DINO-Tracker: Taming DINO for Self-Supervised Point Tracking in a Single Video

Narek Tumanyan*, Assaf Singer*, Shai Bagon, Tali Dekel

Weizmann Institute of Science
*Indicates equal contribution.
Project webpage: dino-tracker.github.io

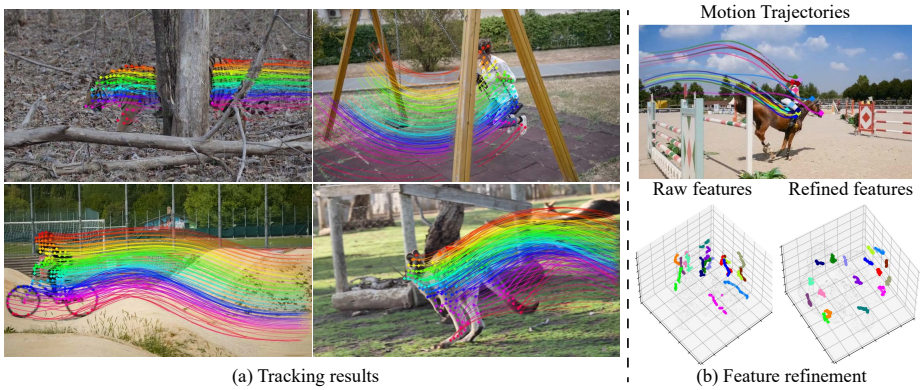

Fig. 1: *DINO-Tracker* provides long-range dense trajectories, past repeating occlusions and during challenging object deformations (a); For visualization purposes, the trajectories are shown for sampled points, yet our method tracks any point. Our test-time training framework leverages a pre-trained DINO-ViT model, and optimizes its internal features for tracking in a single video. (b) *Visualization of trajectory features using t-SNE:* We reduce the dimensionality of foreground features extracted from all frames to 3D using t-SNE, for both raw DINO features and our optimized ones; Features sampled along ground-truth trajectories are marked in color, where each color indicates a different trajectory. Our refined features exhibit tight "trajectory-clusters", allowing our method to associate matching points across distant frames and occlusion.

Abstract. We present DINO-Tracker – a new framework for long-term dense tracking in video. The pillar of our approach is combining test-time training on a single video, with the powerful localized semantic features learned by a pre-trained DINO-ViT model. Specifically, our framework simultaneously adopts DINO's features to fit to the motion observations of the test video, while training a tracker that directly leverages the refined features. The entire framework is trained end-to-end using a combination of self-supervised losses, and regularization that allows us to retain and benefit from DINO's semantic prior. Extensive evaluation demonstrates that our method achieves state-of-the-art results on known benchmarks. DINO-tracker significantly outperforms self-supervised methods and is competitive with state-of-the-art supervised trackers, while outperforming them in challenging cases of tracking under long-term occlusions.

1 Introduction

Establishing dense point correspondences in video has seen tremendous progress in recent years. In the case of short-term dense motion estimation, i.e., optical flow estimation, the research community has been primarily focused on *supervised learning* – designing powerful feedforward models that are trained on various synthetic datasets, using ground-truth supervision [54]. Recently, this trend has been expanded to *long-range* point tracking in video. With the rise of new architectures (e.g., Transformers [14]) and new synthetic datasets that provide long-term trajectories supervision [12,59], various supervised trackers have been developed, demonstrating impressive results [12, 13, 25]. Nevertheless, tracking *every* point in a video across its *entire temporal duration* poses fundamental challenges to this prevalent supervised approach. First, synthetic datasets for point tracking, which often consist of moving objects in unrealistic configurations, are limited in their diversity and scale, relative to the vast distribution of motion and objects in natural videos. In addition, existing models are still restricted in their ability to aggregate information across the entire spatiotemporal extent of a video – a pivotal component in tracking especially under long-term occlusions (e.g., correctly matching a point before it is occluded and after it is revealed).

Aiming to tackle the above challenges, Omnimotion [48] recently proposed to take the opposite direction through a test-time optimization framework that lifts tracking into 3D, using optical flow and video reconstruction as supervision. By optimizing a tracker on a given test video, Omnimotion essentially solves for the motion of all video pixels at once. Nevertheless, their main drawback is the heavy reliance on optical flow and the information available in a *single* video – it does not benefit from *external* knowledge and priors about the visual world.

In this paper, we propose to close the gap between test-time training and learning from extensive data by combining the best of both worlds: a test-time optimization framework that is tailored to a specific video, coupled with the powerful feature representation learned by an external image model trained on broad unlabeled images. Specifically, inspired from the tremendous progress in self-supervised learning, our framework leverages a pre-trained DINOv2 model [35] – a Vision Transformer distilled using a large collection of natural images. DINO's features have been shown to capture fine-grained semantic information and has been used for various visual tasks such as segmentation and semantic correspondences (e.g., [2,32,41]). Our work is the first to consider these features for dense tracking. We show that raw DINO feature matching serves as a strong baseline for tracking, yet the features are not discriminative enough to support sub-pixel accurate tracking on their own, as can be seen in the t-SNE visualization of Fig. 1(b). Our framework simultaneously refines DINO's features to fit to the motion observations of the test video, while training a tracker that directly leverages the refined features. To this end, we formulate a new objective function that goes beyond optical flow supervision by fostering robust semantic feature-level correspondences derived from DINO within our refined feature space.

We extensively evaluate our framework across established benchmarks and demonstrate its superiority in scenarios requiring semantic understanding, deal-

ing with appearance ambiguity, and handling long occlusions. Our tracker achieves state-of-the-art (SOTA) performance compared to previous self-supervised methods, and reveals a significant boost in tracking through long occlusion, compared to SOTA supervised trackers. To summarize, our contributions are as follows:
– We are the first to harness pre-trained DINO features for point-tracking.
– DINO-Tracker is the first to combine test-time training with external priors.
– We achieve a notable performance boost w.r.t. prior methods in tracking through long-term occlusions.

2 Related Work

Optical flow. Classical optical flow optimization methods are based on color constancy and motion smoothness (e.g., [5, 6, 19, 29]). Later, these hand-crafted priors have been replaced by data-driven approaches (e.g., [15, 21, 23, 42, 44, 51, 52]), where modern deep learning-based optical flow methods typically take a supervised learning approach by leveraging synthetic training data containing ground truth optical flow labels. While optical-flow estimation has seen great progress, establishing accurate dense correspondences between nearby frames, extending it to long-term tracking (e.g., by chaining pairwise correspondences) is hampered by occlusions and prone to error accumulation. In our method, we use RAFT [44] to derive short-term motion supervision for our model.
Learning correspondences from videos. While optical flow focuses on dense matches between consecutive frames, other methods were developed for matching points across distant frames. Classical methods used hand-crafted features (e.g, [27, 28]), while more recently, these correspondences were learned in a weakly or self-supervised manner [3,7,26,37,47,49,53]. Some of these methods exploit video data for learning, using various cues such as cycle-consistency in time [24,50,60]. Nevertheless, at test time, these models operate on a pair of frames, and do not consider wider temporal context, making them unsuitable for point tracking.
Feedforward models for dense tracking. Recently, there has been notable progress in developing feedforward neural network-based models for dense tracking (e.g., [12,13,18,25,33,59]). This advancement has been facilitated by the rise of new architectures and synthetic datasets that provide ground truth trajectory supervision [12,59]. TAP-Net [12] estimates the position of a query point by computing a cost volume for each target frame independently, followed by regressing the cost volume to a 2D coordinate and a visibility score. PIPs [18] revisits classical particle-based representation [40] by designing an MLP-based tracker that predicts tracklets in 8-frame window. To predict long-range tracks, PIPs is applied in a sliding-window fashion – an approach that is prone to drifting errors and cannot handle long-term occlusions. Aiming to extend the temporal field of view, PIPs^{++} [59] replaces the MLP-Mixer with a fully-convolutional 1D architecture. However, trajectories of different points are still predicted independently. Co-Tracker [25] aims to tackle this issue through a new Transformer-based architecture that jointly tracks multiple query points, and demonstrated impressive results on several benchmarks such as TAP-Vid-DAVIS. However, their temporal

field of view is still limited due to the expensive attention modules. TAPIR [13] combines TAP-Net and PIPs design in a two-stage framework: first, tracks are initialized using per-frame cost volume estimation, which are then refined similarly to [18]. Our work takes a different route in two fundamental ways: (i) all these methods are *trained from scratch* in a supervised manner. In contrast, we aim to leverage the rich and powerful internal representation learned by an external self-supervised image model, (ii) due to computational and memory requirements, these models are still limited in either their temporal or spatial field of view. We aggregate information across *all* video pixels via the trained weights of the tracker which is optimized to a specific video.

Recently, [43] proposed a self-supervised scheme for improving pre-trained supervised motion estimation models, by self-distilling cycle-consistent predictions. However, their method relies solely on the pre-trained model and does not consider any external priors, which is the focus of our approach.

Optimization-based tracking. The task of long-term tracking dates back to classical works that optimize motion globally over a video (e.g., [9, 38, 40, 58]). However, these methods are restricted to sparse or semi-dense tracking, and struggle to track under occlusions. Recently, Omnimotion [48] proposed a neural-based tracking framework that learns a bijective mapping between each point in the video and a canonical quasi-3D space. Their model is optimized per-video in a self-supervised manner, using optical flow and video reconstruction as supervision. Similarly, our method adopts test-time training, yet fundamentally differs from [48] in utilizing an external visual prior. As a result, DINO-Tracker outperforms Omnimotion in all benchmarks. Moreover, our optimization is more time-efficient as we only refine *pre-trained* features with a lightweight model.

DINO-ViT Features as local semantic descriptors. DINO [7] features were shown to effectively serve as dense and localized visual descriptors [2] for many tasks such as finding semantic correspondences [2,31,41,55,56], performing segmentation and part-segmentation [1, 2, 17, 32], transferring appearance in a semantically aware manner [45, 46], and aligning a set of semantically related images – establishing dense correspondences between them [16, 34]. Recently, Time-tuning [39] took DINO features to the temporal domain to improve the consistency of video segmentation. Our work is the first to harness the semantic prior of DINO for the task of dense, sub-pixel, long-range tracking in video.

3 Method

Given an input video $\{\mathbf{I}^t\}_{t=1}^T$, our goal is to train a tracker $\mathit{\Pi}$ that takes a query point $\mathbf{x_q}$ as input and outputs a set of position estimates $\{\hat{\mathbf{x}}^t\}_{t=1}^T$. As illustrated in Fig. 2, our framework follows the prevailing approach of extracting features, for both the query $\mathbf{x_q}$ and a target frame $\mathbf{I}^t$, and estimating the final position $\hat{\mathbf{x}}^t$ based on the maximal location in the cost volume. The core of our method is harnessing a pre-trained DINOv2-ViT model [35] in our feature extraction. DINO's pre-trained features provide our framework with an initial semantic and localized representation, yet, lacks temporal consistency and fine-grained local-

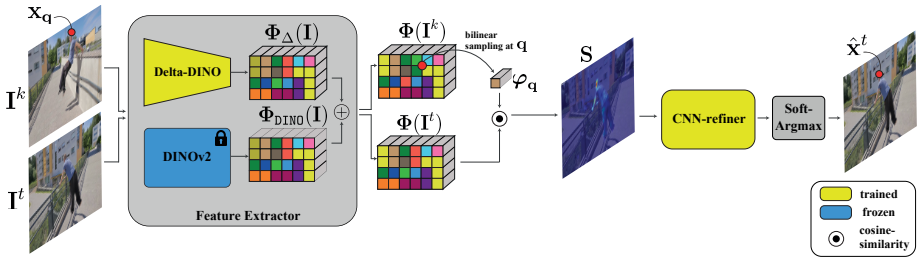

Fig. 2: *DINO-Tracker at inference:* Features are extracted from a reference frame $\mathbf{I}^k$, and a target frame $\mathbf{I}^t$. Our feature extractor consists of a *fixed* pre-trained DINOv2 model, and our CNN Delta-DINO model, which predicts a residual to DINO's features. To track a query point $\mathbf{x}_q \in \mathbf{I}^k$, we compute the cost volume between its sampled feature $\boldsymbol{\varphi}_q$, and the target feature map $\mathbf{\Phi}(\mathbf{I}^t)$. The resulting heatmap $\mathbf{S}$ is refined, and the final tracked position $\hat{\mathbf{x}}^t$ is estimated based on points in the vicinity of the maximal location.

ization required for accurate long-term tracking. We thus train Delta-DINO – a feature extractor that predicts a residual to the pre-trained DINO features.

Our goal is to refine the features such that they can act as "trajectory embeddings", i.e., features sampled along a trajectory should converge to a unique representation, while preserving the original DINO prior. To this end, we formulate a new objective function that is used to train our tracker in a self-supervised manner, on a single input video. Our sources of supervision are: (i) pre-computed optical flow which provides us with pseudo ground truth short-term pixel-level correspondences, (ii) semantic feature-level correspondences extracted from raw DINO features, which are distilled into our refined feature space through a contrastive objective, and (iii) self-distillation losses aiming to sharpen the correlation between reliable correspondences distilled from our refined feature space. We next describe our tracking framework and supervision in detail.

3.1 DINO-Tracker

The core component of our framework is the Delta-DINO model, which predicts the residuals to *frozen* DINO features for frame $\mathbf{I}$ (Fig. 2). That is, our refined features $\mathbf{\Phi}(\mathbf{I}) \in \mathbb{R}^{H' \times W' \times C}$ are given by:

$$\mathbf{\Phi}(\mathbf{I}) = \mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I}) + \mathbf{\Phi}_\Delta(\mathbf{I}) \tag{1}$$

where $\mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I})$ are the pre-trained DINO features, and $\mathbf{\Phi}_\Delta(\mathbf{I})$ are the predicted residual features. We use a CNN-based model for Delta-DINO, to benefit from its inductive bias, i.e., encoding similar RGB patches across frames into similar feature representation. In addition, predicting a residual rather than directly fine-tuning DINO allows us to better retain its prior [57]. To stabilize our fine-tuning process, we zero-initialize our refiner.

Given a query point $\mathbf{x_q}$ in $\mathbf{I}^k$, we bilinearly-sample its feature: $\boldsymbol{\varphi}_\mathbf{q} = \mathbf{\Phi}(\mathbf{I}^k)[\mathbf{q}]$, where $\mathbf{q}$ is the rescaled coordinate of $\mathbf{x_q}$ in the feature map. We then compute the cost volume between $\boldsymbol{\varphi}_\mathbf{q}$ and a target feature map $\mathbf{\Phi}^t = \mathbf{\Phi}(\mathbf{I}^t)$ as follows:

$$\mathbf{S}(\mathbf{p}) = \text{cos-sim}(\boldsymbol{\varphi}_{\mathbf{q}}, \boldsymbol{\Phi}^t(\mathbf{p})) \quad \text{where} \quad \text{cos-sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^T \cdot \mathbf{b}}{||\mathbf{a}||_2 \cdot ||\mathbf{b}||_2}$$

Following [12], we input $\mathbf{S}$ to a small CNN-refiner network followed by a spatial softmax, resulting in the final heatmap $\mathbf{H}$. The final coordinate $\hat{\mathbf{x}}^t$ is computed by considering the points in the vicinity of the maximal location $\mathbf{p}_{max} \in \mathbf{H}$ and computing their weighted sum:

$$\hat{\mathbf{x}}^t = \frac{\sum_{\mathbf{p} \in \Omega} \mathbf{H}(\mathbf{p}) \cdot \mathbf{x}_{\mathbf{p}}}{\sum_{\mathbf{p} \in \Omega} \mathbf{H}(\mathbf{p})} \tag{2}$$

where $\Omega = \{\mathbf{p} : ||\mathbf{x}_{\mathbf{p}} - \mathbf{x}_{\mathbf{p}_{max}}||_2 \leq R\}$. Thus, the final output of our tracker is $\Pi(\mathbf{x}_{\mathbf{q}}, t) = \hat{\mathbf{x}}^t$, and the track of $\mathbf{x}_{\mathbf{q}}$ is $\mathcal{T}_q = \left\{\hat{\mathbf{x}}^t : \hat{\mathbf{x}}^t = \Pi(\mathbf{x}_{\mathbf{q}}, t), t = 1 \dots T\right\}$.

3.2 Self-Supervision

We train DINO-Tracker to match points along trajectories with supervising signals automatically extracted from the test video itself using RAFT optical flow and distilled feature correspondences.

Optical flow provides accurate, sub-pixel displacement information between consecutive frames. We extract short-term tracks by chaining these displacements over time. A point $\mathbf{x}^i$ from frame i is matched to $\mathbf{x}^j$ at frame j if the optical flow tracklet between them is cycle-consistent. At preprocessing, we compute the set of all optical flow correspondences $\Omega_{\texttt{flow}} = \left\{(\mathbf{x}^i, \mathbf{x}^j) \text{ cycle-consistent}\right\}$, which provide high-quality supervision for short tracklets. However, they are unsuitable for providing long-range supervision due to error accumulation and occlusions. Further implementation details can be found in supplementary materials (SM).

Feature correspondences. are used to supplement our training data. We extract feature correspondences from DINO and leverage them for additional supervision. Specifically, we extract reliable matches between pairs of feature maps $\boldsymbol{\Phi}_{\texttt{DINO}}(\mathbf{I}^i), \boldsymbol{\Phi}_{\texttt{DINO}}(\mathbf{I}^j)$ by detecting "best-buddy pairs", i.e., mutual nearest neighbors [11]. Formally, a pair of points $\{\mathbf{p}^i, \mathbf{p}^j\}$ are best-buddies (bb) if:

$$NN(\boldsymbol{\varphi}^i_{\texttt{DINO}}, \boldsymbol{\Phi}_{\texttt{DINO}}(\mathbf{I}^j)) = \boldsymbol{\varphi}^j_{\texttt{DINO}} \wedge NN(\boldsymbol{\varphi}^j_{\texttt{DINO}}, \boldsymbol{\Phi}_{\texttt{DINO}}(\mathbf{I}^i)) = \boldsymbol{\varphi}^i_{\texttt{DINO}} \tag{3}$$

where $NN(\boldsymbol{\varphi}, \boldsymbol{\Phi})$ is the nearest-neighbor of $\boldsymbol{\varphi}$ in feature map $\boldsymbol{\Phi}$. At preprocessing, we compute the set of all DINO best-buddies $\Omega_{\texttt{dino-bb}} = \left\{\left(\mathbf{p}^i, \mathbf{p}^j\right) \text{ DINO bb}\right\}$.

Additionally, during training, our refined features improve their representation and give rise to new reliable correspondences. We detect *new* best buddies (Eq. 3) using the refined features, $\boldsymbol{\varphi}^i, \boldsymbol{\varphi}^j$. The set of refined best buddies, $\Omega_{\texttt{rfn-bb}} = \left\{\left(\mathbf{p}^i, \mathbf{p}^j\right) \text{ refined bb}\right\}$, is constantly updated during training.

Importantly, these two sources of correspondences are complementary: while optical flow provides accurate *sub-pixel* matches for near-by frames, features' best-buddies are extracted on a *coarse* spatial grid but provide long-term matches. DINO-Tracker is optimized using both, enjoying the best of both worlds.

3.3 Objective

Given an input video and the correspondences obtained in Sec. 3.2, we train our model using the following loss terms.

Flow loss. To match our estimated tracks with the motion of the input video, we apply a flow loss $\mathcal{L}_{\texttt{flow}}$, which aligns the estimated positions with correspondences extracted from optical flow:

$$\mathcal{L}_{\texttt{flow}} = \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\Omega_{\texttt{flow}}} L_H(\Pi(\mathbf{x}^i,j),\mathbf{x}^j) + L_H(\Pi(\mathbf{x}^j,i),\mathbf{x}^i)$$

where $\Omega_{\texttt{flow}}$ is the set of optical flow correspondences computed during preprocessing, and L_H is Huber loss [22].

DINO Best-Buddies Loss. Given a best-buddy pair $\{\mathbf{p}^i,\mathbf{p}^j\}\in\Omega_{\texttt{dino-bb}}$, we aim to increase the correlation between their refined features $\{\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j\}$, while decreasing their correlation to other features using a contrastive loss [10]:

$$l(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j) = -\log\frac{\exp(\text{cos-sim}(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j)/\tau)}{\sum_{\mathbf{p}}\exp(\text{cos-sim}(\boldsymbol{\varphi}^i,\boldsymbol{\Phi}^j(\mathbf{p}))/\tau)}$$

where τ is a temperature parameter. Our DINO best-buddies loss is:

$$\mathcal{L}_{\texttt{dino-bb}} = \frac{1}{|\Omega_{\texttt{dino-bb}}|}\sum_{(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j)\in\Omega_{\texttt{dino-bb}}}\frac{1}{2}w^{ij}_{\texttt{dino-bb}}\left(l(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j)+l(\boldsymbol{\varphi}^j,\boldsymbol{\varphi}^i)\right)$$

where $w^{ij}_{\texttt{dino-bb}}$ weights the loss for the corresponding pair based on a confidence metric of the detected best-buddy pair. The confidence is measured based on the unimodality of the similarity distribution between the pair of frames and on the correlation of the feature pair (see more details in SM).

Refined Best-Buddies Loss. We apply a similar contrastive loss for refined best-buddies distilled during training $\{\mathbf{p}^i,\mathbf{p}^j\}\in\Omega_{\texttt{rfn-bb}}$:

$$\mathcal{L}_{\texttt{rfn-bb}} = \frac{1}{|\Omega_{\texttt{rfn-bb}}|}\sum_{(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j)\in\Omega_{\texttt{rfn-bb}}}\frac{1}{2}w^{ij}_{\texttt{rfn-bb}}\left(l(\boldsymbol{\varphi}^i,\boldsymbol{\varphi}^j)+l(\boldsymbol{\varphi}^j,\boldsymbol{\varphi}^i)\right)$$

where $w^{ij}_{\texttt{rfn-bb}}$ weights the loss based on the cosine-similarity of the feature pair.

Cycle-Consistency Loss. We also found it beneficial to encourage the preservation of cycle-consistent tracks produced by DINO-Tracker. A pair of points $\{\mathbf{x}^i,\mathbf{x}^j\}$ is considered cycle-consistent if $\mathbf{x}^j = \Pi\left(\mathbf{x}^i,j\right)$ and $||\Pi(\mathbf{x}^j,i)-\mathbf{x}^i||_2\leq\gamma$, where γ is a small error threshold. Our cycle-consistency loss is given by:

$$\mathcal{L}_{\texttt{rfn-cc}} = \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\Omega_{\texttt{rfn-cc}}}\frac{1}{2}w^{ij}_{\texttt{rfn-cc}}\left(L_H(\Pi(\mathbf{x}^i,j),\mathbf{x}^j)+L_H(\Pi(\mathbf{x}^j,i),\mathbf{x}^i)\right) \quad (4)$$

where $\Omega_{\texttt{rfn-cc}}$ are cycle-consistent coordinate pairs extracted during training, and $w^{ij}_{\texttt{rfn-cc}}$ weights each term according to the cycle error (see SM for details).

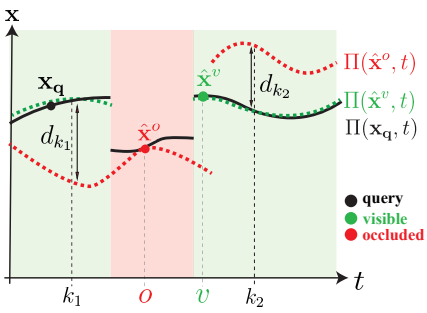

Fig. 3: *Visibility via trajectory agreement.* To determine the visibility of $\mathbf{x_q}$ at time $t=o$, we track $\hat{\mathbf{x}}^o$ across time and check the agreement between $\Pi(\hat{\mathbf{x}}^o, t)$ and $\Pi(\mathbf{x}, t)$. This is done by measuring d_{k_1}, d_{k_2} – displacements between the (black and red) tracks for anchor time steps k_1, k_2. Since these displacements are large, we classify $\mathbf{x_q}$ as occluded for $t = o$. For $t = v$, the track $\Pi(\hat{\mathbf{x}}^v, t)$ (green) agrees with $\Pi(\mathbf{x}, t)$, thus $\mathbf{x_q}$ is classified as visible for $t=v$.

Prior Preservation Loss. We apply regularization losses to preserve DINO's prior in our refined feature space: Specifically, we encourage each refined feature to: 1. maintain a high cosine similarity, and 2. have a close norm to its corresponding DINO feature. Given DINO features $\mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I})$ and refined features $\mathbf{\Phi}(\mathbf{I})$, our prior-preservation loss is defined as:

$$\mathcal{L}_{\texttt{prior}} = \frac{1}{H' \cdot W'} \cdot \sum_{\mathbf{p}} \underbrace{\left| 1 - \frac{||\mathbf{\Phi}(\mathbf{I})[\mathbf{p}]||_2}{||\mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I})[\mathbf{p}]||_2} \right|}_{\mathcal{L}_{\texttt{norm}}} + \underbrace{|1 - \text{cos-sim}\left(\mathbf{\Phi}(\mathbf{I})[\mathbf{p}], \mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I})[\mathbf{p}]\right)|}_{\mathcal{L}_{\texttt{angle}}}$$

Thus, our final objective is:

$$\mathcal{L} = \mathcal{L}_{\texttt{flow}} + \lambda_1 \mathcal{L}_{\texttt{dino-bb}} + \lambda_2 \mathcal{L}_{\texttt{rfn-bb}} + \lambda_3 \mathcal{L}_{\texttt{rfn-cc}} + \lambda_4 \mathcal{L}_{\texttt{prior}} \tag{5}$$

where λ_* sets the relative weights between the terms. We use a fixed set of λ_* in all our experiments. See SM for further implementation and complexity details.

3.4 Occlusion Prediction

Given an estimated trajectory $\mathcal{T}_\mathbf{q}$, our goal is to determine if the query point $\mathbf{x_q}$ is indeed visible at each time t. We do so based on trajectory agreement. That is, if $\mathbf{x_q}$ is visible at time $t = v$, tracking from $\hat{\mathbf{x}}^v \in \mathcal{T}_\mathbf{q}$ will give rise to the same trajectory, i.e., $\Pi(\mathbf{x_q}, k) \approx \Pi(\hat{\mathbf{x}}^v, k)$ for some frames k. This is illustrated by the agreement of the black $\mathcal{T}_\mathbf{q}$ and the green track in Fig. 3. In contrast, if at time $t = o$ $\mathbf{x_q}$ is occluded, tracking from $\hat{\mathbf{x}}^o \in \mathcal{T}_\mathbf{q}$ will result with a different trajectory, i.e., $\|\Pi(\mathbf{x_q}, k) - \Pi(\hat{\mathbf{x}}^o, k)\| = d_k$ will be large. This is illustrated by the red trajectory. We measure this trajectory agreement on a few anchor frames $k = k_1, k_2, \ldots$ as illustrated in the figure. To conclude, $\hat{\mathbf{x}}^t$ is deemed visible if $d_{k_1}, d_{k_2}, \ldots$ are small and the feature $\boldsymbol{\varphi}^t$ is similar to $\boldsymbol{\varphi}_\mathbf{q}$. More technical details on selecting anchor frames and various thresholds can be found in SM.

4 Results

Benchmarks. We evaluate our method on known benchmarks containing annotated tracks on real videos: (i) **TAP-Vid-DAVIS** [12], contains 30 object-centric

Table 1: *Quantitative comparison.* We compare our performance to all the baselines on TAP-Vid-DAVIS, TAP-Vid-Kinetics [12] and BADJA [4] using the metrics described in Sec. 4. Methods that do not predict occlusions lack OA and AJ. Our test-time self-supervised tracker performs on-par with SOTA supervised [13, 25], while substantially outperforming the SOTA test-time training method [48]. Higher is better for all metrics.

Method	DAVIS-256			DAVIS-480			Kinetics-256			Kinetics-480			BADJA	
	δ^x_{avg}	OA	AJ	δ^x_{avg}	OA	AJ	δ^x_{avg}	OA	AJ	δ^x_{avg}	OA	AJ	δ^{seg}	δ^{3px}
RAFT [44]	56.7	–	–	66.7	–	–	50.4	–	–	60.5	–	–	45.0	5.8
DINOv2 [35]	61.4	–	–	64.7	–	–	60.3	–	–	61.0	–	–	62.8	8.4
TAP-Net* [12]	53.4	81.4	38.4	66.4	79.0	46.0	61.7	86.6	48.5	67.1	81.5	47.7	45.4	9.6
PIPs$^{++\star}$ [59]	71.5	–	–	73.6	–	–	68.2	–	–	70.8	–	–	59.0	9.8
TAPIR* [13]	74.7	**89.4**	62.8	77.3	**89.5**	**65.7**	69.5	89.1	57.3	69.8	86.7	57.5	68.7	10.5
Co-Tracker* [25]	**79.2**	89.3	**65.1**	79.4	**89.5**	65.6	72.9	88.9	**59.9**	72.8	88.9	59.8	64.0	11.2
Omnimotion† [48]	67.5	85.3	51.7	74.1	84.5	58.4	69.2	**89.2**	55.0	–	–	–	45.2	6.9
Ours†	78.2	87.5	62.3	**80.4**	88.1	64.6	**73.3**	88.5	59.7	**74.3**	**89.2**	**60.9**	**72.4**	**14.3**

* – supervised. † – test-time training.

videos from [36] of 34-104 frames. (ii) **TAP-Vid-Kinetics** contains 1189 videos of 250 frames each taken from [8], depicting mostly human activity under both camera and objects' motion. We use the same set of 100 sampled videos used in [48] for our evaluation. (iii) **BADJA** [4], contains 9 videos, at 480px resolution, depicting naturally moving animals with ground truth annotated keypoints.

Metrics. The following metrics are measured for TAP-Vid benchmarks [12]:

– *Position accuracy* δ^x_{avg} measures the average position accuracy of visible points: $\delta^x_{avg} = \mathbb{E}_x(\delta^x)$, where each δ^x is the fraction of predicted points within the x pixels neighborhood of the ground-truth position, where $x \in \{1, 2, 4, 8, 16\}$.

– *Occlusion Accuracy (OA)* is the visibility classification accuracy.

– *Average Jaccard (AJ)* jointly measures position and occlusion accuracy.

The following metrics are used for evaluating BADJA:

– δ^{seg} measures the accuracy of the tracked keypoint within the distance of $0.2\sqrt{A}$ of the ground-truth, where A is the area of the foreground object.

– δ^{3px} measures the accuracy within a threshold of 3px.

Baselines. We compare to SOTA supervised feedforward trackers: PIPs^{++} [59], TAP-Net [12], TAPIR [13] and Co-Tracker [25], as well as the test-time optimization tracker Omnimotion [48]. We also consider two additional baselines: RAFT [44], in which tracking is performed by chaining optical flow displacements between consecutive frames, and DINOv2 [35], using nearest neighbor matching between raw DINOv2 features. Since DINO features are computed at low resolution, the position in RGB space is obtained using a weighted sum around the nearest neighbor (Eq. 2). See SM for implementation details. Since Omnimotion requires hours of training for each video, in Kinetics, we evaluate only on 256-resolution, where pre-trained weights are available.

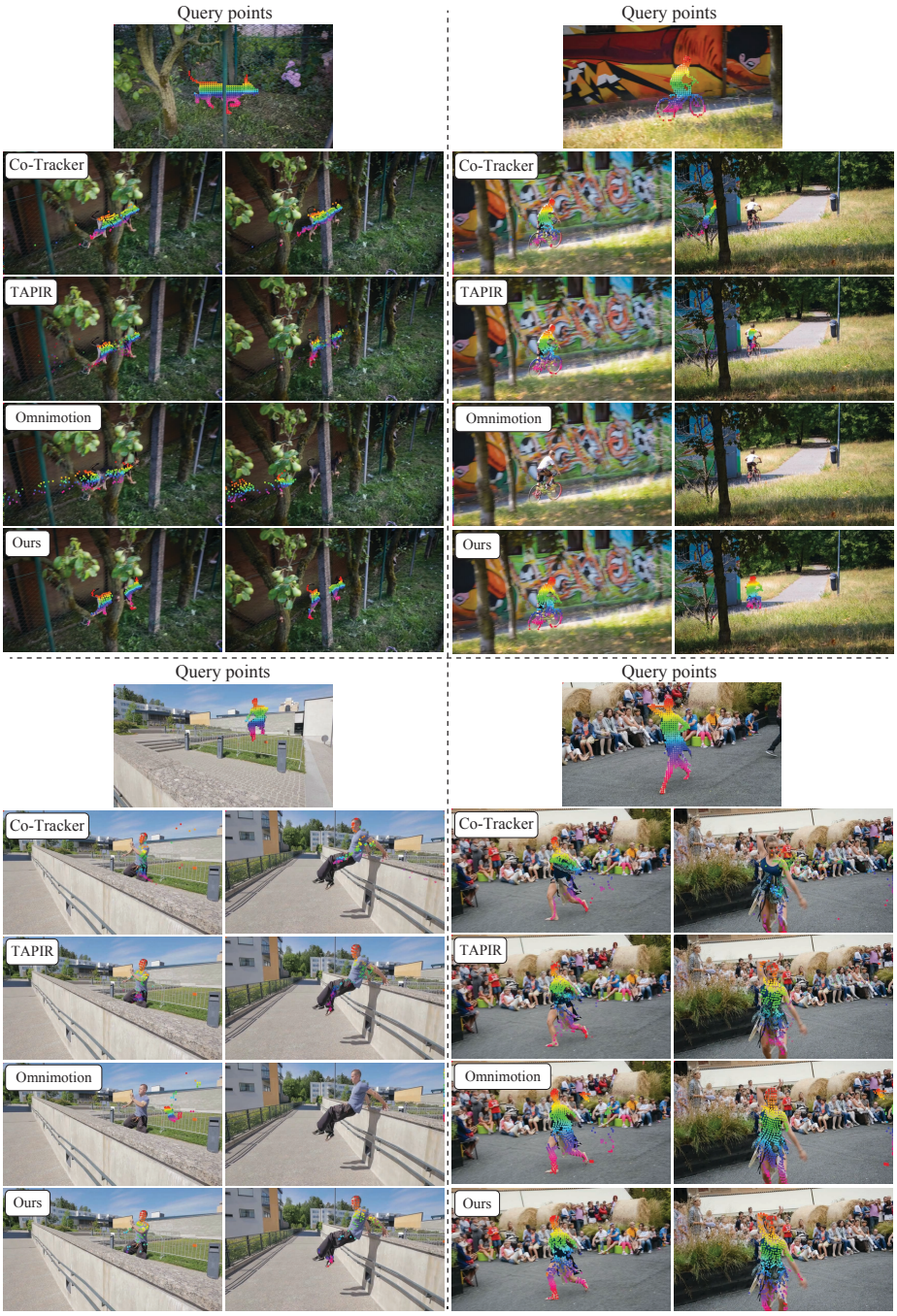

Fig. 4: *Qualitative results on TAP-Vid-DAVIS (480)* Query points are color-coded on a reference frame (top). Our method exhibits better association of tracks across occlusions compared to SOTA trackers. Full videos and additional results are in the SM on our website.

4.1 Comparisons

Table 1 reports our performance on TAP-Vid benchmarks (for both 256px and 480px frame resolution) and BADJA (see SM for details of evaluation). As seen, raw DINOv2 is a surprisingly strong baseline: on DAVIS-256, it outperforms RAFT and even TAP-Net, which is a supervised tracker.

Our method consistently outperforms all baselines on position accuracy (δ^x_{avg}) on TAP-Vid, apart from Co-Tracker on DAVIS-256. Generally, all methods perform better on higher resolution. In our case, this is expected given the performance of raw DINOv2. Notably, compared to Omnimotion (the only test-time optimization competitor), our method exhibit a significant boost in performance across all benchmarks. This makes our method SOTA among self-supervised baselines, and demonstrate the power of combining test-time training with external priors. In terms of our occlusion prediction (OA), our performance is on-par with competitors, including supervised ones trained on ground-truth labels.

Figure. 4 shows sample qualitative results on DAVIS-480. The objects in the top two videos are fast moving and are repeatedly occluded. As seen, all competitors struggle tracking through these occlusions, often tracking points to visually similar yet semantically unrelated regions (e.g. foreground points tracked to the background). Our results depict more semantically consistent trajectories. The bottom videos depict articulated objects and self-occlusion – a particularly challenging scenario for all methods. Here too, our method tracks more persistently the foreground objects (e.g. man's upper-body, woman's hands).

Our results on BADJA, as seen in Table 1, are SOTA in both δ^{seg} and δ^{3px} metrics. Fig. 5 illustrates the prediction w.r.t. ground-truth for sample videos.

Tracking across occlusions. As discussed in Sec. 3.2, DINO's features provide complementary information to pixel-level optical flow, which allows our method to reason about correspondences across distant frames. This grants our method an advantage in tracking across long-term occlusions. To quantify this, we split TAP-Vid-DAVIS into three sets of videos with an increasing rate of occlusion. Specifically, for each trajectory, we compute the ratio of the number of occluded points to the trajectory length. Figure 6 reports the performance of our method and the baselines as a function of the occlusion rate. As seen, DINO-Tracker performs significantly better in high occlusion rate due to the incorporated visual prior, enabling it to associate points across long-term occlusions.

4.2 Ablations and Analysis

We quantitatively ablate our key design choices in Table 2. To quantify the contribution of DINO's prior, we compare our full framework to a baseline in which $\mathbf{\Phi}_{\texttt{DINO}}(\mathbf{I}) = \mathbf{0}$, i.e., training an encoder from scratch *without* DINO, without $\mathcal{L}_{\texttt{prior}}, \mathcal{L}_{\texttt{dino-bb}}$ losses in Eq. 5. This baseline relies on appearance-based features only and performs dramatically worse in all metrics (*w/o DINO* in Tab. 2).

We further consider a baseline in which our Delta-DINO CNN is replaced by fine-tuning DINOv2 weights using LoRA [20], using the same objective (Eq. 5). As seen in Tab. 2, the performance significantly drops. We found that this approach produces jittery trajectories, and that the heatmaps are less localized.

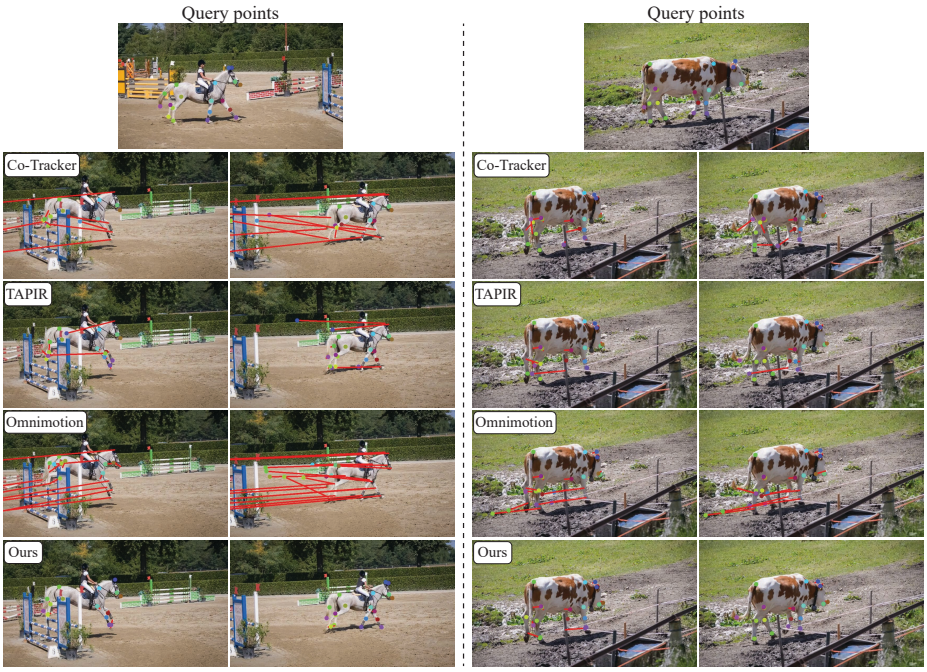

Fig. 5: *Sample results on BADJA w.r.t. ground truth.* Query points are color-coded on the frame at the top. Tracked points are marked on the target frames. Red lines indicate tracking *errors* w.r.t. the ground truth positions.

This is seen in Fig. 7 where we show the predicted tracks and correlation maps (cost volumes) for a couple representative examples. In contrast, our framework benefits from the inductive bias of CNN's as it learns to correlate similar RGB patches/neighborhoods, while also benefiting from the smoothness of CNN features. Another advantage of ours over LoRA is efficiency in memory and time.

In addition, Fig. 7 includes the results of tracking based on *raw* DINOv2 features. As seen, our optimization refines this initialization, leading to highly-localized heatmaps, even in ambiguous regions (multiple fish eyes, paraglider body). This is also evident in Fig. 1, where we used t-SNE [30] to visualize raw DINOv2 features and our refined features along *ground-truth* tracks. DINOv2 features along trajectories are often "spread out" and are intertwined with features from other trajectories. In contrast, our refined features along a trajectory are distinctly clustered, making tracking more robust and accurate.

Finally, we quantify the contribution of each loss term in our objective (last rows of Tab. 2). Removing each term results in a drop in tracking performance and highlights their contribution. Interestingly, w/o $\mathcal{L}_{\texttt{flow}}$ reduces positional accuracy only by 2%. This shows the effectiveness of combining DINO prior with our self-supervision and feature refinement for accurate tracking.

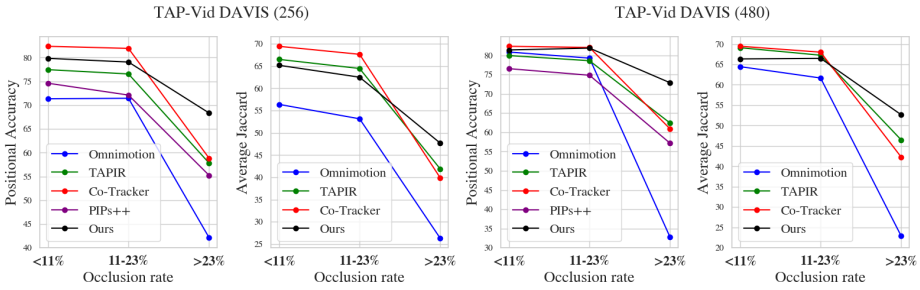

Fig. 6: *Tracking performance by occlusion rate.* We group test videos from TAP-Vid DAVIS into three sets according to occlusion rate (estimated using ground-truth visibility annotations). Positional accuracy and Average Jaccard are reported for each set separately. While the performance of all methods decreases as the occlusion rate increases, our DINO-Tracker exhibits a smaller gap and outperforms all methods with a large margin under a high occlusion rate. This demonstrates the benefit of harnessing the semantic information encoded in DINO's pre-trained features. Omnimotion [48], which solely relies on optical flow and video reconstruction, struggles in this case.

DINO features are the cornerstone of our framework. But which DINO features should we use? Tab. 3 shows track position accuracy for different choices of DINOv2 ViT-L/14 facets. Using tokens extracted from the 16^{th} layer performs the best, and we use these DINO features in all experiments.

5 Discussion and Conclusions

We presented DINO-Tracker for dense point-tracking, which combines test-time training on a single video with the power of external priors of a pre-trained DINO model. We introduced a new optimization-based framework that harnesses DINO's internal representation, while adapting it to the task of point tracking in a self-supervised manner. We demonstrated that our CNN-based design provides implicit smoothness prior effective for tracking. We demonstrated that our design effectively preserves DINO's prior and provides implicit smoothness prior.

Regarding limitations, while our method excels in associating points *across* long-term occlusions, we do not model trajectories *behind* occluders. Previous methods achieve this using synthetic data for supervision, or lifting tracking into 3D. However, a simple interpolation technique such as cubic spline can give plausible tracks during occlusion (see SM for examples). Furthermore, we observed that in challenging videos where there are multiple semantically-similar objects and almost no optical flow supervision, trajectories may jump from one object to another. This is because DINO is mostly dominated by semantic information.

We demonstrated the strengths of our DINO-Tracker through extensive evaluation and showed its superiority in associating points across long-term occlusions. We hope that our work will trigger more research in leveraging self-supervised representation learning for dense tracking in video.

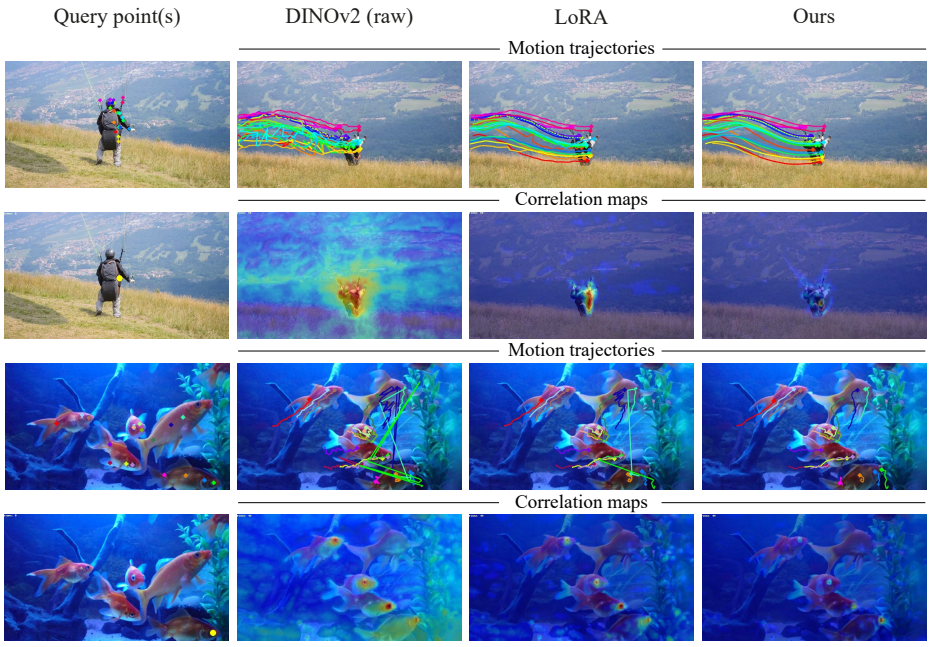

Fig. 7: *Comparing DINO-Tracker to (i) raw DINOv2 tracking, (ii) LoRA fine-tuning of DINOv2 for tracking.* For each example, the top row shows color-coded query points and the corresponding tracks. The second row shows the correlation maps (cost volumes) between a single query point (marked in yellow) and all features of the target frame. Raw and LoRA features are not well localized and are ambiguous for semantically similar objects (e.g., eyes of the fish), yielding imprecise tracks. In contrast, our refined features are well localized and better resolve ambiguities.

Table 2: *Ablation study.* Removing one key component of our method at a time and reporting performance on TAP-Vid-DAVIS videos. $\mathcal{L}_{\texttt{rfn}}$ is the combination of the losses $\mathcal{L}_{\texttt{rfn-bb}}$ and $\mathcal{L}_{\texttt{rfn-cc}}$.

	DAVIS-480		
	δ^x_{avg}	OA	AJ
w/o DINO	71.4	79.7	51.0
LoRA tune	73.2	84.8	58.0
w/o $\mathcal{L}_{\texttt{prior}}$	79.2	84.8	61.0
w/o $\mathcal{L}_{\texttt{rfn}}$	79.6	85.4	63.2
w/o $\mathcal{L}_{\texttt{dino-bb}}$	78.2	87.0	61.9
w/o $\mathcal{L}_{\texttt{flow}}$	78.3	87.2	62.0
Ours	**80.4**	**88.1**	**64.6**

Table 3: *DINO's feature layer ablation.* We evaluate tracking performance using DINOv2 ViT-L/14 features extracted from different layers and facets. We report track position accuracy (δ^x_{avg}) on TAP-Vid-DAVIS 480. Based on these results we use tokens extracted from the 16^{th} layer.

layer	tokens	queries	keys	values
12^{th}	61.1	51.1	50.0	62.4
16^{th}	**64.7**	48.8	46.7	63.9
20^{th}	63.8	56.9	56.0	64.0
23^{rd}	59.9	58.5	58.0	60.0

Acknowledgements

We would like to thank Rafail Fridman for his insightful remarks and assistance. We would also like to thank the authors of Omnimotion for providing the trained weights for TAP-Vid-DAVIS and TAP-Vid-Kinetics videos. The project was supported by an ERC starting grant OmniVideo (10111768), by Shimon and Golde Picker, and by the Carolito Stiftung.

Dr. Bagon is a Robin Chemers Neustein AI Fellow. He received funding from the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center and MBZUAI-WIS Joint Program for AI Research.

References

1. Aflalo, A., Bagon, S., Kashti, T., Eldar, Y.C.: Deepcut: Unsupervised segmentation using graph neural networks clustering. 2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW) pp. 32–41 (2022)
2. Amir, S., Gandelsman, Y., Bagon, S., Dekel, T.: Deep vit features as dense visual descriptors. ECCVW What is Motion For? (2022)
3. Bian, Z., Jabri, A., Efros, A.A., Owens, A.: Learning pixel trajectories with multiscale contrastive random walks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6508–6519 (2022)
4. Biggs, B., Roddick, T., Fitzgibbon, A., Cipolla, R.: Creatures great and SMAL: Recovering the shape and motion of animals from video. In: ACCV (2018)
5. Black, M.J., Anandan, P.: A framework for the robust estimation of optical flow. 1993 (4th) International Conference on Computer Vision pp. 231–236 (1993)
6. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. International journal of computer vision **61**, 211–231 (2005)
7. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
8. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4724–4733 (2017)
9. Chang, J., Wei, D., III, J.W.F.: A video representation using temporal superpixels. 2013 IEEE Conference on Computer Vision and Pattern Recognition pp. 2051–2058 (2013)
10. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)
11. Dekel, T., Oron, S., Rubinstein, M., Avidan, S., Freeman, W.T.: Best-buddies similarity for robust template matching. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2021–2029 (2015)
12. Doersch, C., Gupta, A., Markeeva, L., Continente, A.R., Smaira, K., Aytar, Y., Carreira, J., Zisserman, A., Yang, Y.: Tap-vid: A benchmark for tracking any point in a video. In: NeurIPS Datasets Track (2022)
13. Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., Zisserman, A.: Tapir: Tracking any point with per-frame initialization and temporal refinement. ICCV (2023)

14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
15. Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. IEEE International Conference on Computer Vision (ICCV) pp. 2758–2766 (2015)
16. Gupta, K., Jampani, V., Esteves, C., Shrivastava, A., Makadia, A., Snavely, N., Kar, A.: Asic: Aligning sparse image collections. In: ICCV (2023)
17. Hamilton, M., Zhang, Z., Hariharan, B., Snavely, N., Freeman, W.T.: Unsupervised semantic segmentation by distilling feature correspondences. In: International Conference on Learning Representations (2022)
18. Harley, A.W., Fang, Z., Fragkiadaki, K.: Particle video revisited: Tracking through occlusions using point trajectories. In: ECCV (2022)
19. Horn, B.K., Schunck, B.G.: Determining optical flow. Artificial Intelligence **17**(1), 185–203 (1981)
20. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022)
21. Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K.C., Qin, H., Dai, J., Li, H.: Flowformer: A transformer architecture for optical flow. ArXiv **abs/2203.16194** (2022)
22. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics **35**, 492–518 (1964)
23. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1647–1655 (2016)
24. Jabri, A., Owens, A., Efros, A.A.: Space-time correspondence as a contrastive random walk. Advances in Neural Information Processing Systems (2020)
25. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: CoTracker: It is better to track together (2023)
26. Li, X., Liu, S., De Mello, S., Wang, X., Kautz, J., Yang, M.H.: Joint-task self-supervised learning for temporal correspondence. Advances in Neural Information Processing Systems **32** (2019)
27. Liu, C., Yuen, J., Torralba, A.: Sift flow: Dense correspondence across scenes and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 978–994 (2011)
28. Lowe, G.: Sift-the scale invariant feature transform. Int. J **2**(91-110), 2 (2004)
29. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2. p. 674–679. IJCAI'81, Morgan Kaufmann Publishers Inc. (1981)
30. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
31. Mariotti, O., Aodha, O.M., Bilen, H.: Improving semantic correspondence with viewpoint-guided spherical maps (2023)
32. Melas-Kyriazi, L., Rupprecht, C., Laina, I., Vedaldi, A.: Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 8354–8365 (2022)

33. Neoral, M., Šerých, J., Matas, J.: MFT: Long-term tracking of every pixel. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6837–6847 (2024)
34. Ofri-Amar, D., Geyer, M., Kasten, Y., Dekel, T.: Neural congealing: Aligning images to a joint semantic atlas. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 19403–19412 (2023)
35. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023)
36. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 davis challenge on video object segmentation. arXiv:1704.00675 (2017)
37. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. Advances in neural information processing systems **31** (2018)
38. Rubinstein, M., Liu, C.: Towards longer long-range motion trajectories. In: British Machine Vision Conference (2012)
39. Salehi, M., Gavves, E., Snoek, C.G.M., Asano, Y.M.: Time does tell: Self-supervised time-tuning of dense image representations. ICCV (2023)
40. Sand, P., Teller, S.J.: Particle video: Long-range motion estimation using point trajectories. International Journal of Computer Vision **80**, 72–91 (2006)
41. Shtedritski, A., Vedaldi, A., Rupprecht, C.: Learning universal semantic correspondences with no supervision and automatic data curation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops. pp. 933–943 (October 2023)
42. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8934–8943 (2017)
43. Sun, X., Harley, A.W., Guibas, L.J.: Refining pre-trained motion models. In: Proceedings of the IEEE International Conference on Robotics and Automation (2024)
44. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European Conference on Computer Vision (ECCV). pp. 402–419 (2020)
45. Tumanyan, N., Bar-Tal, O., Amir, S., Bagon, S., Dekel, T.: Disentangling structure and appearance in vit feature space. ACM Trans. Graph. (nov 2023)
46. Tumanyan, N., Bar-Tal, O., Bagon, S., Dekel, T.: Splicing vit features for semantic appearance transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10748–10757 (2022)
47. Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K.: Tracking emerges by colorizing videos. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
48. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: International Conference on Computer Vision (2023)
49. Wang, Q., Zhou, X., Hariharan, B., Snavely, N.: Learning feature descriptors using camera pose supervision. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. pp. 757–774. Springer (2020)
50. Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: CVPR (2019)

51. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D.: Gmflow: Learning optical flow via global matching. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 8111–8120 (2021)
52. Xu, J., Ranftl, R., Koltun, V.: Accurate optical flow via direct cost volume processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1289–1297 (2017)
53. Xu, J., Wang, X.: Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10075–10085 (2021)
54. Zhai, M., Xiang, X., Lv, N., Kong, X.: Optical flow and scene flow estimation: A survey. Pattern Recognition **114**, 107861 (2021)
55. Zhang, J., Herrmann, C., Hur, J., Cabrera, L.P., Jampani, V., Sun, D., Yang, M.H.: A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence (2023)
56. Zhang, J., Herrmann, C., Hur, J., Chen, E., Jampani, V., Sun, D., Yang, M.H.: Telling left from right: Identifying geometry-aware semantic correspondence (2023)
57. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models (2023)
58. Zhao, W., Liu, S., Guo, H., Wang, W., Liu, Y.: Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild. In: European Conference on Computer Vision (2022)
59. Zheng, Y., Harley, A.W., Shen, B., Wetzstein, G., Guibas, L.J.: Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In: ICCV (2023)
60. Zhou, T., Krahenbuhl, P., Aubry, M., Huang, Q., Efros, A.A.: Learning dense correspondence via 3d-guided cycle consistency. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 117–126 (2016)