, trading away high-frequency details and often degrading tracking accuracy, especially around thin structures and object boundaries.

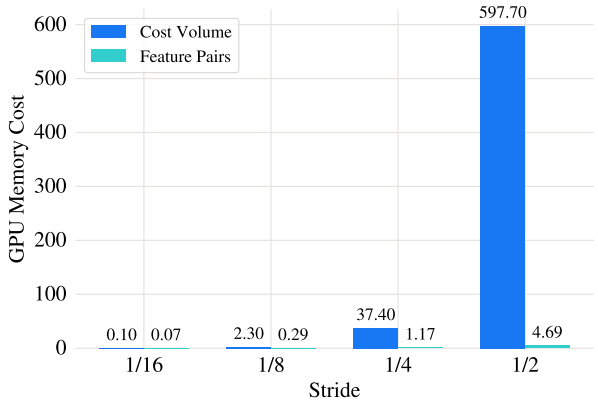



Figure 9. **Cost volumes are prohibitively expensive at high resolution.** Memory grows by $\sim 16\times$ whenever the image size doubles, forcing most methods to use stride 1/8 and sacrifice accuracy. Compared to cost volumes, feature pairs are far more memory efficient. Numbers computed using 100-frame video length, 384x512 resolution, and 16-bit precision.

Our method avoids cost volumes entirely and operates directly on feature maps using a warping-based head. This design removes the quadratic dependence on the spatial search window and yields much more favorable memory scaling with resolution. As shown in Fig. 9, our head can run comfortably on stride $1/2$ features, which only takes a few gigabytes of memory to store, enabling high-resolution tracking without sacrificing practicality.

# 9. Extended Qualitative Results

In this section, we provide additional qualitative visualizations to further demonstrate the robustness and generalization capabilities of our proposed method.

## 9.1. Comparison to Existing Methods

We first evaluate our model on highly challenging sequences characterized by non-rigid motion and severe occlusions and compare with existing methods. As illustrated in Figure 10, the top row ("Dive-in") presents a scenario where a person enters the water, causing significant non-rigid deformation and drastic appearance changes. While baseline methods like DELTA and AllTracker lose the target or drift significantly, our method maintains a consistent track.

Furthermore, the bottom row of Figure 10 highlights a scenario with rapid camera motion where the target object frequently exits the camera frame. This sequence also necessitates tracking a small object (a drone). Our method


Figure 10. **Qualitative comparison on challenging sequences involving non-rigid water motion and small object tracking.** Columns compare DELTA, AllTracker, and Ours. The top sequence exhibits large non-rigid motion and drastic appearance changes as the person dives into the water. The bottom sequence features large camera motion where objects repeatedly go out of frame, requiring the tracking of a small object. Our method maintains a consistent track in these difficult conditions, whereas DELTA and AllTracker fail to recover or exhibit drift. Notably, our approach successfully tracks the small object in the bottom sequence, thanks to our high-resolution feature maps that preserve fine details. Numbers in lower-right boxes indicate frame numbers.

successfully handles these re-entry scenarios and accurately tracks the small target, a capability we attribute to our utilization of high-resolution feature maps which preserve fine-grained spatial details.

### 9.2. Generalization Across Domains

To assess the universality of our tracking model, we test CoWTracker on videos from three distinct domains without fine-tuning. Figure 11 displays qualitative results on:

- **Ego-centric videos** (top row), where camera motion is erratic and hand-object interaction is frequent.
- **Self-driving scenes** (middle row), involving dynamic street environments.
- **Robotics manipulation** (bottom row), requiring precise tracking of manipulated objects.

Notably, our method exhibits strong temporal stability, successfully maintaining accurate tracks over very long sequences (up to 600 frames), as demonstrated in the ego-centric and robotics examples.

### 9.3. Zero-Shot Optical Flow Estimation

Although our architecture is designed for tracking videos, it naturally produces optical flow by treating an image pair as a 2-frame "video". In the main paper, we presented results on the Sintel dataset. Here, in Figure 12, we extend this analysis to the Spring [26] (top two rows) and KITTI [12] (bottom two rows) datasets.

Qualitatively, our predicted flow exhibits sharp motion boundaries and accurate alignment with the ground truth. Crucially, we emphasize that our model is **not** trained

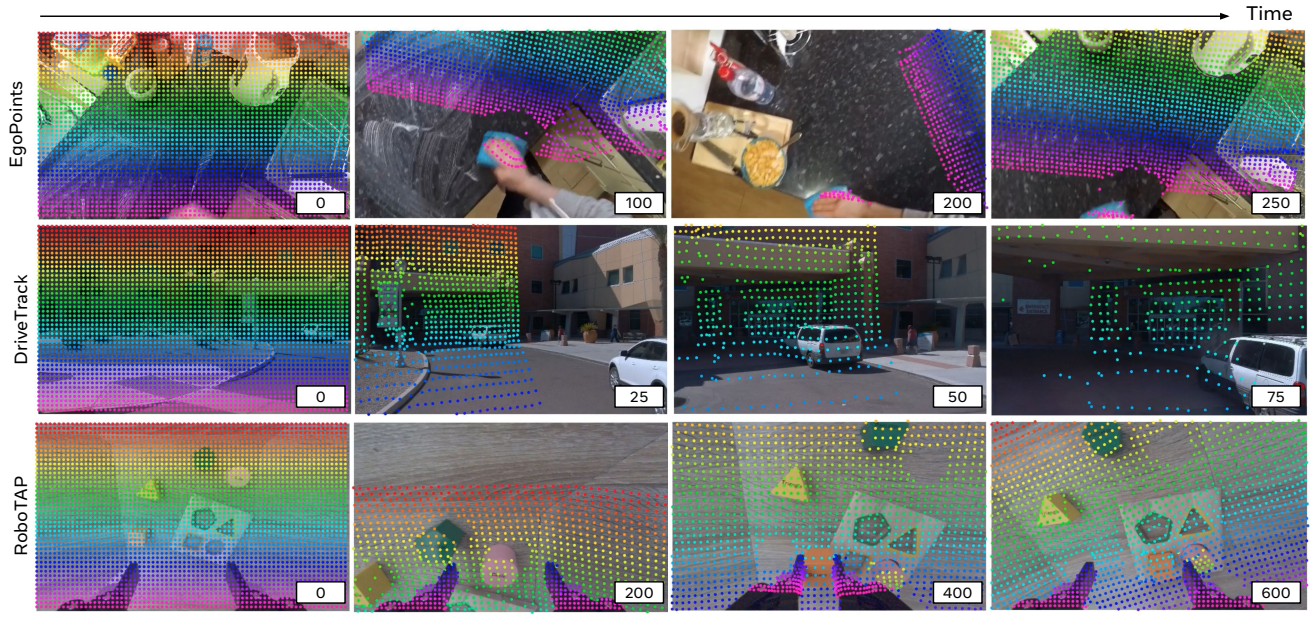



Figure 11. **Generalization to diverse video domains.** Our model demonstrates high tracking accuracy across a variety of domains, including ego-centric videos (top), self-driving scenes (middle), and robotics manipulation tasks (bottom). Notably, our approach is robust over long temporal extents, successfully maintaining tracks in sequences as long as 600 frames (see bottom row).

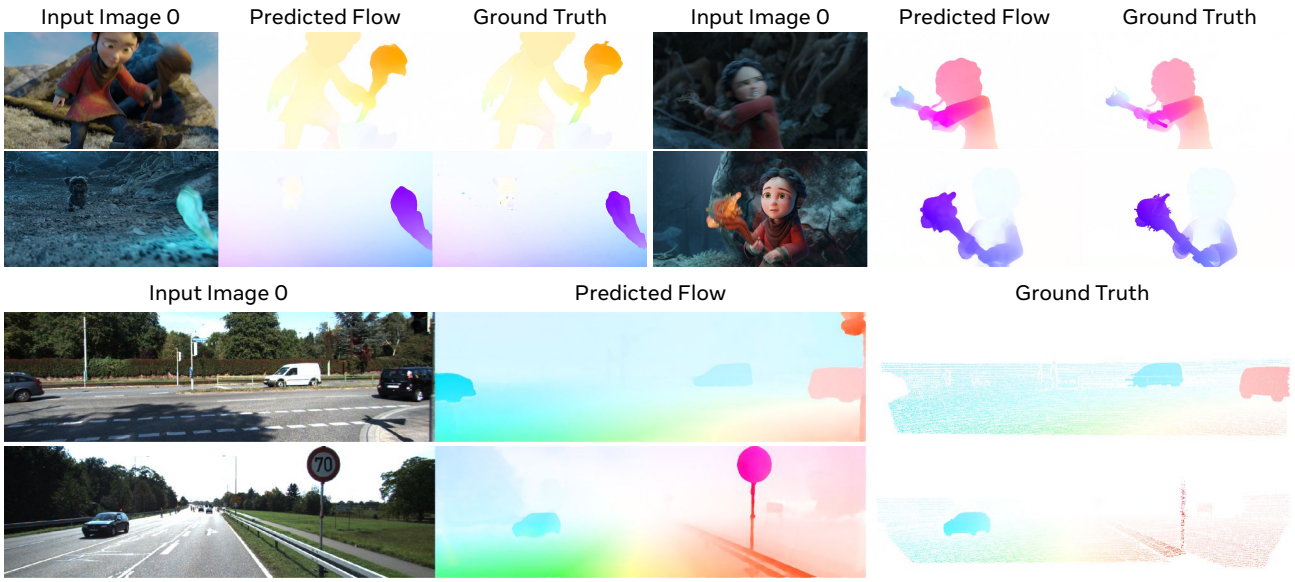



Figure 12. **Additional qualitative optical flow results.** Complementing the Sintel results in the main paper, we visualize performance on the Spring dataset (top two rows) and the KITTI dataset (bottom two rows). Our model predicts high-quality optical flow with sharp motion boundaries and accurate dense correspondence that closely matches the ground truth. It is important to note that we employ the exact same tracking model used throughout the paper; it was *not* trained on any optical flow datasets, including Spring or KITTI, demonstrating its strong generalization capability.

on any optical flow datasets (including Sintel, Spring, or KITTI). These results are achieved using the exact same tracking weights, highlighting the generality of motion understanding ability embedded in our model.