CoTracker3: Simpler and Better Point Tracking by Pseudo-Labeling Real Videos

Nikita Karaev[1,2] Yuri Makarov[1] Jianyuan Wang[1,2] Natalia Neverova[1]
Andrea Vedaldi[1] Christian Rupprecht[2]

[1] Meta AI [2] Visual Geometry Group, University of Oxford
nikita@robots.ox.ac.uk

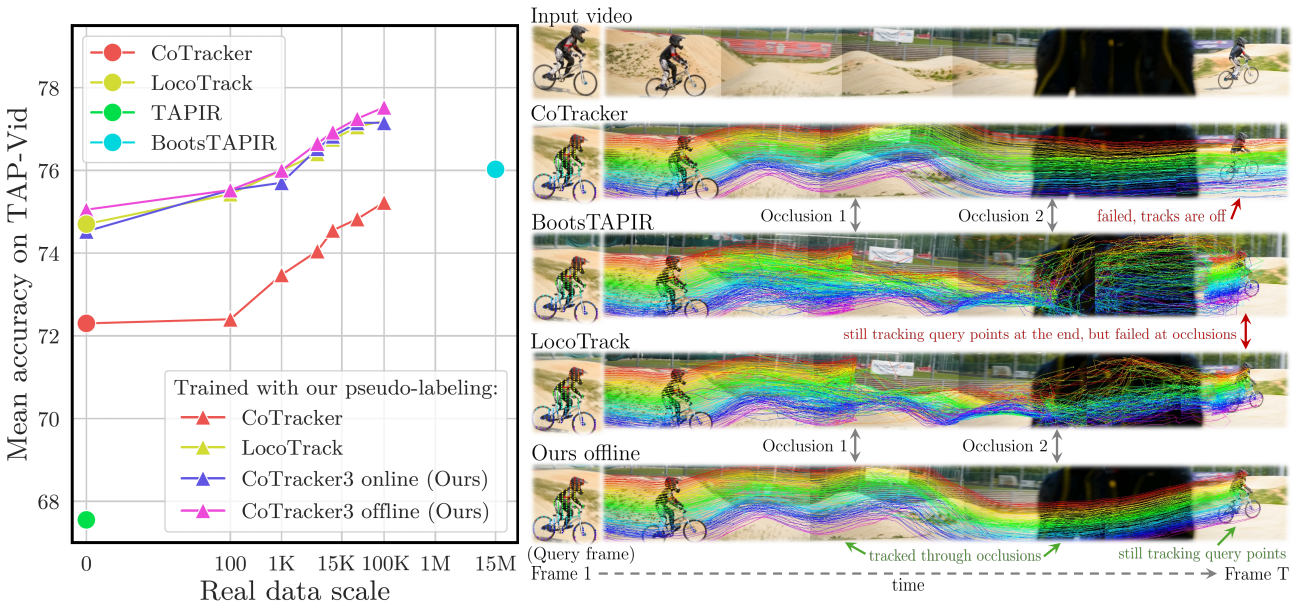

Figure 1. **Scaling point trackers with unsupervised videos.** *Left:* We compare our CoTracker3 to other point trackers. Each model is pre-trained on synthetic data (Kubric) and then fine-tuned on real videos using our simple unsupervised training protocol. Our model and training protocol outperform SoTA by a large margin using only 0.1% of the real training data. *Right:* CoTracker3 is robust to occlusions.

Abstract

We introduce CoTracker3, a new state-of-the-art point tracker. With CoTracker3, we revisit the design of recent trackers, removing components and reducing the number of parameters while also improving performance. We also explore the interplay of synthetic and real data. Recent trackers are trained on synthetic videos due to the difficulty of collecting tracking annotations for real data. However, this can result in suboptimal performance due to the statistical gap between synthetic and real videos. We thus suggest using off-the-shelf trackers as teachers to annotate real videos with pseudo-labels. Compared to other recent attempts at using real data for learning trackers, this scheme is much simpler and achieves better results using 1,000 times less data. CoTracker3 is available here in online (causal) and offline variants.

1. Introduction

Tracking points is a key step in video analysis, particularly for tasks like 3D reconstruction and video editing that require precise recovery of correspondences. Point trackers have evolved significantly in recent years, with designs based on transformer neural networks inspired by PIPs [12]. Notable examples include TAP-Vid [6], which introduced a new benchmark for point tracking, and TAPIR [7], which extended PIPs' design with a global matching stage. CoTracker [16] proposed a transformer architecture that tracks multiple points jointly, achieving further gains in tracking quality, particularly for points that are partially occluded in the video.

In this paper, we introduce a new point tracking model, *CoTracker3*, that builds on the ideas of recent trackers but is significantly simpler, more data-efficient, and more flex-

ible. Our architecture removes some components that recent trackers proposed as necessary for good performance, while still improving on the state of the art. For the first time, we also investigate the data scaling behaviour of such point trackers and show the advantages of different model architectures and training protocols in terms of final tracking quality and data efficiency.

The excellent performance of recent trackers is due to the ability of high-capacity neural networks to learn a robust prior from large numbers of training videos and use this prior to tackle complex and ambiguous tracking cases, such as occlusions and fast motion. Therefore, the availability of high-quality training data is crucial for obtaining solid tracking results [26].

While, in principle, there is no shortage of videos that could be used to train point trackers, it is difficult to annotate them with point tracks [6] manually. Fortunately, synthetic videos [11], which can be annotated automatically, have been found to be a good substitute for real data for low-level tasks like point tracking [12]. Still, building a diverse collection of synthetic videos is expensive at scale, and the sim-to-real gap is not entirely negligible. Hence, using real videos to train point trackers remains an attractive option.

Recent works have thus explored utilizing large collections of real but unlabelled videos to train point trackers. BootsTAPIR [8], in particular, recently achieved state-of-the-art accuracy on the TAP-Vid benchmark by training a model on 15 million unlabelled videos. While the benefits of using more training data have thus been demonstrated, the data scaling behaviour of point trackers is not well understood. In particular, it is unclear if the millions of real training videos used in BootsTAPIR are necessary to train a good tracker. The same can be said about the benefits of their relatively complex semi-supervised training recipe.

Another largely unexplored aspect is the competing designs of different trackers. Transformer architectures like TAPIR [7] and CoTracker [16], as well as recent contributions like LocoTrack [4], each propose significant changes, extensions, new components, and different design decisions. While these are shown to help in the respective papers, it is less clear if they are all essential or whether these designs can be simplified and made more efficient.

CoTracker3 contributes to answering these questions. Our model is based on a simpler architecture and training protocol than recent trackers such as BootsTAPIR and LocoTrack. It outperforms BootsTAPIR by a significant margin on the TAP-Vid and Dynamic Replica [15] benchmarks, while using *three orders of magnitude fewer unlabeled videos* and a simpler training protocol. We also study the data scaling behaviour of this model under increasing amounts of real training videos. We show that LocoTrack benefits in a similar manner to CoTracker3 from scaling data, but cannot track occluded points well.

CoTracker3 borrows elements from prior models, including iterative updates and convolutional features from PIPs, cross-track attention for joint tracking, proxy tokens for efficiency, and unrolled training for windowed operation from CoTracker, as well as the 4D correlation from LocoTrack. At the same time, it significantly simplifies some of these components and removes others, such as the global matching stage of BootsTAPIR and LocoTrack. This helps to identify which components are vital for a good tracker. CoTracker3's architecture is also flexible, as it can operate both offline (i.e., single window) and online (i.e., sliding window) if trained in the same way.

2. Related work

Tracking-Any-Point. The task of *tracking any point* (TAP) was introduced by PIPs [12], who revisited the classic Particle Video [29] method and proposed using deep learning for point tracking. Inspired by RAFT [34], an optical flow algorithm, PIPs extracts correlation maps between frames and feeds them into a network to refine the track estimates. TAP-Vid [6] improved the problem framing, proposed three benchmarks, and introduced TAP-Net, a model for point tracking. TAPIR [7] combined TAP-Net-like global matching with PIPs, resulting in much-improved performance. Zheng et al. [44] introduced another synthetic benchmark, PointOdyssey, and PIPs++, an improved version of PIPs that can track points over extended durations. CoTracker [16] noted a strong correlation between different tracks, which can be exploited to improve tracking, particularly behind occlusions and out-of-frame. DOT [18] further improved CoTracker by densifying its output. VG-GSfM [39] proposed a coarse-to-fine tracker design where tracks are validated through 3D reconstruction, but it only targets static scenes. Inspired by DETR [2], Li et al. [19] introduced TAPTR, an end-to-end transformer architecture for point tracking, representing points as queries in the transformer decoder. LocoTrack [4] extended 2D correlation features to 4D correlation volumes while simplifying the point-tracking pipeline and enhancing efficiency. Our work proposes a further simplified framework that runs 27% faster than LocoTrack, while maintaining the ability to track occluded points via joint tracking, like CoTracker.

Annotating data for point tracking is particularly challenging due to the required precision: the annotation should have (at least) pixel-level accuracy. The prevailing paradigm in point tracking is thus to train models using synthetic data, where such annotations can be obtained automatically and without errors, and to show that the resulting models can generalize to real data. All the methods mentioned above follow this paradigm, and most are trained solely on synthetic Kubric [11].

Semi-supervised correspondence. An alternative to synthetic data is unlabeled real data in combination with unsupervised or semi-supervised learning. For example, one

can use photometric consistency as a proxy for correspondences. Such training is well suited for optical flow and dense tracking but often leads to false matches due to occlusions, repeated textures, or lighting changes. Therefore, it usually requires multi-frame estimates [14], explicit reasoning about occlusions [42], hand-crafted loss terms [23, 25], or various data augmentation strategies [21]. Alternatively, one can use an existing tracker to train another in a process akin to distillation [22]. More robust unsupervised learning signals for long-range tracking can be obtained via simple colorization of grayscale videos by copying colors from the reference frame [38] or other visual patterns [17].

Accounting for cycle consistency [13, 30, 41] or temporal continuity [9, 43] in videos is another way to obtain a reliable proxy signal to learn correspondences without full supervision, or even to learn generic visual features [10, 40]. Recently, Sun et al. [32] proposed refining PIPs and RAFT on a pre-generated dataset with pseudo-labels using color constancy and cycle consistency signals. This pipeline improves tracking, but performance quickly saturates. DINO-Tracker [35] combined test-time per-video optimization with DINOv2 [27] to improve point tracking.

Most relevant to our work, BootsTAPIR [8] improved TAPIR trained on Kubric by fine-tuning it on 15 million real videos using self-training, while retaining a small synthetic dataset with ground-truth supervision to avoid catastrophic forgetting. They proposed applying augmentations to student predictions and trained the model with an exponential moving average (EMA), while computing three different loss masks for robustness. In contrast, our approach uses a simpler design that does not require augmentations, masks, or EMA for training. We also do not need ground-truth supervised data during fine-tuning on pseudo-labels. Instead, our idea is to train a student model by utilizing existing trackers with complementary qualities as teachers. We also show that this protocol only requires a small fraction of the real videos utilized in BootsTAPIR.

3. Method

In this section, we formally introduce the point tracking task and outline the proposed CoTracker3 architecture and the semi-supervised training pipeline we use to train it.

Given a video $(\mathcal{I}_t)_{t=1}^T$, which is a sequence of T frames $\mathcal{I}_t \in \mathbb{R}^{3\times H\times W}$, and a query point $\mathcal{Q} = (t^q, x^q, y^q) \in \mathbb{R}^3$ where t^q indicates the query frame index and (x^q, y^q) represents the initial location of the query point. Our goal is to predict the corresponding point track $\mathcal{P}_t = (x_t, y_t) \in \mathbb{R}^2$, $t = 1, \dots, T$, with $(x_{t^q}, y_{t^q}) = (x^q, y^q)$. As is common in modern point tracking models [7, 16], CoTracker3 also estimates visibility $\mathcal{V}_t \in [0, 1]$ and confidence $\mathcal{C}_t \in [0, 1]$. Visibility indicates whether the tracked point is visible ($\mathcal{V}_t = 1$) or occluded ($\mathcal{V}_t = 0$) in the current frame. At the same time, confidence measures whether the network is confident that the tracked point is within a certain distance from the ground truth in the current frame ($\mathcal{C}_t = 1$). The model initializes all tracks with the query coordinates $\mathcal{P}_t := (x_{t^q}, y_{t^q})$ and sets confidence and visibility to zero, $\mathcal{C}_t := 0$, $\mathcal{V}_t := 0$ for all $t = 1, \dots, T$, then updates these iteratively.

3.1. Training using unlabeled videos

Recent trackers are trained primarily on synthetic data [11] due to the challenge of collecting tracking annotations for real data at scale. However, BootsTAPIR [8] has shown that it is possible to train better trackers by adding unlabeled real videos to the mix. To do so, they propose a sophisticated self-training protocol that uses a large number of unlabeled videos (15M), self-training, data augmentations, and transformation equivariance.

Here, we propose a much simpler protocol that allows us to surpass the performance of BootsTAPIR with 1,000× less data: we use *a variety of existing* trackers to label a collection of real videos, using them as *teachers*, and then use the pseudo-labels to train a new *student* model, which we pre-train using synthetic data.

Notably, the teacher models are *also* trained using only synthetic data. Thus, one may wonder why this protocol should result in a student that is better than any of the teachers. There are several reasons for this: (1) the student benefits from learning from a much larger (albeit noisy) dataset than the synthetic data alone; (2) learning from real videos mitigates the distribution shift between synthetic and real data; (3) there is an ensembling/voting effect which reduces pseudo-annotation noise; (4) the student model may inherit strengths of different teachers, which may excel in different aspects of the task (e.g., offline trackers track occluded points better, and online trackers tend to stick to the query points more closely near the track's origin).

Dataset. To enable such training, we collected a large-scale dataset of Internet-like videos (around 100,000 videos of 30 seconds each) featuring diverse scenes and dynamic objects, primarily humans and animals. We demonstrate that performance improves when training on larger subsets of this data, starting from as few as 100 videos (see Fig. 1).

Teacher models. To create a diverse set of supervisory signals, we employ multiple teacher models trained only on synthetic data from Kubric [11]. Our set of teachers consists of our proposed models CoTracker3 online and CoTracker3 offline, CoTracker [16], and TAPIR [7]. During training, we randomly sample a frozen teacher model for each batch, so the same video may receive pseudo-labels from different teachers over epochs, preventing overfitting and promoting generalization. The teacher models are not updated.

Query point sampling. Trackers require a query point to track in addition to a video. After randomly choosing a teacher for the current batch, we sample a set of query

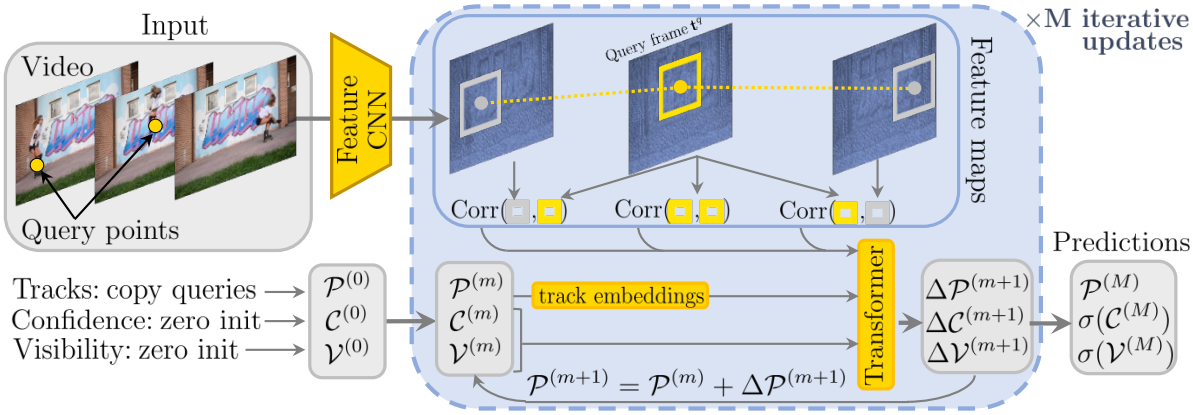

Figure 2. **Architecture.** We compute convolutional features for every frame of the given video, and then the correlations between the feature sampled around the query frame for the query point and all the other frames. We then iteratively update tracks $\mathcal{P}^{(m)} = \mathcal{P}^{(m)} + \Delta\mathcal{P}^{(m+1)}$, confidence $\mathcal{C}^{(m)}$, and visibility $\mathcal{V}^{(m)}$ with a transformer that takes the previous estimates $\mathcal{P}^{(m)}$, $\mathcal{C}^{(m)}$, $\mathcal{V}^{(m)}$ as input.

points for each video. To select such queries, we use the SIFT detector [24], biasing the selection of points to those which are "good to track" [31]. Specifically, we randomly select $\hat{T}$ frames across a video and apply SIFT to generate points to start tracks on these keyframes. Our intuition behind using a feature extractor is guided by its ability to detect descriptive image features whenever possible, while failing to do so in ambiguous cases. We hypothesize that this will serve as a filter for hard-to-track points and thus improve the stability of training. Following this intuition, if SIFT fails to produce a sufficient number of points for any frame, we skip the video completely during training to maintain the quality of our training data.

Supervision. We supervise tracks predicted by the student model with the same loss used to pre-train the model on synthetic data, with only minor modifications for handling occlusion and tracking confidence. These details are given later in Sec. 3.3.

3.2. CoTracker3

We provide two model versions of CoTracker3: offline and online. The online version operates in a sliding window manner, processing the input video sequentially and tracking points forward only. In contrast, the offline version processes the entire video as a single sliding window, enabling point tracking in both forward and backward directions. The offline version tracks occluded points better and also improves the long-term tracking of visible points. However, the maximum number of tracked frames is memory-bound, while the online version can track indefinitely.

Feature maps. We start by computing dense d-dimensional feature maps with a convolutional neural network inspired by [12] for each video frame, i.e., $\Phi_t = \Phi(\mathcal{I}_t), t = 1, \dots, T$. We downsample the input video by a factor of $k = 4$ for efficiency so that $\Phi_t \in \mathbb{R}^{d \times \frac{H}{k} \times \frac{W}{k}}$, and compute the feature maps at $S = 4$ different scales, i.e., $\Phi_t^s \in \mathbb{R}^{d \times \frac{H}{k2^{s-1}} \times \frac{W}{k2^{s-1}}}$, $s = 1, \dots, S$.

4D correlation features. In order to allow the network to locate the query point $\mathcal{Q} = (t^q, x^q, y^q)$ in frames $t = 1, \dots, T$, we compute the correlation between the feature vectors extracted from the map Φ_{t^q} at the query frame t^q around the query coordinates (x^q, y^q) and feature vectors extracted from maps Φ_t, $t = 1, \dots, T$ around current track estimates $\mathcal{P}_t = (x_t, y_t)$ at the other frames.

More specifically, every point $\mathcal{P}_t$ is described by extracting a square neighbourhood of feature vectors at different scales. We denote this collection of feature vectors as:

$$\phi_t^s = \left[\Phi_t^s \left(\frac{x}{ks} + \delta, \frac{y}{ks} + \delta \right) : \delta \in \mathbb{Z}, \ \|\delta\|_\infty \leq \Delta \right], \quad (1)$$

for $s = 1, \dots, S$, where the feature map Φ_t^s is sampled using bilinear interpolation around the point (x_t, y_t). Therefore, for each scale s, $\phi_t^s \in \mathbb{R}^{d \times (2\Delta+1)^2}$ contains a grid of $(2\Delta + 1)^2$ pointwise d-dimensional features.

Next, we define the *4D correlation* [4] $\langle \phi_{t^q}^s, \phi_t^s \rangle = \text{stack}((\phi_{t^q}^s)^\top \phi_t^s) \in \mathbb{R}^{(2\Delta+1)^4}$ for every scale $s = 1, \dots, S$. Intuitively, this operation compares each feature vector around the query point (x^q, y^q) to each feature vector around the track point (x_t, y_t), which the network uses to predict the track update. Before passing them to the transformer, we project these correlations with a multi-layer perceptron (MLP) to reduce their dimensionality, defining the *correlation features* to be $\text{Corr}_t = \big(\text{MLP}(\langle \phi_{t^q}^1, \phi_t^1 \rangle), \dots, \text{MLP}(\langle \phi_{t^q}^S, \phi_t^S \rangle)\big) \in \mathbb{R}^{pS}$, where p is the projection dimension. This MLP architecture is much simpler than the ad hoc module used by LocoTrack [4] for computing correlation features.

Iterative updates. We initialize the confidence $\mathcal{C}_t$ and visibility $\mathcal{V}_t$ with zeros, and the tracks $\mathcal{P}_t$ for all times $t =$

$1, \ldots, T$ with the initial coordinates from the query point $\mathcal{Q}$. We then iteratively update all these quantities with a transformer.

At every iteration, we embed the tracks using the Fourier Encoding [33] of the per-frame displacements, i.e., $\eta_{t \to t+1} = \eta(\mathcal{P}_{t+1} - \mathcal{P}_t)$., Then, we concatenate the track embeddings (in both directions $\eta_{t \to t+1}$ and $\eta_{t-1 \to t}$), confidence $\mathcal{C}_t$, visibility $\mathcal{V}_t$, and the 4D correlations Corr_t for every query point $i = 1, \ldots, N$ as $\mathcal{G}_t^i = \left(\eta_{t-1 \to t}^i, \eta_{t \to t+1}^i, \mathcal{C}_t^i, \mathcal{V}_t^i, \mathrm{Corr}_t^i\right)$. $\mathcal{G}_t^i$ forms a grid of input tokens for the transformer that span time T and the number of query points N. The transformer Ψ takes this grid as input, adds standard Fourier time embeddings, and applies factorized time attention [1] with $t = 1, \ldots, T$ and group attention with $i = 1, \ldots, N$. It also uses the proxy tokens of [16] for efficiency. This transformer estimates the updates to tracks, confidence, and visibility incrementally as $(\Delta\mathcal{P}, \Delta\mathcal{C}, \Delta\mathcal{V}) = \Psi(\mathcal{G})$. We update tracks $\mathcal{P}$, confidence $\mathcal{C}$ and visibility $\mathcal{V}$ M times, setting $\mathcal{P}^{(m+1)} = \mathcal{P}^{(m)} + \Delta\mathcal{P}^{(m+1)}$, $\mathcal{C}^{(m+1)} = \mathcal{C}^{(m)} + \Delta\mathcal{C}^{(m+1)}$, and $\mathcal{V}^{(m+1)} = \mathcal{V}^{(m)} + \Delta\mathcal{V}^{(m+1)}$. Note that we resample the pointwise features ϕ around updated tracks $\mathcal{P}^{(m+1)}$ and recompute the correlations Corr after every update.

3.3. Model training

During pre-training on synthetic data, we supervise both visible and occluded tracks using the Huber loss with threshold $\rho = 6$ and weight exponentially increasing with the iteration m. We assign a smaller weight to the loss term for occluded points, setting

$$\mathcal{L}_{\text{track}}(\mathcal{P}, \mathcal{P}^\star) = \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma^{M-m} w_t \, \mathrm{Huber}_\rho(\mathcal{P}_t^{(m)}, \mathcal{P}_t^\star), \tag{2}$$

where $\gamma = 0.8$ is a discount factor, $w_t = 1$ if the point is visible and $w_t = 1/5$ otherwise, and the $\star$ symbol denotes the ground truth.

Confidence and visibility are supervised with the Binary Cross Entropy (BCE) loss at every iterative update. The ground truth for confidence is defined by an indicator function that checks whether the predicted track is within $r = 12$ pixels of the ground truth track for the current update. We apply the sigmoid function to the predicted confidence and visibility before computing the loss. The confidence loss $\mathcal{L}_{\text{conf}}(\mathcal{C}, \mathcal{P}, \mathcal{P}^\star)$ is given by

$$\sum_{m=1}^{M} \sum_{t=1}^{T} \gamma^{M-m} \, \mathrm{CE}\left(\sigma(\mathcal{C}_t^{(m)}), \mathbb{1}_{\|\mathcal{P}_t^{(m)} - \mathcal{P}_t^\star\|_2 < r}\right). \tag{3}$$

The visibility loss is defined similarly as:

$$\mathcal{L}_{\text{occl}}(\mathcal{V}, \mathcal{V}^\star) = \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma^{M-m} \, \mathrm{CE}\left(\sigma(\mathcal{V}_t^{(m)}), \mathcal{V}_t^\star\right). \tag{4}$$

Training using pseudo-labels. When using pseudo-labelled videos, we supervise CoTracker3 using the same loss (2) as for the synthetic data, but found it more stable not to supervise confidence and visibility. To avoid forgetting the latter predictions, we use a separate linear layer to estimate confidence and visibility and simply freeze it at this training stage.

Online model. Both online and offline versions of CoTracker3 have the same architecture. The main difference between them is the way of training. The online version processes videos in a windowed manner: it takes T' frames as input, predicts tracks, then moves forward by $T'/2$ frames, and repeats. It uses overlapped predictions for tracks, confidence, and visibility from the previous sliding window as initialization for the current window.

During training, we compute the same losses (2) to (4) for the online version separately for each sliding window. Then, we take the mean across all the sliding windows. Since the online version can track points only forward in time, we compute the losses only starting from the first window with the query frame t^q onwards. For the offline version, however, we compute the losses for every frame because it tracks points in both directions. We train the online version on videos of the same length, while the offline version needs to see videos of different lengths during training to avoid overfitting to a specific length. With this in mind, for the offline version, we randomly trim a video between $T/2$ and T frames and linearly interpolate time embeddings during training.

3.4. Discussion

Our model includes several simplifications and improvements compared to previous architectures like PIPs, TAPIR, and CoTracker, including: (1) The model borrows 4D correlation from LocoTrack but simplifies it by utilizing an MLP to process the correlation features instead of their ad-hoc architecture; (2) It estimates confidence for every tracked point; (3) Compared to CoTracker, the grid of tokens $\mathcal{G}$ is simplified, using only correlation features and the Fourier embeddings of displacements; (4) The visibility flags are updated at each iteration along with other quantities instead of using a separate network; (5) Compared to TAPIR, BootsTAPIR, and LocoTrack, CoTracker3 *does not* use a global matching module, as we found it redundant.

A benefit of these simplifications is that CoTracker3 is considerably leaner and faster than other similar trackers. Specifically, CoTracker3 has $2\times$ fewer parameters than CoTracker, while the absence of global matching and the use of an MLP to process correlations make CoTracker3 27% faster than the fastest tracker (LocoTrack), despite cross-track attention.

Method	Train	Size↓	Time↓	Kinetics			RGB-S			DAVIS			Dynamic Replica		RoboTAP			Mean
				AJ ↑	δ^{vis}_{avg} ↑	OA ↑	AJ ↑	δ^{vis}_{avg} ↑	OA ↑	AJ ↑	δ^{vis}_{avg} ↑	OA↑	δ^{vis}_{avg} ↑	δ^{occ}_{avg} ↑	AJ ↑	δ^{vis}_{avg} ↑	OA↑	δ^{vis}_{avg} ↑
PIPs++ [44]	PO	25M	—	—	63.5	—	—	58.5	—	—	73.7	—	64.0	28.5	—	63.0	—	64.5
TAPIR [7]	Kub	31M	293	49.6	64.2	85.0	55.5	69.7	88.0	56.2	70.0	86.5	66.1	27.2	59.6	73.4	87.0	68.7
CoTracker [16]	Kub	45M	472	49.6	64.3	83.3	67.4	78.9	85.2	61.8	76.1	88.3	68.9	37.6	58.6	70.6	87.0	71.8
TAPTR [19]	Kub	—	—	49.0	64.4	85.2	60.8	76.2	87.0	63.0	76.1	91.1	69.5	34.1	60.1	75.3	86.9	72.3
LocoTrack [4]	Kub	**12M**	290	52.9	66.8	85.3	69.7	83.2	89.5	62.9	75.3	87.2	71.4	29.8	62.3	76.2	87.1	74.6
CoTracker3 (Ours, online)	Kub	25M	405	54.1	66.6	87.1	71.1	81.9	90.3	**64.5**	76.7	89.7	72.9	41.0	60.8	73.7	87.1	74.4
CoTracker3 (Ours, offline)	Kub	25M	**209**	53.5	66.5	86.4	74.0	84.9	90.5	63.3	76.2	88.0	69.8	41.8	59.9	73.4	87.1	74.2
BootsTAPIR [8]	Kub+15M	78M	303	54.6	68.4	86.5	70.8	83.0	89.9	61.4	73.6	88.7	69.0	28.0	64.9	**80.1**	86.3	74.8
CoTracker3 (Ours, online)	Kub+15k	25M	405	**55.8**	**68.5**	**88.3**	71.7	83.6	91.1	63.8	76.3	90.2	**73.3**	40.1	**66.4**	78.8	**90.8**	**76.1**
CoTracker3 (Ours, offline)	Kub+15k	25M	**209**	54.7	67.8	87.4	**74.3**	**85.2**	**92.4**	64.4	**76.9**	**91.2**	72.2	**42.3**	64.7	78.0	89.4	76.0

Table 1. **Results on TAP-Vid benchmarks and Dynamic Replica** CoTracker3 trained on synthetic Kubric shows strong performance compared to other models, while the online version fine-tuned on 15k additional real videos (Kub+15k) outperforms all the other methods, even BootsTAPIR trained on 1,000× more real videos. Training data: (Kub) Kubric [11], (PO) Point Odyssey [44]. Size in number of params; speed expressed as μs per frame and per tracked point. See Fig. 3 for qualitative results on RoboTAP.

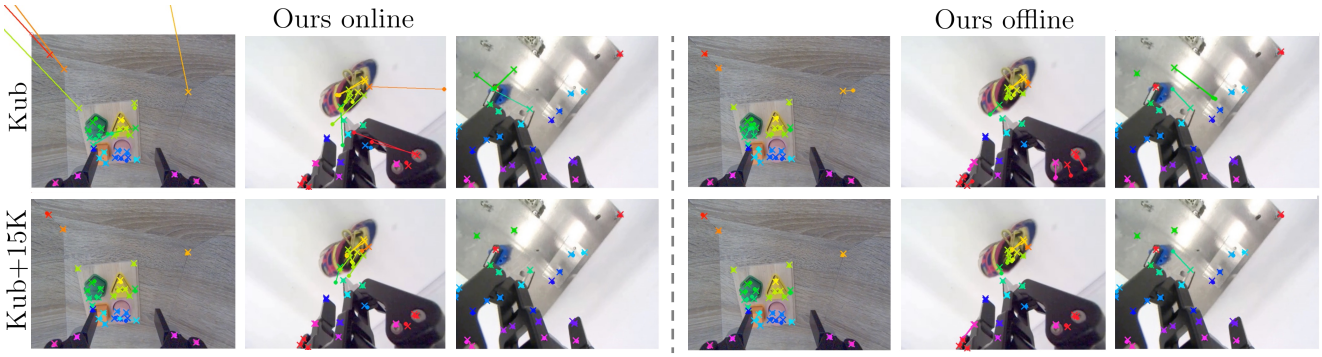

Figure 3. **Qualitative results on RoboTAP.** Predictions of online (first three columns) and offline (last three columns) models on RoboTAP before (first row) and after (second row) scaling. We visualize the distance between ground truth (crosses) and model predictions (points). Scaling helps to improve predictions of both online and offline models.

4. Experiments

In this section, we describe our evaluation protocol. We then compare our online and offline models to state-of-the-art trackers, analyze their performance on occluded points (Sec. 4.1), show how models scale with the proposed pseudo-labeling pipeline (Sec. 4.2), and ablate the design choices of the architecture and scaling pipeline (Sec. 4.3).

Evaluation protocol. We conduct our evaluation on **TAP-Vid** [6], comprising TAP-Vid-Kinetics, TAP-Vid-DAVIS, and RGB-Stacking. TAP-Vid-Kinetics consists of 1,144 YouTube videos from the Kinetics-700–2020 validation set [3], featuring complex camera motion and cluttered backgrounds, with an average of 26 tracks per video. TAP-Vid-DAVIS comprises 30 real-world videos from the DAVIS 2017 validation set [28], with an average of 22 tracks per video. RGB-Stacking is a synthetically generated dataset of robotic videos with many texture-less regions that are difficult to track.

We use the standard TAP-Vid metrics: Occlusion Accuracy (OA; accuracy of occlusion prediction as binary classification), δ^{vis}_{avg} (fraction of visible points tracked within 1, 2, 4, 8, and 16 pixels, averaged over thresholds), and Average Jaccard (AJ, measuring tracking and occlusion prediction accuracy together). All videos are resized to 256×256 pixels before being processed by the model.

Similarly, we evaluate CoTracker3 on **RoboTAP** [37], which contains 265 real-world videos of robotic manipulation tasks with an average duration of 272 frames. Following [6], we evaluate TAP-Vid and RoboTAP in the "first query" mode: sampling query points from the first frame where they become visible. Additionally, we evaluate on **DynamicReplica** [15] following [16]. Because this dataset is synthetic, the tracker can be evaluated on occluded points. The evaluation subset of Dynamic Replica consists of 20 long (300 frames) sequences of articulated 3D models. We evaluate these benchmarks at their native resolution but resize the predictions to 256×256 pixels and report the accuracy of visible (δ^{vis}_{avg}) and occluded points (δ^{occ}_{avg}) using the same thresholds as in TAP-Vid.

Cross-track attention	Dynamic Replica	
	δ^{vis}_{avg} ↑	δ^{occ}_{avg} ↑
✗	71.3	35.9
✓	**72.9**	**41.0**

Table 2. **Impact of cross-track attention on occluded tracking.** Cross-track attention improves the tracking of occluded points substantially. It also improves visible points, but the effect is smaller.

Self-training	Mean on TAP-Vid		
	AJ↑	δ_{avg} ↑	OA↑
✗	62.2	74.5	88.2
✓	**63.5**	**75.7**	**89.5**

Table 3. **Self-training.** Training CoTracker3 online on its own predictions improves the model. We use 10k real videos and train to convergence.

4.1. Comparison to the state-of-the-art

For fairness with trackers blind to the correlation between tracks, we evaluate CoTracker3 on TAP-Vid on one query point at a time and sample additional support points to leverage joint tracking, following [16]. This ensures that no information about objects in videos leaks to the tracker through the selection of benchmark points (which correlate with objects in benchmarks). We multiply predicted visibility by predicted confidence and apply a threshold to the resulting quantity as in [7], improving the AJ and OA metrics.

As shown in Tab. 1, CoTracker3 is highly competitive with other trackers on various benchmarks even when trained only on synthetic data (Kub). Adding unlabeled videos using the approach of Sec. 4.2 (+15k) boosts the results well above the state of the art for all metrics on DAVIS, RGB-S, and Kinetics, and for two out of three metrics (AJ and OA) on RoboTAP. The +15k offline version is even better than the online one on DAVIS and RGB-S, but worse on Kinetics and RoboTAP. In terms of data efficiency, despite being trained on just 15k additional real videos, our models outperform BootsTAPIR, which was trained using 15M videos (i.e., **1,000 times more**). Slightly better performance can be obtained by increasing the data further (Sec. 4.2). LocoTrack also benefits similarly from our training scheme but struggles during occlusions, as shown next.

Tracking occluded points We compare CoTracker3 with other methods on Dynamic Replica in Tab. 1 (δ^{occ}_{avg} and OA columns). On this benchmark, CoTracker3 online outperforms all other methods even when trained solely on Kubric; in particular, it is much better than LocoTrack, which justifies additional parameters in cross-track attention modules. Adding 15k real videos improves tracking of visible points for both online and offline versions, but only the offline model improves tracking of occluded points. Furthermore, CoTracker3 offline tracks occluded points better than the online version. This is because accessing all video frames at once helps to interpolate trajectories behind occlusions, as the offline version can see future frames, unlike the online one. See Fig. 4 for qualitative results.

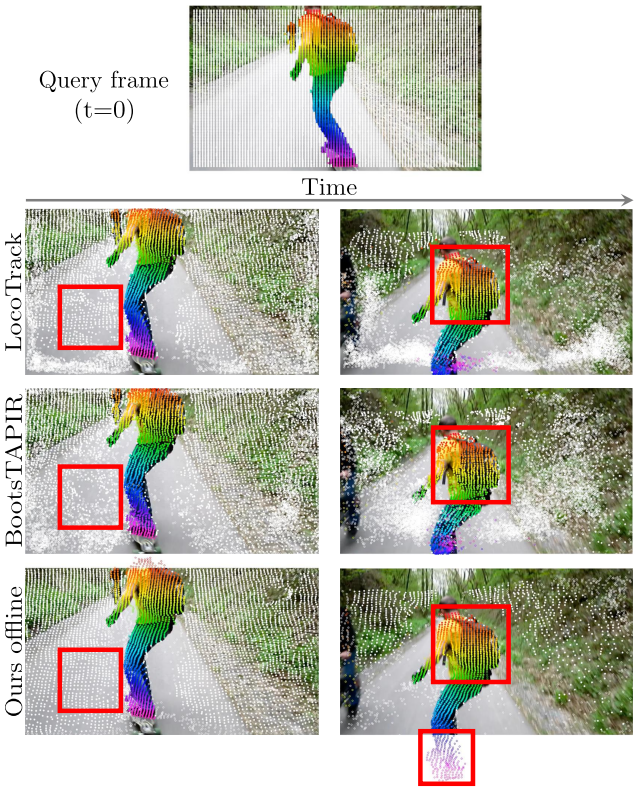

Figure 4. **Qualitative comparison.** Tracking a grid of 100×100 points from the first frame should maintain grid patterns in future frames when the motion is simple. LocoTrack and CoTracker3 are more consistent than BootsTAPIR, but neither LocoTrack nor BootsTAPIR can track through occlusions (2nd column) and also lose more background (1st column) points.

4.2. Scaling experiments

In Fig. 1, we show how CoTracker3, LocoTrack, and CoTracker [16] improve with our pseudo-labeling pipeline as the training set size increases. Starting with models pretrained on a synthetic dataset [11] (0 along the x-axis), we train them on progressively larger real datasets: 0.1k, 1k, 5k, 15k, 30k, and 100k videos. Models are trained to convergence on their respective subsets. All models already improve with only 0.1k real-world videos and continue improving with more. Improvements for CoTracker3 online, offline, and LocoTrack tend to plateau after 30k videos, likely because the student surpasses the teachers. This may also explain why CoTracker, initially much weaker than two of its teachers (CoTracker3 online and offline), keeps improving up to and possibly beyond 100k videos, which is the maximum we could afford to explore. Our training strategy is effective for all these models. We analyze the effect of using a scaled CoTracker3 as a new teacher in the supplement. For comparison, BootsTAPIR [8] uses 15 million real videos and a complex protocol involving augmen-

RT onl.	RT offl.	TAPIR	CoTr.	Mean on TAP-Vid		
				AJ↑	δ_{avg} ↑	OA↑
✗	✗	✗	✗	62.2	74.5	88.2
✓	✗	✗	✗	63.5	75.7	89.5
✓	✓	✗	✗	**64.5**	76.4	89.9
✓	✗	✓	✗	63.6	76.2	89.7
✓	✗	✗	✓	64.2	76.5	90.1
✓	✓	✓	✗	64.0	76.6	89.9
✓	✗	✓	✓	64.2	76.6	90.1
✓	✓	✗	✓	64.0	76.6	90.0
✓	✓	✓	✓	64.0	**76.8**	**90.2**

Table 4. **Models used as teachers.** We use CoTracker3 online as a student model and ablate different combinations of teacher models. The first row corresponds to the model trained only on synthetic data. The second row corresponds to self-training. Generally, the more diverse teachers we have, the better is the tracking accuracy (δ_{avg}).

Sampling	Kinetics	DAVIS	RoboTAP	RGB-S
Uniform	67.9	76.9	78.4	**84.0**
SuperPoint	68.1	76.7	**78.9**	81.9
DISK	68.0	76.7	78.6	82.7
SIFT	**68.2**	**77.0**	78.8	83.3

Table 5. **Point sampling strategies** on δ_{avg} on TAP-Vid. SIFT is overall best, but the method is robust w.r.t. this choice.

Frozen head	Average on TAP-Vid		
	AJ ↑	δ_{avg} ↑	OA ↑
✗	63.2	76.6	86.3
✓	**64.0**	**76.8**	**90.2**

Table 6. Average AJ, δ_{avg} and OA on TAP-Vid, where **freezing** the confidence and visibility heads improves performance, **avoiding forgetting**.

tations, loss masks, and more.

Interestingly, we found that training CoTracker3 with its own predictions as annotations without other teachers (i.e., self-training) further improves the results on all the TAP-Vid benchmarks by $+1.2$ points on average (see Tab. 3). Hence, self-training on real data helps the model reduce the domain gap between real and synthetic data.

4.3. Ablations

Cross-track attention. Table 2 shows that cross-track attention improves results, particularly for occluded points ($+5.1$ occluded vs. $+1.6$ visible on Dynamic Replica). This is because, by using cross-track attention, the model can guess the positions of occluded points based on the positions of visible ones. This cannot be done if the points are tracked independently.

Teacher models. We assess the impact of using multiple teachers for generating pseudo-labels in Tab. 4. We start by removing weaker models and always keep the student model itself as a teacher. We demonstrate that removing a teacher always leads to worse results compared to the last row, where we train with all four teacher models. This shows that every teacher is important and that the student model can always extract complementary knowledge, even from weaker teachers.

Point sampling. In Tab. 5, we have explored alternative point sampling methods, including LightGlue [20], SuperPoint [5], and DISK [36]. The choice of sampling method does not significantly affect performance. However, SIFT sampling results are consistently high across all the TAP-Vid datasets.

Freezing the confidence and visibility head. In Tab. 6, we show that splitting the transformer head into a separate head for tracks and a head for confidence and visibility helps to avoid forgetting when supervising only tracks while training on real data. We freeze the head for confidence and visibility at this stage. This improves AJ by $+0.8$ and OA by $+3.9$ on TAP-Vid on average.

5. Conclusion

We introduced CoTracker3, a new point tracker that outperforms the state of the art on TAP-Vid and other benchmarks. CoTracker3's architecture combines several effective ideas from recent trackers while eliminating unnecessary components and simplifying others.

Importantly, CoTracker3 demonstrates the power of a simple pseudo-labeling training protocol, where real videos are annotated using several off-the-shelf trackers and then used to fine-tune a model that outperforms all its teachers. With this protocol, CoTracker3 can surpass trackers trained on 1,000 times more videos. By tracking points jointly, CoTracker3 handles occlusions better than any other model, particularly in offline mode. Our model can serve as a building block for tasks requiring motion estimation, such as 3D tracking, controlled video generation, dynamic 3D reconstruction, or motion data annotation in robotics. Moreover, CoTracker3's efficiency enables real-time applications, making it practical for deployment in interactive systems and real-world scenarios.

Acknowledgements

We want to thank Zihang Lai, Edgar Sucar, Laurynas Karazija, Stanislaw Szymanowicz, Shalini Maiti, Minghao Chen,

Ruining Li, Carl Doersch and Adam W. Harley for the insightful discussions.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Icml*, page 4, 2021. 5

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. ECCV*. Springer, 2020. 2

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[4] Seokju Cho, Jiahui Huang, Jisu Nam, Honggyu An, Seungryong Kim, and Joon-Young Lee. Local all-pair correspondence for point tracking. *Proc. ECCV*, 2024. 2, 4, 6

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018. 8

[6] 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. *arXiv*, 2022. 1, 2, 6

[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. *arXiv*, 2306.08637, 2023. 1, 2, 3, 6, 7

[8] 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 preprint arXiv:2402.00847*, 2024. 2, 3, 6, 7

[9] Peter Földiák. Learning invariance from transformation sequences. *Neural computation*, 3(2), 1991. 3

[10] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proc. ICCV*, 2015. 3

[11] 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 *Proc. CVPR*, 2022. 2, 3, 6, 7

[12] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *Proc. ECCV*, 2022. 1, 2, 4

[13] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. *Proc. NeurIPS*, 33, 2020. 3

[14] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proc. ECCV*, 2018. 3

[15] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Dynamicstereo: Consistent dynamic depth from stereo videos. In *Proc. CVPR*, 2023. 2, 6

[16] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *Proc. ECCV*, 2024. 1, 2, 3, 5, 6, 7

[17] Zihang Lai, Erika Lu, and Weidi Xie. Mast: A memory-augmented self-supervised tracker. In *Proc. CVPR*, 2020. 3

[18] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. In *CVPR*, 2024. 2

[19] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, and Lei Zhang. Taptr: Tracking any point with transformers as detection. *arXiv preprint arXiv:2403.13042*, 2024. 2, 6

[20] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed, 2023. 8

[21] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *Proc. CVPR*, 2020. 3

[22] Pengpeng Liu, Irwin King, Michael R Lyu, and Jia Xu. Ddflow: Learning optical flow with unlabeled data distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. 3

[23] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *Proc. CVPR*, 2019. 3

[24] David G Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999. 4

[25] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 3

[26] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European conference on computer vision (ECCV)*, pages 300–317, 2018. 2

[27] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv*, 2023. 3

[28] 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 *Proc. CVPR*, 2016. 6

[29] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80, 2008. 2

[30] Qiuhong Shen, Lei Qiao, Jinyang Guo, Peixia Li, Xin Li, Bo Li, Weitao Feng, Weihao Gan, Wei Wu, and Wanli Ouyang. Unsupervised learning of accurate siamese tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8101–8110, 2022. 3

[31] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proc. CVPR*, 1994. 4

[32] Xinglong Sun, Adam W Harley, and Leonidas J Guibas. Refining pre-trained motion models. *arXiv preprint arXiv:2401.00850*, 2024. 3

[33] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020. 5

[34] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, 2020. 2

[35] Narek Tumanyan, Assaf Singer, Shai Bagon, and Tali Dekel. Dino-tracker: Taming dino for self-supervised point tracking in a single video. In *European Conference on Computer Vision*, pages 367–385. Springer, 2025. 3

[36] Michał J. Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient, 2020. 8

[37] 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, 2023. 6

[38] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proc. ECCV*, 2018. 3

[39] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *Proc. CVPR*, 2024. 2

[40] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015. 3

[41] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *Proc. CVPR*, 2019. 3

[42] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *Proc. CVPR*, 2018. 3

[43] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4), 2002. 3

[44] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2, 6