Learning to LEAP: Efficient Dense Point Tracking by Focusing Where It Matters

Chenzhi Zhao[1], **Wufan Wang**[1*], **Bo Zhang**[1*], **Wendong Wang**[1]

[1]State Key Laboratory of Networking and Switching Technology, School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications
chenzhi_zhao@bupt.edu.cn, wufanwang@bupt.edu.cn, zbo@bupt.edu.cn, wdwang@bupt.edu.cn

Abstract

Tracking Any Point (TAP) is a foundational task in computer vision with broad applicability. The state-of-the-art self-supervised TAP method leverages a global matching transformer and contrastive random walks to learn point correspondences. However, its dense all-pairs attention and correlation volume computation tend to introduce irrelevant features and produce less informative training signals, degrading both learning efficiency and tracking accuracy. To address these limitations, we introduce LEAP-Track, a self-supervised TAP approach that computes the attention matrices and correlation volume over adaptively selected sparse pairs. It consists of two core designs: (1) Curriculum-based Sparse Attention (CSA), which dynamically focuses on the most relevant keys, promoting the learning of discriminative features; and (2) Progressive k-NN Transition (PkT), which reformulates the contrastive random walk to operate on an increasingly sparse k-NN affinity graph to reinforce the learning of the most informative correspondences. By integrating the above two designs into a two-stage training paradigm, LEAP-Track is shown both theoretically and empirically to effectively boost learning efficiency, achieving superior tracking accuracy over existing self-supervised TAP methods.

1 Introduction

The task of Tracking Any Point (TAP) (Doersch et al. 2022) serves as a fundamental cornerstone in computer vision, underpinning diverse applications such as 3D reconstruction (Feng* et al. 2025), robotic manipulation (Vecerik et al. 2024; Doersch et al. 2022), and controllable video generation (Geng et al. 2025). The goal is to estimate tracks of arbitrarily specified query pixels across the entire video, thereby establishing point correspondences over long time horizons. Recently, fully-supervised approaches have achieved remarkable point tracking accuracy on the TAP task (Karaev et al. 2024; Li et al. 2024; Xiao et al. 2024). However, the inherent reliance of these methods on extensive manual annotations severely limits their scalability and cross-domain generalization. Consequently, significant research interest has shifted towards self-supervised methods that learn to track points directly from unlabeled videos.

The state-of-the-art self-supervised TAP approach (e.g., GMRW (Shrivastava and Owens 2024)) leverages the global matching transformer (Xu et al. 2022) and contrastive random walk to learn cycle-consistent tracks over video. Specifically, the global matching transformer first extracts inter-frame correlated features from consecutive video frames via stacked self-attention and cross-attention layers. The correlation volume of the extracted features is then computed to construct the probabilistic transition matrix, modeling random walks between points in adjacent frames to ensure cycle consistency. However, the *all pairs* computation paradigm in the construction of the attention matrices and correlation volume incurs two inherent limitations. On the one hand, since only a small fraction of points is relevant to the query point (as shown in Figure 1), computing the attention over all keys inevitably incorporates the features of irrelevant points, hindering the model from learning discriminative inter-frame correlated features. On the other hand, the correlation volume obtained by global pair-wise similarity computation can generate a vast number of less informative random walk paths, which tend to dominate the training process and overwhelm informative training signals. These limitations considerably impair the learning efficiency, leading to slow model convergence and degraded tracking accuracy.

To address these limitations, we propose to construct the attention matrices and correlation volume over *sparse pairs* instead of *all pairs*. This naturally raises the question of how to select these *sparse pairs*? We observe that, for a given query point, the relevant and informative points concerning it gradually stand out (i.e., exhibiting high similarity with the query point) in the attention matrix and correlation volume as training progresses (as shown in Figure 1), motivating an adaptive similarity-based selection of *sparse pairs*, to focus on learning the most relevant and informative pairs.

Building on this insight, this paper presents LEAP-Track (Learning Efficient Adaptive Point Tracking), a self-supervised TAP approach that boosts the learning efficiency by computing the attention matrices and correlation volume over adaptively selected *sparse pairs*. To implement the idea, LEAP-Track encompasses two core designs: (1) *Curriculum-based Sparse Attention (CSA)* that leverages curriculum learning to compute attention over only highly correlated keys to promote the learning of discriminative features. (2) *Progressive k-NN Transition (PkT)* that per-

*Corresponding Author

Figure 1: Evolution of attention patterns in GMRW during training. Given a query point in the source frame (red dot, left), we visualize cross-attention and correlation probability maps as we track to the target frame. Early in training, attention is dispersed across the image. As training progresses, attention focuses more on the target. This suggests that dense attention mechanisms and global random walks waste resources on non-informative areas, thereby weakening the learning signal.

forms the contrastive random walk on a k-NN affinity graph with progressively shrinking neighbor size k, to make the model focus on learning the most informative pairs. Finally, the above two designs are integrated into a two-stage training paradigm that first undergoes a short-time *all pairs* dense computation phase to warm up the model, and then switches to the adaptive *sparse pairs* computation phase for efficient learning. Extensive experiments conducted on several benchmarks demonstrate that LEAP-Track can significantly boost the learning efficiency, surpassing the state-of-the-art self-supervised TAP method in terms of tracking accuracy and model convergence speed.

2 Related Work

Transformers for Visual Correspondence Learning. Recently, transformers have been widely applied in visual correspondence learning due to their ability to model long-range dependencies. Pioneering work leverages transformer-like self- and cross-attention to reason about scene geometry and match sparse features via optimal transport (Sarlin et al. 2020) or as a functional query solved by a transformer (Jiang et al. 2021b). In dense optical flow estimation, this has challenged the dominant iterative cost-volume paradigm, leading to architectures that enhance the cost volume with a Transformer encoder (Huang et al. 2022) or replace it with a global matching transformer (Xu et al. 2022). The efficacy of the latter is shown by its generalization into a unified model for flow, stereo, and depth estimation (Xu et al. 2023). Furthermore, these powerful architectures have enabled a shift in learning paradigms, where a joint-tracking Transformer can be trained on real videos at scale using semi-supervision with pseudo-labels (Karaev et al. 2025), or a global matching Transformer can be trained via self-supervision by contrastive random walks (Shrivastava and Owens 2024). A common characteristic of these methods is their reliance on computing dense all-pairs attention, which hinders the learning of discriminative features due to the interference of irrelevant regions and is computationally demanding, motivating the exploration of sparse alternatives.

Self-Supervised Learning with Cycle-Consistency. A powerful self-supervisory signal for learning spatio-temporal correspondence from unlabeled video is cycle-consistency. Some approaches enforce consistency via local matching (Dwibedi et al. 2019) or exploit it as a self-supervisory signal for temporal association and cross-view matching (Qian et al. 2025), while hybrid strategies combine region-level and pixel-level objectives (Li et al. 2019) or employ spatial-then-temporal curricula (Li and Liu 2023). Alternatively, this principle is explored through the deterministic forward-backward tracking of TimeCycle (Wang, Jabri, and Efros 2019), and later reformulated as the more robust Contrastive Random Walk (CRW) (Jabri, Owens, and Efros 2020) on a space-time graph. The CRW framework has since been extended to dense pixel-level tasks via multiscale hierarchies (Bian et al. 2022), enhanced with intra-frame spatial relations for better instance discrimination (Zhao, Jin, and Heng 2021), and adapted to global matching transformers for long-range tracking (Shrivastava and Owens 2024). However, the performance of these methods tends to be degraded by the uninformative training signals provided by the dense all-pairs affinity graphs. This motivates our work on sparse random walks for cycle-consistency.

3 Design of *LEAP-Track*

3.1 Problem Formulation

We adopt the Tracking Any Point (TAP) task definition from TAP-Vid (Doersch et al. 2022). Given a video $\mathcal{V} = \{I_t\}_{t=1}^{T}$ and a set of N queries $\mathcal{Q} = \{(\mathbf{q}_i, t_i^*)\}_{i=1}^{N}$, the goal is to predict a set of tracks $\{(\boldsymbol{\tau}_i, \mathbf{v}_i)\}_{i=1}^{N}$. Each track comprises a trajectory $\boldsymbol{\tau}_i = (\mathbf{p}_{i,t})_{t=1}^{T}$ and a visibility vector $\mathbf{v}_i = (v_{i,t})_{t=1}^{T}$, where $v_{i,t} = 1$ indicates the point is visible at frame t. A valid track must start at the query point, $\mathbf{p}_{i,t_i^*} = \mathbf{q}_i$, and maintain physical consistency. The query point can be set for any time step in the video, not just the first frame. In our notation, we use t to index the frame within a video sequence and s to denote the training step, up to a total of S.

3.2 Overview of LEAP-Track

As illustrated in Figure 2, the training pipeline begins with randomly selecting two frames per video and applying asymmetric transformations to avoid shortcuts, following GMRW (Shrivastava and Owens 2024). A CNN Encoder

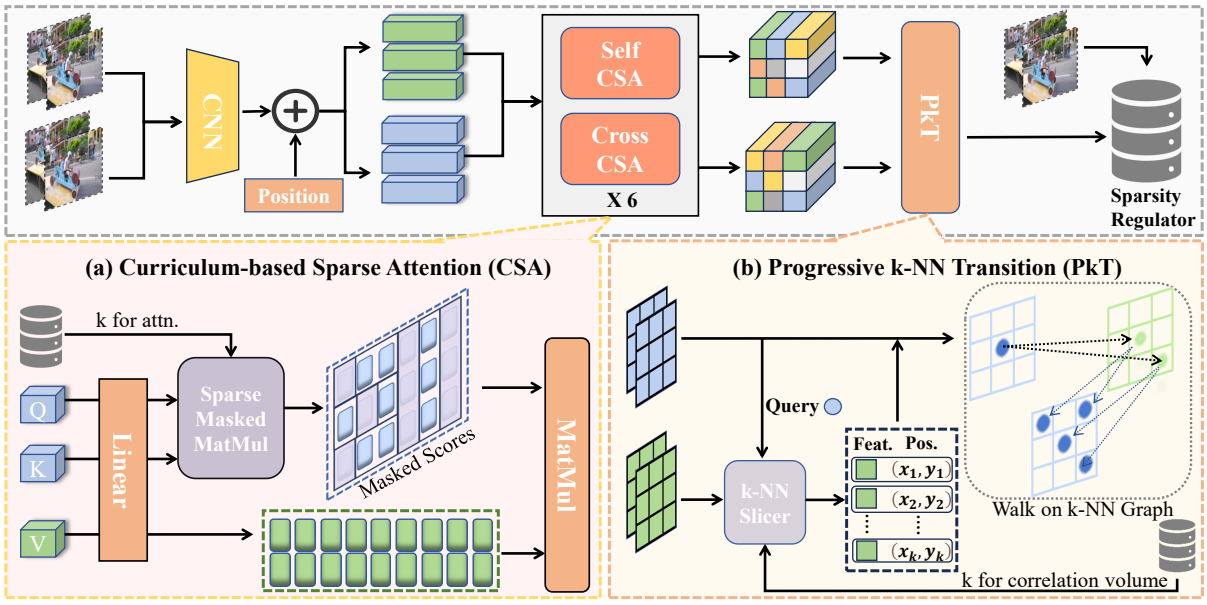

Figure 2: The framework of LEAP-Track. Features from a CNN backbone, augmented with positional encodings, are processed by two modules: (a) the Curriculum-based Sparse Attention (CSA), and (b) the Progressive k-NN Transition (PkT). A Sparsity Regulator progressively reduces the neighbor size k for both modules during training based on the loss condition.

extracts features, which, after positional encoding augmentation, are processed by a 6-layer Transformer with self- and cross-Curriculum-based Sparse Attention (CSA). Progressive k-NN Transition (PkT) then performs contrastive random walks on a sparse k-NN graph to compute cycle-consistency loss. An edge-aware regularizer adds a smoothness loss. Based on the total loss, a Sparsity Regulator progressively reduces the neighbor size for both modules.

3.3 Curriculum-based Sparse Attention (CSA)

In the standard dense attention mechanism, every query point attends to all key points in the other frame. Given feature maps from two frames, traditional dense attention computes an attention matrix over all $HW \times HW$ pairs:

$$\mathbf{A}_{\text{dense}} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{HW \times HW} \tag{1}$$

where $\mathbf{Q} = \mathbf{F}_t^{(q)}\mathbf{W}_Q$ and $\mathbf{K} = \mathbf{F}_t^{(k)}\mathbf{W}_K$ are the query and key projections, and d_k is the key dimension. This all-pairs computation forces the model to process features from a vast number of irrelevant or uninformative locations, hindering the model's ability to learn discriminative features that are essential for distinguishing targeted points.

To address this limitation, we introduce Curriculum-based Sparse Attention (CSA), which is designed to replace the standard dense attention matrix with a sparse one, thereby focusing the model's attention on the most relevant regions. Instead of attending to all keys, we dynamically prune the search space, compelling the model to learn from a sparse yet high-informative correspondence. For each query $\mathbf{q}_i$, CSA computes attention over only the top-$k(s)$ most relevant keys $\mathbf{k}_j$ within a local window $\mathcal{W}_i$:

$$\mathcal{N}_i^{k(s)} = \text{TopK}\{\mathbf{q}_i^T\mathbf{k}_j | j \in \mathcal{W}_i\} \tag{2}$$

where $\mathcal{W}_i$ denotes the set of positions within the same window as position i. The neighbor size $k(s)$ is determined by a scheduling function that decreases over training steps. The sparse attention matrix is thus:

$$\mathbf{A}_{\text{sparse}}[i,j] = \begin{cases} \frac{\exp(\mathbf{q}_i^T\mathbf{k}_j/\sqrt{d_k})}{\sum_{j' \in \mathcal{N}_i^{k(s)}} \exp(\mathbf{q}_i^T\mathbf{k}_{j'}/\sqrt{d_k})} & \text{if } j \in \mathcal{N}_i^{k(s)} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

To enable cross-window information exchange, we employ shifted window attention (Liu et al. 2021) in alternating layers, similar to Swin Transformers (Liu et al. 2021).

3.4 Progressive k-NN Transition (PkT)

Prevailing self-supervised methods based on Contrastive Random Walks (CRW) (Jabri, Owens, and Efros 2020; Shrivastava and Owens 2024) learn cycle-consistency by constructing a dense, all-pairs affinity graph that models probabilistic transitions between frames. A fundamental drawback of this approach is that it generates overwhelming random walk paths for uninformative pairs, which dilutes the learning signal from the few correct, cycle-consistent tracks, hindering the model's focus on the most informative pairs.

To rectify this, we propose the Progressive k-NN Transition (PkT) module, which reformulates the random walk to operate on a sparse k-Nearest-Neighbor (k-NN) graph, as detailed in Algorithm 1. PkT's key insight is that progressively reducing k enhances the model's confidence in its predictions. A larger k in early training allows for exploration of multiple hypotheses, while a smaller k later on promotes precise localization. For each point, we use efficient k-NN search (Jiang et al. 2021a) to find its k nearest neighbors in

Algorithm 1: Progressive k-NN Transition (PkT)

Input: Features $\{\mathbf{F}_t\}_{t=1}^{T}$
Parameter: Initial $k_{\max}$, final $k_{\min}$, training step s
Output: Cycle matrix $\mathbf{A}_{\text{cycle}}^{(k)}$

$k \leftarrow k(s)$ {Compute current neighbor size}
$\mathbf{A}_{\text{cycle}}^{(k)} \leftarrow \mathbf{I}$ {Initialize cycle matrix}
for $t = 1$ to T **do**
  $\mathbf{f}^{(t)}, \mathbf{f}^{(t+1)} \leftarrow \mathbf{F}_t, \mathbf{F}_{t+1}$ {Extract features}
  $\mathbf{Sim} \leftarrow \text{normalize}(\mathbf{f}^{(t)} \mathbf{f}^{(t+1)^T})$ {Compute similarities}
  for each spatial position i **do**
    $\mathcal{I}_i^{(k)} \leftarrow \arg\text{top-}k(\mathbf{Sim}[i,:])$ {Find k-NN}
    $\mathbf{P}_t^{(k)}[i,:] \leftarrow \text{softmax}(\mathbf{Sim}[i, \mathcal{I}_i^{(k)}]/\tau)$ {Compute sparse probabilities}
  end for
  $\tilde{\mathbf{P}}_t^{(k)} \leftarrow \text{expand}(\mathbf{P}_t^{(k)}, \mathcal{I}^{(k)})$ {Expand to full matrix}
  $\mathbf{A}_{\text{cycle}}^{(k)} \leftarrow \mathbf{A}_{\text{cycle}}^{(k)} \cdot \tilde{\mathbf{P}}_t^{(k)}$ {Update cycle matrix}
end for
return $\mathbf{A}_{\text{cycle}}^{(k)}$

the subsequent frame based on feature similarity:

$$\mathbf{Sim}_{ij} = \frac{\mathbf{f}_i^{(t)} \cdot \mathbf{f}_j^{(t+1)}}{\|\mathbf{f}_i^{(t)}\| \|\mathbf{f}_j^{(t+1)}\|} \tag{4}$$

$$\mathcal{I}_i^{(k)} = \arg\text{top-}k_{j \in \{1,\dots,N\}} \mathbf{Sim}_{ij} \tag{5}$$

where $\mathbf{f}_i^{(t)}$ is the feature at position i in frame t, and $\mathcal{I}_i^{(k)}$ denotes its k nearest neighbors.

We construct a sparse affinity matrix $\mathbf{P}_t^{(k)} \in \mathbb{R}^{N \times k}$ where $N = HW$ is the number of spatial positions, k is the neighbor size, and τ is the temperature parameter:

$$\mathbf{P}_t^{(k)}[i,j] = \begin{cases} \frac{\exp(\mathbf{Sim}_{i,\mathcal{I}_i^{(k)}[j]}/\tau)}{\sum_{j'=1}^{k} \exp(\mathbf{Sim}_{i,\mathcal{I}_i^{(k)}[j']}/\tau)} & \text{if } j \in \{1,\dots,k\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We determine the neighbor size with a stage-wise curriculum: $k(s) = \sum_{i=0}^{M-1} k_i \, \mathbf{1}_{[\eta_i, \eta_{i+1})}(s/S)$, where $\mathbf{1}$ is the indicator function, η_i is the set threshold. Instead of a global $(HW)^2$ matrix, PkT creates a compact $H \times W \times k^2$ cycle-consistency transition matrix, by chaining the expanded sparse matrices that are mapped back to the full spatial grid:

$$\mathbf{A}_{\text{cycle}}^{(k)} = \prod_{t=1}^{T} \tilde{\mathbf{P}}_t^{(k)} \tag{7}$$

During Evaluation, PkT computes a $HW \cdot k$ correlation volume for two frames, as shown in Figure 3.

3.5 Self-Supervised Training Objective

Our model is trained end-to-end using a composite loss that combines the cycle-consistency objective from PkT with an edge-aware smoothness regularizer, following GMRW (Shrivastava and Owens 2024):

$$\begin{aligned} \mathcal{L}_{\text{cycle}} &= -\frac{1}{|\mathcal{M}|} \sum\nolimits_{i \in \mathcal{M}} \log \left(\sum\nolimits_{j \in \mathcal{N}_i^{(k^2)}} \mathbf{A}_{\text{cycle}}[i,j] \cdot \mathbf{Y}[i,j] \right) \\ \mathcal{L}_{\text{smooth}} &= \frac{1}{2} \sum\nolimits_{d \in \{x,y\}} \mathbb{E}\left[\exp(-\alpha |I_{dd}|) \cdot \rho(f_{dd})\right] \\ \mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{cycle}} + \lambda_s \mathcal{L}_{\text{smooth}} \end{aligned} \tag{8}$$

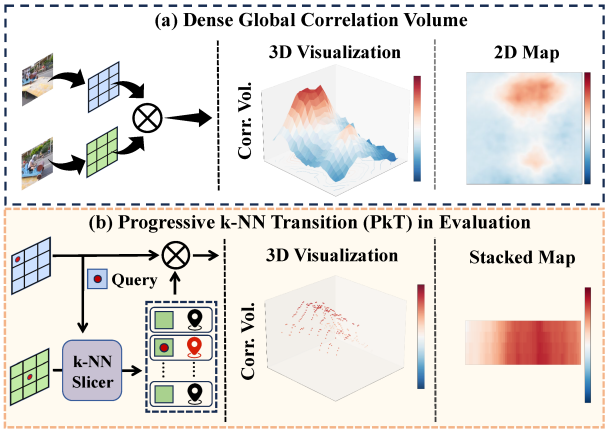

Figure 3: Correlation volume construction during evaluation. (a) Dense global correlation computes all-pairs similarity, resulting in a full $(HW)^2$ probability matrix. (b) PkT employs a k-NN Slicer to form a sparse affinity matrix of dimensions $HW \cdot k$, filtering out irrelevant matches.

where $\mathbf{Y}$ contains binary correspondence labels, $\mathcal{M}$ denotes valid pixels under the affine transformation, I_{dd} is the second-order derivative of the image, while f_{dd} is the second-order derivative of the flow field. A curriculum learning strategy is used for the smoothness loss weight λ_s, increasing linearly. For the Kinetics dataset, λ_s ranges from 0 to 1.0, and for the Kubric dataset, from 0 to 180.0.

4 Theoretical Analysis

In this section, we theoretically justify LEAP-Track's efficiency gains in training throughput and convergence speed. First, LEAP-Track reduces spatial complexity from quadratic, $O((HW)^2)$, to nearly linear, $O(HW \cdot k)$, where $k \ll HW$. This reduction can significantly speed up sample processing during the backward pass's gradient computation. Second, convergence is accelerated by improving gradient stability during optimization. Through optimization theory, we show that convergence bounds for adaptive optimizers like Adam/AdamW (Kingma and Ba 2014) are fundamentally constrained by the maximum gradient norm, $G_\infty = \max_{s \le S} |\nabla f_s(\theta)|_\infty$. A simplified regret bound $R(S)$ illustrates this relationship:

$$R(S) \le \mathcal{C}_v + \mathcal{C}_g \cdot G_\infty \tag{9}$$

where constants $\mathcal{C}_v$ and $\mathcal{C}_g$ are independent of the instantaneous gradient. This highlights that reducing the peak gradient magnitude G_∞ is critical for faster convergence.

Following the analysis in KVT (Wang et al. 2022), the gradient of loss $\mathcal{L}$ with respect to the attention layer's learnable weights (e.g., W_Q) is directly proportional to the variance of the input tokens, weighted by the attention scores:

$$\frac{\partial \hat{V}_l}{\partial W_{Q,ij}} \propto \text{Var}_{a_l}(X) \tag{10}$$

where $\text{Var}_{a_l}(X)$ is the covariance matrix of the input tokens X under the attention distribution a_l. In dense, global attention, a_l includes all N tokens, causing high variance from

Method	Train	Kubric			DAVIS			Kinetics			RGB-Stacking			Mean		
		AJ↑	δ^{vis}_{avg} ↑	OA↑	AJ↑	δ^{vis}_{avg} ↑	OA↑	AJ↑	δ^{vis}_{avg} ↑	OA↑	AJ↑	δ^{vis}_{avg} ↑	OA↑	AJ↑	δ^{vis}_{avg} ↑	OA↑
COTR	SL	40.1	60.7	78.6	35.4	51.3	80.2	19.0	38.8	57.4	6.8	13.5	79.1	25.3	41.1	73.8
RAFT-C	SL	41.2	58.2	86.4	30.7	46.6	80.2	31.7	51.7	84.3	42.0	56.4	91.5	36.4	53.2	85.6
RAFT-D	SL	61.8	79.1	87.9	34.1	48.9	76.1	72.1	85.1	92.1	50.6	66.9	85.5	54.6	70.0	85.4
Kubric-VFS-Like	SL	51.9	69.8	84.6	33.1	48.5	79.4	40.5	59.0	80.0	57.9	72.6	91.9	45.9	62.5	84.0
PIPs	SL†	59.1	74.8	88.6	42.0	59.4	82.1	35.3	54.8	77.4	37.3	51.0	91.6	43.4	60.0	84.9
TAP-Net	SL	65.4	77.7	93.0	38.4	53.1	82.3	46.6	60.9	85.0	59.9	72.8	90.4	52.6	66.1	87.7
TAPIR	SL†	84.7	92.1	95.8	59.9	72.8	88.4	57.2	70.1	87.8	62.7	74.6	91.6	66.1	77.4	90.9
CoTracker	SL†	—	—	—	64.8	79.1	88.7	—	—	—	65.4	80.3	85.1	65.1	79.7	86.9
OmniMotion (RAFT)	TTT†	—	—	—	51.7	67.5	85.3	55.1	69.6	89.6	77.3	86.7	93.2	61.4	74.6	89.4
CaDeX++	TTT†	—	—	—	59.4	77.4	85.9	—	—	—	75.0	86.8	93.2	67.2	82.1	89.6
The Chained Setting (-C)																
CRW-C	SSL	31.4	48.1	76.3	7.7	13.5	72.9	20.2	33.6	70.6	25.3	35.2	70.1	21.1	32.6	72.5
DIFT-C	SSL	28.3	45.2	69.0	18.1	33.0	68.8	19.8	33.7	68.7	13.2	21.9	56.3	19.9	33.5	65.7
ARFlow-C	SSL	52.3	68.1	81.4	35.0	51.8	79.7	27.3	44.3	79.5	33.0	47.2	**91.9**	36.9	52.8	83.1
Flow-Walk-C	SSL	53.2	69.9	83.9	35.2	51.2	**80.2**	**36.3**	**54.6**	**80.5**	41.1	55.2	91.4	41.5	57.7	**84.0**
GMRW-C	SSL	**54.2**	72.4	82.6	41.8	60.9	78.3	31.9	52.3	72.9	39.8	56.5	90.8	41.9	60.5	81.1
GMRW-C (`argmax`)	SSL	52.6	70.8	83.7	39.6	58.3	79.6	31.7	52.2	74.2	47.1	65.0	90.1	42.8	61.6	81.9
Ours - LEAP-Track-C	SSL	**54.2**	**72.6**	**84.5**	**42.3**	**61.6**	79.6	32.4	53.1	73.8	**55.4**	**73.2**	91.4	**46.1**	**65.1**	82.3
The Direct Setting (-D)																
CRW-D	SSL	35.8	52.4	80.9	23.6	38.0	77.2	21.9	36.8	70.4	13.1	23.0	83.4	23.6	37.5	78.0
DIFT-D	SSL	41.6	59.8	83.9	29.7	48.2	77.2	19.5	34.4	70.1	24.4	38.9	89.9	28.8	45.3	80.3
Flow-Walk-D	SSL	54.1	71.1	82.2	24.5	41.3	76.6	**39.6**	61.1	**75.0**	**66.3**	**82.7**	91.2	46.1	64.0	81.2
GMRW-D	SSL	51.4	71.7	83.9	30.3	49.4	77.3	36.3	59.2	71.0	56.4	74.1	90.9	43.6	63.6	80.8
GMRW-D (`argmax`)	SSL	50.7	70.6	83.9	29.8	47.2	**78.5**	36.0	58.9	72.0	57.1	74.8	90.9	43.4	62.9	81.3
Ours - LEAP-Track-D	SSL	**57.2**	**77.6**	**84.3**	**32.1**	**51.4**	78.3	37.3	**61.9**	71.3	58.8	76.4	**92.2**	**46.4**	**66.8**	**81.5**

Table 1: Quantitative results on the TAP-Vid benchmark. LEAP-Track achieves state-of-the-art (SOTA) performance among self-supervised methods on the Kubric and DAVIS datasets, and remains competitive across all benchmarks. Best SSL results are in bold. SL: Supervised, TTT: Test-Time Training, SSL: Self-Supervised. † denotes methods using multiple frames.

irrelevant points, which inflates G_∞^{dense} and slows convergence. Our CSA module addresses this by focusing on a limited set of the most relevant key features, reducing attention-weighted variance, significantly lowering $G_\infty^{\text{CSA}} \ll G_\infty^{\text{dense}}$.

Similarly, our PkT module reduces the number of paths considered in the random walk from $O((HW)^{2T-1})$ in a dense graph to $O(HW \cdot k^{2T-2})$ in a sparse k-NN graph. This exponentially prunes the search space, yielding a more stable learning signal with substantially lower variance:

$$\text{Var}(\nabla_{\text{PkT}}) \propto \left(\frac{k}{N}\right)^{2T-2} \text{Var}(\nabla_{\text{dense}}) \tag{11}$$

This drastic reduction in gradient variance leads to a more stable optimization landscape and faster convergence.

5 Experiment

5.1 Experimental Setup

Datasets and Metrics. The Kinetics-400 (Kay et al. 2017) and TAP-Vid-Kubric (Doersch et al. 2022) datasets are utilized in our method for self-supervised training. For performance comparison, we evaluate different methods on the full TAP-Vid benchmark (Doersch et al. 2022), which includes four diverse subsets: DAVIS (real-world, high-resolution video), Kinetics (large-scale, real-world clips), Kubric (complex synthetic physics), and RGB-Stacking (challenging texture-less objects). Following previous studies (Doersch et al. 2022; Li, Zhang, and Huang 2019; Jain et al. 2024), performance is evaluated on the following metrics: (1) *Occlusion Accuracy* (OA), which measures the accuracy of binary prediction on whether a point is visible or occluded. (2) *Positional Accuracy* (δ^{vis}_{avg}), which depicts the mean fraction of visible points within thresholds of $\{1, 2, 4, 8, 16\}$ pixels over the ground truth. (3) *Average Jaccard* (AJ), which is the mean Jaccard index for visible points evaluated over the same distance thresholds. (4) *Clips per second* (Clips/s), which measures the training throughput via processed video clips per second.

Training. We employ a two-stage training paradigm. The initial stage (9k iterations) uses dense attention to learn robust global features, followed by our main adaptive sparse phase. Loss stability condition triggers transitions in the sparse phase. We use AdamW (Loshchilov and Hutter 2019) optimizer with a learning rate of 10^{-4} and weight decay of 10^{-4}, scaling the learning rate linearly with the number of GPUs for distributed training. All models are trained by sampling frame pairs (t_1, t_2) to form palindromic sequences $\{t_1 \to t_2 \to t_1\}$ for self-supervised cycle consistency.

Evaluation. We adopt the evaluation protocol in previous work (Wang et al. 2023), which consists of two settings: The *chained* setting (*-C*) reconstructs long-range trajecto-

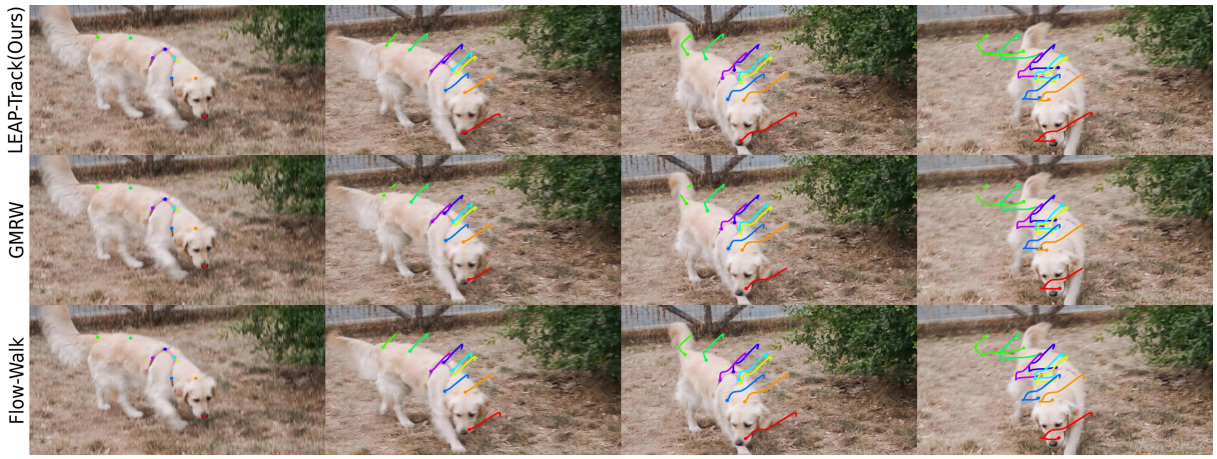

Figure 4: Qualitative comparison on a challenging DAVIS sequence. Our method produces visibly smoother and more accurate trajectories that adhere tightly to the deforming object. In contrast, both Flow-Walk and GMRW exhibit more jitter and lose track of key points over time (*dog's nose and back*). This highlights LEAP-Track's robustness to complex non-rigid motion.

ries by successively chaining pairwise correspondences between adjacent frames. The *direct* setting (*-D*) predicts point correspondences for each given query-target frame pair in a single pass. Fine-grained point correspondences are obtained by upsampling feature maps and then applying contrastive random walks (Neoral, Šerých, and Matas 2024). Following GMRW (Shrivastava and Owens 2024), we use a stride of 1 for the chained setting, 2 for the direct setting, and 4 for RGB-Stacking. For self-supervised models, cycle-consistency is used to infer occlusions.

5.2 Main Results

Quantitative Comparison. For performance evaluation, we compare LEAP-Track with supervised methods including COTR (Jiang et al. 2021b), RAFT (Teed and Deng 2020), Kubric-VFS-Like (Greff et al. 2022), PIP (Harley, Fang, and Fragkiadaki 2022), TAP-Net (Doersch et al. 2022), TAPIR (Doersch et al. 2023), CoTracker (Karaev et al. 2024), test-time-training methods including OmniMotion (Song et al. 2024) and CaDeX++ (Wang et al. 2023), as well as self-supervised methods including CRW (Jabri, Owens, and Efros 2020), DIFT (Tang et al. 2023), ARFlow (Liu et al. 2020), Flow-Walk (Bian et al. 2022) and GMRW (Shrivastava and Owens 2024). In addition to the original baseline method GMRW, we also evaluate an `argmax` variant of it that replaces the original weighted-average flow by a hard selection, which we find improves performance on low-texture datasets such as RGB-Stacking.

As shown in Table 1, LEAP-Track surpasses other self-supervised methods on most metrics over four datasets, yielding the best mean performance under both the chained and direct settings. Specifically, LEAP-Track achieves the highest mean AJ of 46.1, 46.4, and the highest mean $\delta^{\text{vis}}_{\text{avg}}$ of 65.1, 66.8 under chained and direct settings, respectively. Meanwhile, LEAP-Track establishes a new state-of-the-art on the challenging Kubric and DAVIS datasets, which demonstrates its ability to handle both complex synthetic physical environments and real-world videos. Although Flow-Walk achieves better performance than LEAP-Track on a few metrics, its multiscale coarse-to-fine architecture significantly increases the model complexity (Shrivastava and Owens 2024). In contrast, LEAP-Track can achieve superior overall performance than Flow-Walk with only a simple single-scale matching architecture, without requiring the extensive pretraining used by Flow-Walk. This demonstrates LEAP-Track's ability to learn highly discriminative feature representations despite its streamlined design.

Qualitative Comparison. For qualitative comparison, we present tracking results of a challenging sequence in Figure 4. From the figure, it can be observed that LEAP-Track estimates more stable and accurate trajectories compared to strong baselines like Flow-Walk (Bian et al. 2022) and GMRW (Shrivastava and Owens 2024). Notably, LEAP-Track also successfully tracks non-rigid motions of the dog's nose and back, where other methods fail, which demonstrates the superiority of our proposed method.

5.3 Analysis of Learning Efficiency

To further validate the capability of LEAP-Track to boost learning efficiency, Figure 5 compares the convergence speed and training throughput against the baseline method GMRW. As shown in Figure 5 (a)-(c), LEAP-Track exhibits a remarkably faster convergence speed and consistently outperforms GMRW on all metrics at every training checkpoint. In addition, Figure 5 (d) demonstrates the training throughput improvement on the Kinetics dataset. After the initial dense computation phase, LEAP-Track transitions to its adaptive sparse computation phase, boosting throughput from 0.3923 clips/s to a peak of 0.5806 clips/s, i.e., a **48% speed-up** over the baseline. The above results verify the ability of LEAP-Track to promote learning efficiency.

5.4 Model Ablations and Variations

We first conduct ablation studies on the Kubric dataset to dissect the contributions of LEAP-Track's core designs and

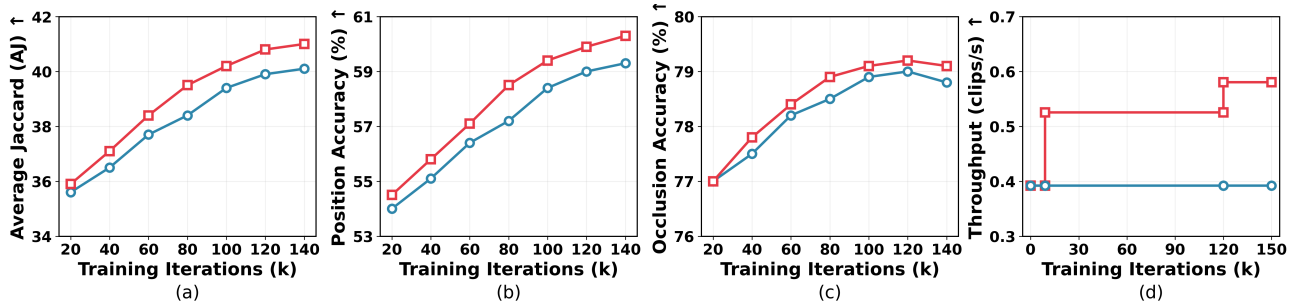

Figure 5: Learning efficiency comparison on the DAVIS dataset. LEAP-Track (red squares) consistently converges faster than GMRW (blue circles) across (a-c). In (d), the training throughput of LEAP-Track initially aligns with the baseline during the dense warm-up phase (gray area), but accelerates significantly upon transitioning to the adaptive sparse phase.

Method	CSA	PkT	Adaptive Train	Kubric		
				AJ↑	δ^{vis}_{avg} ↑	OA↑
Baseline				51.4	71.7	83.9
Baseline	✓			54.9	75.1	83.7
Baseline		✓		54.0	74.4	84.2
Baseline	✓	✓		55.3	75.3	84.2
LEAP-Track	✓	✓	✓	**57.2**	**77.6**	**84.3**
w/ Stride=4	✓	✓	✓	48.4	67.0	84.0
w/ Stride=1	✓	✓	✓	58.6	79.7	84.2
w/ Chained	✓	✓	✓	44.1	61.3	84.9
w/ Stride=4, Chained	✓	✓	✓	35.0	50.6	84.7
w/ Stride=1, Chained	✓	✓	✓	54.2	72.6	84.5

Table 2: Component and setting ablation on the Kubric test set. The top section shows the incremental benefit of each core module. The bottom section analyzes the final model's performance under different evaluation settings. The default configuration (bold) is Direct with an evaluation stride of 2.

analyze the impact of different evaluation settings. The top part of Table 2 validates our design, showing that CSA, PkT, and our adaptive training scheme are synergistic. Each component contributes incremental gains, with their combination in the full LEAP-Track model yielding the best results. The bottom part of Table 2 further lists model performance under different evaluation settings: Models evaluated under the direct setting generally achieve better performance since the direct setting inherently prevents error accumulation. In addition, model performance scales inversely with the evaluation stride, suggesting that the model's tracking performance is proportional to the feature resolution.

5.5 Sensitivity Analysis

CSA Hyperparameters. Figure 6 (a) illustrates how LEAP-Track's performance is affected by the proportion of keys used in the attention computation. The results show that as the proportion increases, the Average Jaccard score on the Kubric dataset first gradually increases and peaks at 75%, after which it slightly degrades. This suggests that while a larger attention scope helps, a too broad scope can incur noise from irrelevant points and impair model performance.

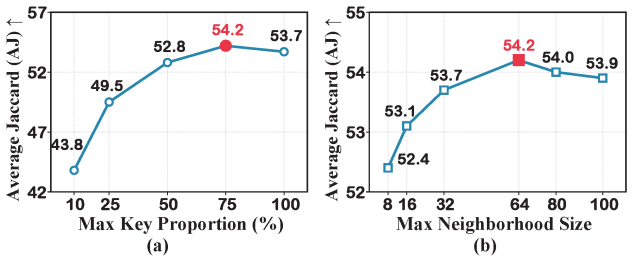

Figure 6: Hyperparameter Sensitivity Analysis on Kubric. (a) CSA module performance as a function of the maximum key proportion. The best Average Jaccard (AJ) score is achieved at 75%. (b) PkT module performance versus the neighbor size k. Performance peaks at $k = 64$.

PkT Hyperparameters. For the PkT module, we evaluate the impact of the maximum neighbor size k for the contrastive random walk. As illustrated in Figure 6 (b), performance on the Kubric dataset improves as k increases, peaking at $k = 64$. Further increasing k degrades the AJ score, suggesting that overly large neighbors incorporate less informative paths, which can harm tracking performance.

6 Conclusion

This paper introduces LEAP-Track, a self-supervised approach for the Tracking Any Point (TAP) task that addresses the inherent inefficiencies in the state-of-the-art self-supervised methods. By replacing all-pairs computations with adaptive sparse operations, LEAP-Track achieves superior learning efficiency and tracking accuracy. The proposed CSA module encourages the model to acquire more discriminative feature representations, while the PkT mechanism guides it to focus on the most informative correspondences. Integrated into a two-stage training paradigm, these designs enable LEAP-Track to converge significantly faster and achieve markedly higher training throughput than the state-of-the-art baseline, while also achieving superior tracking accuracy over existing self-supervised TAP methods. Collectively, these results confirm that adaptive sparsification provides an effective pathway for learning robust visual correspondence from unlabeled videos.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.62002020, 61802022, and 61802027), and the Open Foundation of the State Key Laboratory of Precision Space-time Information Sensing Technology under Grant STL2023-B-04-01(K).

References

Bian, Z.; Jabri, A.; Efros, A. A.; and Owens, A. 2022. Learning Pixel Trajectories with Multiscale Contrastive Random Walks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6508–6519. IEEE.

Doersch, C.; Gupta, A.; Markeeva, L.; Recasens, A.; Smaira, L.; Aytar, Y.; Carreira, J.; Zisserman, A.; and Yang, Y. 2022. TAP-Vid: A Benchmark for Tracking Any Point in a Video. *Advances in Neural Information Processing Systems*, 35: 13610–13626.

Doersch, C.; Yang, Y.; Vecerik, M.; Gokay, D.; Gupta, A.; Aytar, Y.; Carreira, J.; and Zisserman, A. 2023. Tapir: Tracking Any Point with Per-Frame Initialization and Temporal Refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10061–10072.

Dwibedi, D.; Aytar, Y.; Tompson, J.; Sermanet, P.; and Zisserman, A. 2019. Temporal Cycle-Consistency Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1801–1810. IEEE.

Feng*, H.; Zhang*, J.; Wang, Q.; Ye, Y.; Yu, P.; Black, M. J.; Darrell, T.; and Kanazawa, A. 2025. St4RTrack: Simultaneous 4D Reconstruction and Tracking in the World. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Geng, D.; Herrmann, C.; Hur, J.; Cole, F.; Zhang, S.; Pfaff, T.; Lopez-Guevara, T.; Aytar, Y.; Rubinstein, M.; Sun, C.; et al. 2025. Motion Prompting: Controlling Video Generation with Motion Trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1–12. IEEE.

Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y.; Duckworth, D.; Fleet, D. J.; Gnanapragasam, D.; Golemo, F.; Herrmann, C.; and Kipf, T. 2022. Kubric: A Scalable Dataset Generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3749–3761. IEEE.

Harley, A. W.; Fang, Z.; and Fragkiadaki, K. 2022. Particle Video Revisited: Tracking through Occlusions Using Point Trajectories. In *Proceedings of the European Conference on Computer Vision*, 59–75. Springer.

Huang, Z.; Shi, X.; Zhang, C.; Wang, Q.; Cheung, K. C.; Qin, H.; Dai, J.; and Li, H. 2022. FlowFormer: A Transformer Architecture for Optical Flow. In *Proceedings of the European Conference on Computer Vision*, 668–685. Springer.

Jabri, A.; Owens, A.; and Efros, A. A. 2020. Space-Time Correspondence as a Contrastive Random Walk. *Advances in Neural Information Processing Systems*, 33: 19545–19560.

Jain, G.; Hegde, N.; Kusupati, A.; Nagrani, A.; Buch, S.; Jain, P.; Arnab, A.; and Paul, S. 2024. Mixture of Nested Experts: Adaptive Processing of Visual Tokens. *Advances in Neural Information Processing Systems*, 37: 58480–58497.

Jiang, S.; Lu, Y.; Li, H.; and Hartley, R. 2021a. Learning Optical Flow from a Few Matches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16592–16600. IEEE.

Jiang, W.; Trulls, E.; Hosang, J.; Tagliasacchi, A.; and Yi, K. M. 2021b. COTR: Correspondence Transformer for Matching Across Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6207–6217. IEEE.

Karaev, N.; Makarov, Y.; Wang, J.; Neverova, N.; Vedaldi, A.; and Rupprecht, C. 2025. Cotracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6013–6022.

Karaev, N.; Rocco, I.; Graham, B.; Neverova, N.; Vedaldi, A.; and Rupprecht, C. 2024. CoTracker: It is Better to Track Together. In *Proceedings of the European Conference on Computer Vision*, 18–35. Springer.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The Kinetics Human Action Video Dataset. arXiv:1705.06950.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Li, H.; Zhang, H.; Liu, S.; Zeng, Z.; Li, F.; Li, B.; Ren, T.; and Zhang, L. 2024. TAPTRv2: Attention-Based Position Update Improves Tracking Any Point. *Advances in Neural Information Processing Systems*, 37: 101074–101095.

Li, J.; Zhang, S.; and Huang, T. 2019. Multi-Scale 3D Convolution Network for Video-Based Person Re-Identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8618–8625. AAAI Press.

Li, R.; and Liu, D. 2023. Spatial-Then-Temporal Self-Supervised Learning for Video Correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2279–2288. IEEE.

Li, X.; Liu, S.; De Mello, S.; Wang, X.; Kautz, J.; and Yang, M.-H. 2019. Joint-Task Self-Supervised Learning for Temporal Correspondence. *Advances in Neural Information Processing Systems*, 32.

Liu, L.; Zhang, J.; He, R.; Liu, Y.; Wang, Y.; Tai, Y.; Luo, D.; Wang, C.; Li, J.; and Huang, F. 2020. Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6489–6498. IEEE.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022. IEEE.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Neoral, M.; Šerých, J.; and Matas, J. 2024. MFT: Long-Term Tracking of Every Pixel. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 6837–6847. IEEE.

Qian, Z.; Han, R.; Hou, J.; Song, L.; and Feng, W. 2025. VOVTrack: Exploring the Potentiality in Raw Videos for Open-Vocabulary Multi-Object Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7472–7482.

Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. SuperGlue: Learning Feature Matching with Graph Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4938–4947. IEEE.

Shrivastava, A.; and Owens, A. 2024. Self-Supervised Any-Point Tracking by Contrastive Random Walks. In *Proceedings of the European Conference on Computer Vision*, 267–284. Springer.

Song, Y.; Lei, J.; Wang, Z.; Liu, L.; and Daniilidis, K. 2024. Track Everything Everywhere Fast and Robustly. In *Proceedings of the European Conference on Computer Vision*, 343–359. Springer.

Tang, L.; Jia, M.; Wang, Q.; Phoo, C. P.; and Hariharan, B. 2023. Emergent Correspondence from Image Diffusion. In *Advances in Neural Information Processing Systems*, volume 36, 1363–1389.

Teed, Z.; and Deng, J. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Proceedings of the European Conference on Computer Vision*, 402–419. Springer.

Vecerik, M.; Doersch, C.; Yang, Y.; Davchev, T.; Aytar, Y.; Zhou, G.; Hadsell, R.; Agapito, L.; and Scholz, J. 2024. RoboTAP: Tracking Arbitrary Points for Few-Shot Visual Imitation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 5397–5403. IEEE.

Wang, P.; Wang, X.; Wang, F.; Lin, M.; Chang, S.; Li, H.; and Jin, R. 2022. KVT: K-NN Attention for Boosting Vision Transformers. In *Proceedings of the European Conference on Computer Vision*, 285–302. Springer.

Wang, Q.; Chang, Y.-Y.; Cai, R.; Li, Z.; Hariharan, B.; Holynski, A.; and Snavely, N. 2023. Tracking Everything Everywhere All at Once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19795–19806. IEEE.

Wang, X.; Jabri, A.; and Efros, A. A. 2019. Learning Correspondence from the Cycle-Consistency of Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2566–2576. IEEE.

Xiao, Y.; Wang, Q.; Zhang, S.; Xue, N.; Peng, S.; Shen, Y.; and Zhou, X. 2024. SpatialTracker: Tracking Any 2D Pixels in 3D Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20406–20417. IEEE.

Xu, H.; Zhang, J.; Cai, J.; Rezatofighi, H.; and Tao, D. 2022. GMFlow: Learning Optical Flow via Global Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8121–8130. IEEE.

Xu, H.; Zhang, J.; Cai, J.; Rezatofighi, H.; Yu, F.; Tao, D.; and Geiger, A. 2023. Unifying Flow, Stereo and Depth Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11): 13941–13958.

Zhao, Z.; Jin, Y.; and Heng, P.-A. 2021. Modelling Neighbor Relation in Joint Space-Time Graph for Video Correspondence Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9960–9969. IEEE.