A memory-based deep learning method for long-term visual point tracking

Lazar Đoković

Supervisor: prof. dr. Matej Kristan
Co-supervisor: doc. dr. Alan Lukežič

June 2026

University of Ljubljana
Faculty of Computer and Information Science

FRI

A memory-based deep learning method for long-term visual point tracking

Lazar Đoković

Erasmus Mundus Joint Master in Artificial Intelligence (EMAI)

Faculty of Computer and Information Science

Supervisor: prof. dr. Matej Kristan
Cosupervisor: doc. dr. Alan Lukežič

Dense point tracking estimates long-range trajectories for all reference pixels, making it a powerful representation of video motion. This is particularly challenging in a causal online setting, where future frames are unavailable at inference time, while points may exhibit substantial motion, and undergo occlusion and appearance changes. Maintaining reliable point identities over long videos thus remains challenging, especially when appearances must be updated over time without corrupting dense and temporally stable predictions. We introduce DeMeTer, a dense online point tracker that combines memory-conditioned global matching with localized motion refinement. Instead of matching only static reference features, DeMeTer maintains a confidence-gated appearance memory that updates each point with reliable past observations, enabling re-detection after occlusion or appearance changes. Simultaneously, a separate motion memory stores recent local dynamics, allowing the refininig module to generate accurate and temporally consistent trajectories. This reference-aligned dual-memory architecture separates identity preservation from short-term motion modeling, making explicit re-detection feasible at dense scale while remaining fully causal. DeMeTer establishes a new state-of-the-art by outperforming related methods by 4.1% in δ_{avg}, while comfortably running in realtime.

Computer vision | Deep learning | Visual point tracking | Correspondence | Optical flow

Correspondence matching is a fundamental problem in computer vision, whose central tasks is to establish point-level correspondences between two or more of images [1]. Its video extension, visual point tracking, has progressed significantly with architectures that improve robustness, temporal consistency, and efficiency [2–4]. However, long-term tracking remains difficult under large motion, extended occlusion, and appearance changes due to deformation, viewpoint shifts, or illumination variation.

A common approach in strong point trackers is to decompose tracking into appearance-based matching followed by iterative refinement [2–6]. This decomposition is effective because the two stages address complementary requirements: global matching handles large displacements and re-detection, while local refinement improves localization and temporal consistency. However, maintaining this design becomes challenging when tracking densely sampled points. Sparse trackers can afford global correlation for a limited set of query points, whereas dense trackers must estimate correspondences for every reference pixel at every frame of a video sequence. As a result, dense methods often rely on local, optical-flow-style refinement and windowed processing [7, 8]. While efficient, these choices can weaken long-term re-detection after extended occlusions and introduce a small-motion bias.

We argue that robust appearance matching should remain central to dense tracking. Dense correspondences provide a complete motion representation, preserving object boundaries, local deformations, and motion cues that sparse tracks may miss. However, density alone does not ensure robustness: when points are occluded or visually transformed, the tracker must still recover their identities from appearance rather than merely propagate their previous locations. Recent work, such as ReTracker [6], shows that strengthening global matching improves long-term tracking under occlusion and appearance variation. The key challenge, therefore, is to preserve the re-detection ability of appearance matching while making it efficient for dense online tracking.

This challenge is amplified in the online setting, where trackers process frames sequentially without access to future frames. Points may disappear under occlusion, change appearance over time, or become ambiguous in textureless regions; consequently, many state-of-the-art methods rely on window-based or offline processing to use future context for recovery [5, 7, 8]. However, such designs introduce latency and are less suitable for reactive systems. Online, or causal, tracking therefore requires memory to compensate for the missing future context. In this setting, appearance and motion memory serve complementary roles: appearance memory propagates long-term point identities and enables re-detection, while motion memory regularizes motion predictions under appearance uncertainty using recent local motion cues.

We address the aforementioned issues with DeMeTer (Dense Memory-based Point Tracker), an online point tracker that decomposes tracking into global appearance match-

Metoda globokega učenja s spominom za dolgoročno vizualno sledenje točk

Gosto sledenje slikovnim točkam je proces, ki ocenjuje dolgoročne trajektorije za vse referenčne točke, in predstavlja zmogljivo reprezentacijo gibanja v videoposnetkih. To je posebej zahtevno v kavzalnem sprotnem sledenju, kjer prihodnje slike ob času inference v trenutni sliki niso na voljo, premiki točk so lahko drastični, točke se zakrijejo ali spremenijo svoj videz. Zanesljivo ohranjanje identitet točk skozi dolge videoposnetke zato ostaja zahtevno, zlasti kadar je potrebno njihovo podobo skozi čas posodabljati, ne da bi s tem poslabšali stabilnost napovedi. Predstavljamo DeMeTer, gosto sprotno metodo za sledenje točkam, ki združuje globalno ujemanje, pogojeno s pomnilnikom, z lokalno izboljšavo gibanja. Namesto ujemanja zgolj s statičnimi referenčnimi značilkami DeMeTer vzdržuje pomnilnik videza, ki se posodablja na podlagi zaupanja. Ta pomnilnik vsako točko posodablja z zanesljivimi preteklimi meritvami, kar omogoča ponovno zaznavo po zakritju ali spremembi videza. Hkrati ločen pomnilnik gibanja shranjuje kratkoročno lokalno dinamiko, kar modulu za izboljšavo lokacij omogoča ohranjanje natančnih in časovno skladnih trajektorij. Ta z referenčnim okvirjem poravnana arhitektura z dvojnim pomnilnikom ločuje ohranjanje identitete točk od kratkoročnega modeliranja njihovega gibanja. S tem omogoča eksplicitno gosto ponovno zaznavo ob popolni kavzalnosti. DeMeTer postavlja novo stanje tehnike in sorodne metode preseže za 4,1% po metriki δ_{avg}, hkrati pa deluje v realnem času.

Računalniški vid | Globoko učenje | Vizualno sledenje točk | Ujemanje značilk | Optični tok

ing and localized motion refinement. For each target frame, an appearance memory adapts reference features using reliable past appearances of each point. A global matcher then correlates these memory-conditioned reference features with the full target frame, producing dense correspondences and occlusion-aware confidence estimates. The predictions are then refined using local correlation neighborhoods and a separate motion memory that stores recent motion and local correlation information. By separating memory according to function, DeMeTer preserves stable appearance cues for long-term point identity while using recent motion information to resolve local tracking ambiguities.

For efficiency, both memories are maintained in the reference-frame coordinate system, keeping each slot aligned with the same query pixel throughout the video. The appearance memory is updated only for reliable visible points, preserving identity through occlusion and enabling re-detection after appearance changes. The motion memory is updated as a recent rolling history, providing short-term temporal context for smoother refinement. This decouples robustness and accuracy: the appearance memory supports long-range matching, while the motion memory supports local accuracy and temporal consistency.

Our contributions are twofold. First, we propose a reference-aligned dual-memory design that separates (i) a confidence-gated appearance memory for preserving reliable point identities through occlusion and appearance changes, and (ii) a recent motion memory for temporally consistent local refinement. Second, we introduce DeMeTer, a fully online dense point tracker that preserves explicit global appearance matching for every reference pixel, enabling robust long-range re-detection without requiring future frames or a sliding window. On challenging visual point-tracking benchmarks, DeMeTer outperforms the previous state-of-the-art by 4.1% in δ_{avg}, while remaining efficient for dense tracking.

1. Related Work

1.1. Optical Flow and Dense Tracking. Optical flow estimates dense correspondences between frames, and has a long history, spanning classical variational [9, 10] and local optimization methods [11] to recent learning-based architectures [12–16]. Modern flow models such as RAFT [14] and SEA-RAFT [16] iteratively refine dense motion using all-pairs correlation and local recurrent updates. These methods provide accurate short-range motion but are primarily designed for two-frame correspondence. Long-range dense tracking is often achieved by chaining frame-to-frame flows [17, 18] or by predicting reference-to-target flow fields over a temporal window [7, 19, 20]. While effective, these approaches can accumulate drift, rely on fixed temporal context, or fail to re-detect points after long occlusions. CoWTracker [8] further departs from correlation-heavy flow estimation by propagating motion through warping. In contrast, DeMeTer maintains dense global appearance matching as an explicit first stage and uses local refinement only after large-displacement correspondences are initialized.

1.2. Tracking Any Point. The track-any-point task was popularized by TAP-Vid [21] and early successful point trackers such as PIPs [22], TAP-Net [21], and TAPIR [2]. TAPIR combines global matching with local temporal refinement, while later methods improve robustness through local all-pair correlation [3], longer synthetic training data [23, 24], joint multi-point tracking [4, 5], query-based tracking [25, 26], next-token prediction [24, 27], image-matching pretraining [6], and self-training on real videos [28]. Other works extend point tracking with 3D reconstruction [29, 30], or multi-view constraints [31, 32]. These methods demonstrate that appearance-based matching, temporal refinement, and interaction between tracks are important for long-term robustness. However, most are designed for sparse query sets, where expensive global correlation or self-attention is computationally feasible. Because dense tracking requires correspondence and visibility estimates for every pixel in each frame, scaling this strategy to all reference pixels is challenging. By contrast, DeMeTer makes point tracking practical in the dense online setting by using compact appearance and motion memories to estimate correspondences and visibility for all points.

1.3. Online and Memory-based Point Tracking. Online tracking removes access to future frames and therefore requires information from past observations to be stored compactly. Online, or causal, variants of sparse trackers improve sequential tracking, but are still limited by the cost of tracking many query points densely [5, 33]. Dense methods such as DOT [34], which densifies sparse point tracks through interpolation and optical-flow refinement, and AllTracker [7], which predicts dense trajectories primarily through refinement-based updates, improve scalability but does not perform explicit dense per-pixel global appearance matching, making long-term identity recovery more challenging under occlusion and appearance changes. Temporally-aware features [35] and streaming memory have recently been used to improve online tracking, with SPOT [36] being closest to our setting because it targets online dense point tracking through sensory

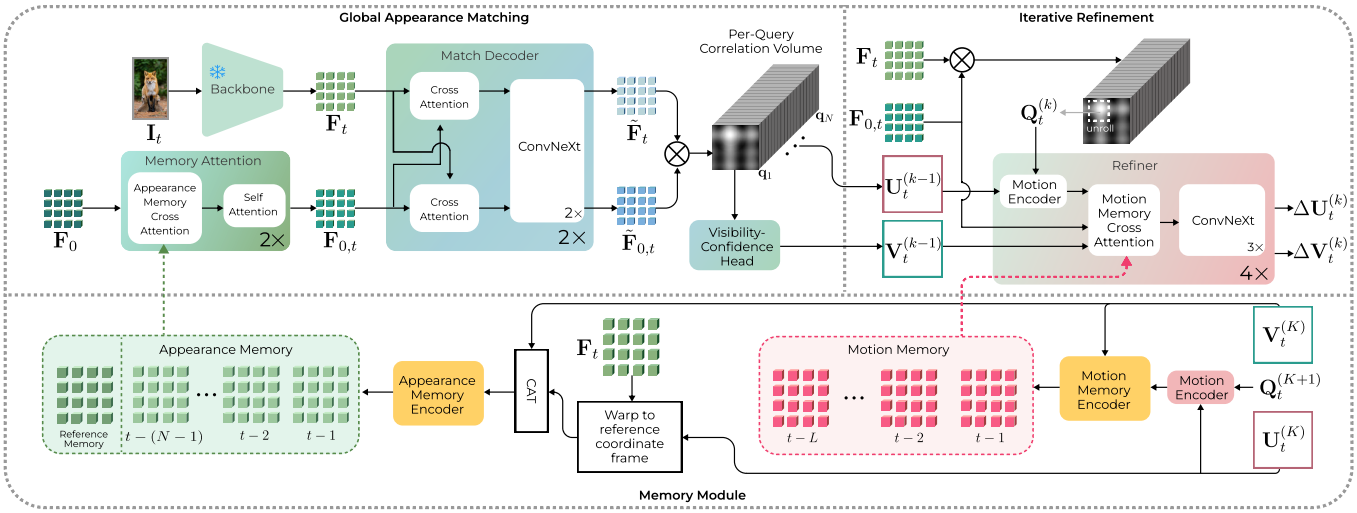

Figure 1. Overview of DeMeTer. The model decomposes tracking into global appearance matching, where appearance memory adapts reference features to build per-query correlation maps, and localized refinement, where local correlations and motion memory produce temporally consistent updates. The memory module encodes reliable appearance and motion features in the reference coordinate frame for future use.

memory and forward visibility-guided splatting. However, SPOT uses memory to enhance optical-flow-style prediction, where long-range correspondences are obtained through flow regression and propagation. In contrast, it does not explicitly maintain a per-query appearance memory that performs global matching between each reference point and the full target frame. DeMeTer differs by maintaining reference-aligned appearance memory for each query pixel, using it to condition dense global correlation maps for re-detection, and separating this from a motion memory used only for local temporal refinement.

2. DeMeTer

DeMeTer is initialized on the first frame and delivers corresponding pixel locations in all subsequent frames, along with their visibility indicator. For each target frame, the model outputs dense correspondences from the reference (query) frame to the target frame, along with visibility and confidence estimates, as shown in Figure 1.

Formally, let $\mathcal{V} = \{\mathbf{I}_t\}_{t=0}^{T}$ denote a video stream, where $T \in \mathbb{N}$ and $\mathbf{I}_t \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}} \times 3}$. The first frame, $\mathbf{I}_0$, serves as the query frame, and each frame $\mathbf{I}_t$ with $t > 0$ is treated as a target frame. For each target frame, the tracker predicts a flow field $\mathbf{U}_t \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}} \times 2}$, which maps query-frame pixels to their corresponding target-frame locations, along with a visibility-confidence map $\mathbf{V}_t \in [0,1]^{H_{\text{in}} \times W_{\text{in}} \times 2}$.

For efficiency, each frame is first resized to a fixed processing resolution of $H_p \times W_p = 512 \times 512$ and passed through a lightweight encoder $\mathcal{E}(\cdot)$, which extracts low-resolution features

$$\mathbf{F}_t = \mathcal{E}(\mathbf{I}_t), \qquad \mathbf{F}_t \in \mathbb{R}^{H \times W \times D}, \tag{1}$$

where $H = H_p/8$, $W = W_p/8$, and $D = 128$.

At a high level, DeMeTer processes the video stream sequentially, one frame at a time. The query frame features are first adapted using an appearance memory and matched with the current target frame features to obtain initial correspondence, visibility, and confidence estimates (Section 2.1). An iterative refiner then updates these predictions using local correlation features and a motion memory to improve temporal consistency (Section 2.2). Finally, the refined flow is used to update the reference-aligned memories: target features are warped back to the reference coordinate frame to update the appearance memory, while the corresponding local correlations and flows are encoded into the motion memory (Section 2.3).

2.1. Global Appearance Matching. Let $\mathbf{F}_0 \in \mathbb{R}^{H \times W \times D}$ denote the query frame feature map. The global matching stage handles large displacements and reappearances by correlating each query frame feature with the entire target frame feature map $\mathbf{F}_t$. To address appearance changes, the query frame features $\mathbf{F}_0$ are first conditioned on an appearance memory $\mathbf{M}_t^A \in \mathbb{R}^{N \times H \times W \times D}$, which stores the encoded reference frame appearance along with the $N-1$ most recent reliable appearances of each query point (see Section 2.3). This adaptation is performed using per-pixel temporal cross-attention, with cost quadratic to the memory length [7], followed by self-attention over the query frame, to preserve local context among neighboring points, producing $\mathbf{F}_{0,t} \in \mathbb{R}^{H \times W \times D}$.

$\mathbf{F}_{0,t}$ and $\mathbf{F}_t$ are then passed to the match decoder, which further conditions the reference and target features to improve matching. To provide a valid match for occluded

or absent query points, a learned dustbin token $\mathbf{b} \in \mathbb{R}^D$ is appended to the target features $\mathbf{F}_t$. A bidirectional match decoder then applies mutual cross-attention between $\mathbf{F}_{0,t}$ and $[\mathbf{F}_t; \mathbf{b}]$, where $[\cdot;\cdot]$ denotes concatenation along the token dimension. This is followed by a shared ConvNeXt [37] block, producing $\tilde{\mathbf{F}}_{0,t} \in \mathbb{R}^{H\times W\times D}$ and $\tilde{\mathbf{F}}_t \in \mathbb{R}^{(HW+1)\times D}$.

Global matches are obtained by constructing an all-pairs correlation volume between $\tilde{\mathbf{F}}_{0,t}$ and $\tilde{\mathbf{F}}_t$, resulting in $\mathbf{C}_t \in \mathbb{R}^{H\times W\times(HW+1)}$. Let $\mathcal{G} = \{1,\dots,H\} \times \{1,\dots,W\}$ denote the 2D spatial grid. For a query pixel at $\mathbf{p} \in \mathcal{G}$, the first HW channels form a spatial response map over the target frame, while the final channel acts as a dustbin score for occluded or unmatched points. The initial target location is computed as

$$\mathbf{X}_t^{(0)}(\mathbf{p}) = \sum_{\mathbf{g}\in\mathcal{G}} w_{\mathbf{p},\mathbf{g}}\, \mathbf{g}, \tag{2}$$

where $w_{\mathbf{p},\mathbf{g}}$ is the normalized matching score for target location $\mathbf{g}$, obtained by applying a softmax over the spatial response map

$$w_{\mathbf{p},\mathbf{g}} = \frac{\exp\left(\tau \mathbf{C}_t(\mathbf{p},\mathbf{g})\right)}{\sum_{\mathbf{g}'\in\mathcal{G}} \exp\left(\tau \mathbf{C}_t(\mathbf{p},\mathbf{g}')\right)}. \tag{3}$$

with temperature $\tau = 20$. Evaluating this for all query positions gives the initial coordinate map $\mathbf{X}_t^{(0)} \in \mathbb{R}^{H\times W\times 2}$, from which we obtain the initial flow $\mathbf{U}_t^{(0)}$.

The correlation volume $\mathbf{C}_t$ captures match reliability, with peaked responses indicating confidence and diffuse responses indicating ambiguity or occlusion. We use it to initialize the visibility-confidence scores by compressing $\mathbf{C}_t$ to D channels with a ConvNeXt encoder $\psi(\cdot)$. To retain an explicit strongest-match cue, we extract a maximum-correlation map $\mathbf{S}_t \in \mathbb{R}^{H\times W\times 1}$ by taking the maximum of $\mathbf{C}_t$ over its $HW+1$ channel dimension, including the dustbin channel. The two representations are concatenated along the channel dimension and projected to two output channels

$$\mathbf{V}_t^{(0)} = \rho\left([\psi(\mathbf{C}_t); \mathbf{S}_t]\right), \tag{4}$$

where $\rho(\cdot)$ denotes 1×1 convolution.

2.2. Iterative Refinement. After global appearance matching, the iterative refiner $\mathcal{R}$ improves the initial flow $\mathbf{U}_t^{(0)}$ and visibility-confidence $\mathbf{V}_t^{(0)}$ by reducing the position errors and enforcing temporal consistency. The design follows AllTracker [7], but replaces interleaved temporal and spatial processing with a single temporal cross-attention to motion memory followed by three ConvNeXt blocks, improving efficiency without reducing accuracy.

Since large displacements are handled by the global matcher, refinement only considers local neighborhoods around the current estimates. For each target frame, the refiner computes a fine-grained correlation volume $\mathbf{G}_t \in \mathbb{R}^{H\times W\times H\times W}$ between the memory-adapted reference features $\mathbf{F}_{0,t}$ and target features $\mathbf{F}_t$. This volume is computed once per frame and reused across refinement iterations. At iteration k, local correlations $\mathbf{Q}_t^{(k)} \in \mathbb{R}^{H\times W\times(2R+1)^2}$ are sampled from $\mathbf{G}_t$ around the current coordinates $\mathbf{X}_t^{(k-1)}$ within radius $R = 4$.

The local correlations $\mathbf{Q}_t^{(k)}$, current flow $\mathbf{U}_t^{(k-1)}$, visibility-confidence $\mathbf{V}_t^{(k-1)}$, and memory-adapted reference features $\mathbf{F}_{0,t}$ are encoded into a per-pixel motion representation, which attends to the motion memory $\mathbf{M}_t^M$ through temporal cross-attention (see Section 2.3 for more details on memory construction). The resulting features are processed by ConvNeXt blocks to predict residual updates

$$\Delta\mathbf{U}_t^{(k)}, \Delta\mathbf{V}_t^{(k)} = \mathcal{R}\left(\mathbf{U}_t^{(k-1)}, \mathbf{V}_t^{(k-1)}, \mathbf{Q}_t^{(k)}, \mathbf{F}_{0,t}\right). \tag{5}$$

The residuals are applied additively and refinement is repeated for $K = 4$ iterations with shared weights. Final predictions are upsampled using learned convex upsampling [38] and resized to the input resolution with bilinear interpolation.

2.3. Memory Module. To enable efficient per-pixel temporal attention, DeMeTer stores both appearance and motion memories in the reference coordinate frame, keeping each slot tied to the same query pixel throughout the video. After refinement, these memories are updated with information extracted from the current frame.

Appearance memory. The appearance memory $\mathbf{M}_t^A \in \mathbb{R}^{N\times H\times W\times D}$ stores a static reference frame entry and $N-1$ recent reliable appearances for each query point. To form a new memory entry, the refined coordinates $\mathbf{X}_t^{(K)}$ are used to warp target frame features $\mathbf{F}_t$ back into the query-frame coordinate system

$$\mathbf{A}_t = \mathcal{T}(\mathbf{F}_t, \mathbf{X}_t^{(K)}), \tag{6}$$

where $\mathcal{T}(\cdot,\cdot)$ denotes bilinear interpolation.

The warped features $\mathbf{A}_t$ are concatenated with the final visibility-confidence estimates $\mathbf{V}_t^{(K)}$ and encoded with a 1×1 convolution and two ConvNeXt blocks, producing $\tilde{\mathbf{A}}_t \in \mathbb{R}^{H \times W \times D}$. The appearance memory is updated with a confidence-gated FIFO rule. The current aligned feature $\tilde{\mathbf{A}}_t$ is inserted only at pixels whose visibility-confidence product exceeds a threshold $\theta = 0.9$. At these reliable pixels, older memory entries are shifted back by one slot; at unreliable pixels, all memory entries remain unchanged. This prevents ambiguous or occluded regions from corrupting the stored appearance features.

Motion memory. The motion memory $\mathbf{M}_t^M \in \mathbb{R}^{L \times H \times W \times D}$ provides per-point motion context for temporal refinement. After refinement, local correlation neighborhoods $\mathbf{Q}_t^{(K+1)}$ are recomputed at the final coordinates $\mathbf{X}_t^{(K)}$, thus aligning motion features to the reference coordinate frame. The local correlations $\mathbf{Q}_t^{(K+1)}$, final flow $\mathbf{U}_t^{(K)}$, and visibility-confidence estimates $\mathbf{V}_t^{(K)}$ are encoded using the refiner's motion encoder and a lightweight memory encoder to form a new motion entry $\tilde{\mathbf{B}}_t \in \mathbb{R}^{H \times W \times D}$. The motion memory is updated in a FIFO fashion with length $L = 16$, without reliability masking, keeping only recent motion and correlation information as a short-term prior when appearance-based matching is ambiguous.

2.4. Training. DeMeTer is trained end-to-end with full backpropagation through time on 24-frame sparse point-tracking sequences generated using Kubric [39]. The global matching stage is supervised with a weighted BCE response-map loss $\mathcal{L}_{\text{hm}}$ on annotated query points: visible points use Gaussian targets with $\sigma = 1.5$ at the ground-truth location, while invisible points use the dustbin target.

The refiner is supervised in pixel coordinates with an L_1 trajectory loss $\mathcal{L}_{\text{track}}$ on visible points at each refinement step, assigning larger weights to later iterations following [7]. Visibility and confidence are supervised with BCE losses $\mathcal{L}_{\text{vis}}$ and $\mathcal{L}_{\text{conf}}$, where confidence is positive if the prediction is within 12 pixels of the ground truth. The final loss is

$$\mathcal{L} = 0.1 \cdot \mathcal{L}_{\text{hm}} + 0.05 \cdot \mathcal{L}_{\text{track}} + \mathcal{L}_{\text{vis}} + \mathcal{L}_{\text{conf}}. \tag{7}$$

3. Experiments

This section presents the quantitative and qualitative evaluation of DeMeTer. We evaluate dense online point tracking on a broad set of public benchmarks and further assess zero-shot transfer to optical flow. We also analyze refinement behavior and ablate the main architectural components of the model.

3.1. Implementation Details. DeMeTer has 19.9M parameters and uses a modified ConvNeXt-Tiny [37] encoder following [7]. For all attention operations, we add fixed sinusoidal positional encodings to both queries and keys: 2D spatial encodings are used for self-attention, while 1D temporal encodings are used for per-pixel cross-attention over memory entries. We train the model for 200k steps on eight H100 GPUs using AdamW [40] with a learning rate 2.5×10^{-4}. Training uses standard spatial and photometric augmentations [22], including random shifting, scaling, color jitter, and square occlusions. We run inference on A100 GPUs and use an A100_80GB only when evaluating CoWTracker due to its high memory requirements.

3.2. Point Tracking. The following seven public benchmarks are employed: CroHD [41], DriveTrack [42], TAPVid-DAVIS [21], TAPVid-Kinetics [21], RGB-Stacking [43], RoboTAP [33], and EgoPoints [44]. These benchmarks cover diverse synthetic and real-world video domains, including YouTube videos, surveillance footage, driving videos, egocentric videos, and robotic manipulation.

Following the standard point tracking protocol [21], we report δ_{avg} for all datasets, which measures how close the extracted point trajectories are to the ground-truth trajectories. For each threshold $k \in \{1, 2, 4, 8, 16\}$, δ_k gives the percentage of predicted points that fall within k pixels of the ground truth at 256×256 resolution. The final score, δ_{avg}, is calculated by averaging these values over all thresholds. On TAPVid-DAVIS, TAPVid-Kinetics, RGB-Stacking, and RoboTAP, we also report Occlusion Accuracy (OA) and Average Jaccard (AJ). OA measures how accurately the model predicts whether a point is visible or occluded, while AJ combines localization and visibility into a single score. Following prior work [7], we do not report AJ and OA on datasets with unreliable visibility annotations.

DeMeTer is compared with a broad set of online trackers, including TAP-Net [21], MFT [17], DOT† [34], Online TAPIR [33], CoTracker3 [5], AllTracker [7], SPOT [36], and ReTracker [6]. Unless stated otherwise, AllTracker is trained on Kubric to ensure a fair comparison. CoTracker3 and AllTracker were adapted for causal inference by reducing their sliding windows to two frames and advancing one frame at a time. We denote these

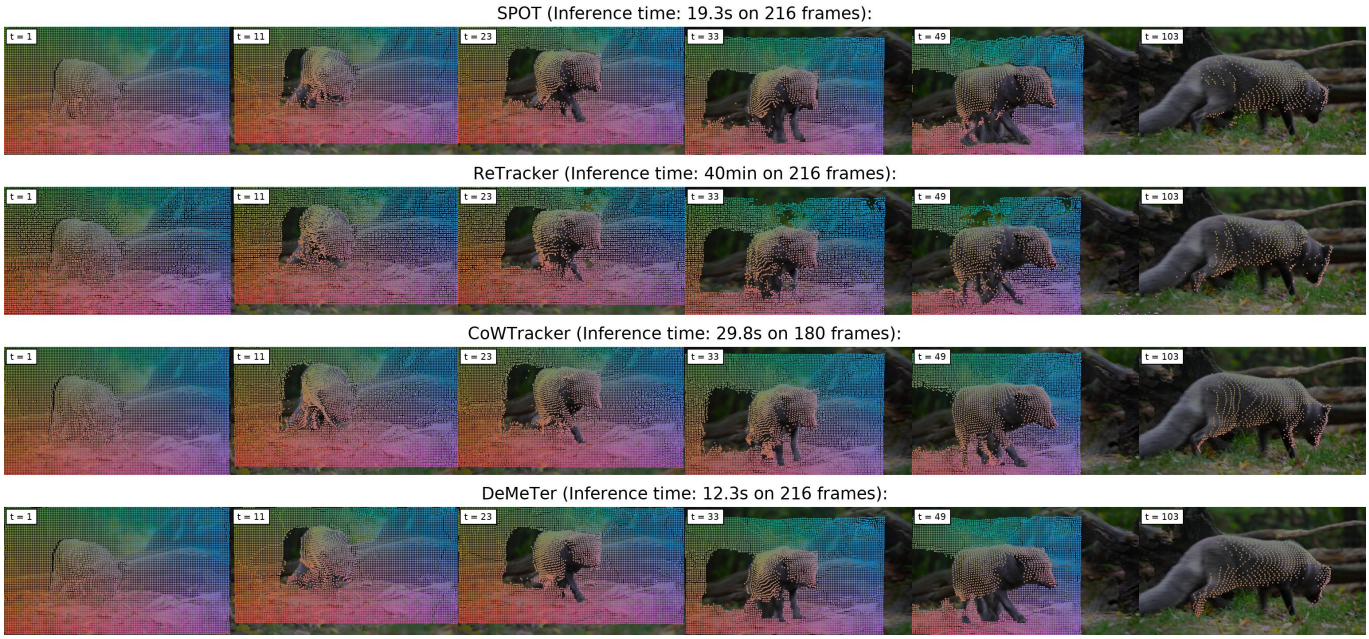

Figure 2. Qualitative comparison of state-of-the-art online trackers with a visibility threshold of 0.5. Runtime is measured as the time required to output dense tracks for a 216-frame fox sequence. DeMeTer produces coherent dense trajectories and is faster than state-of-the-art online trackers.

variants as CoTracker3$_{S=1}$ and AllTracker$_{S=1}$. This enables a direct comparison with recent sparse, dense, and memory-based trackers under an online evaluation protocol.

Table 1 presents the main point-tracking results, separating offline and online methods. Among online trackers, DeMeTer achieves the highest average performance across all three metrics. Compared with previous state-of-the-art online methods, DeMeTer outperforms SPOT by 11.4% in average AJ and 7.8% in δ_{avg}, and surpasses ReTracker by 8.6% in average AJ and 5.8% in δ_{avg}. Compared with the online AllTracker$_{S=1}$ variant, DeMeTer improves average AJ by 6.9%, δ_{avg} by 5.0%, and OA by 4.2%.

The gains are consistent across both synthetic and real-world datasets: DeMeTer obtains the best online results on RGB-Stacking, Kinetics, and RoboTAP for all three metrics. ReTracker performs best on DAVIS, but its performance does not transfer as consistently across the full evaluation suite. DeMeTer achieves its largest improvements over prior state-of-the-art online trackers on Kinetics and RoboTAP, which feature longer sequences with challenging camera motion, articulation, and robotic interactions. These results indicate that DeMeTer is better at maintaining point identity over extended temporal horizons, where tracks must remain stable and points often need to be re-identified after occlusion or appearance changes.

Overall, the results demonstrate that decomposing tracking into two steps and conditioning both on appropriate memories provides a strong mechanism for robust causal tracking. Although CoWTracker achieves the highest offline scores, DeMeTer remains competitive while operating fully online. In particular, DeMeTer performs on par with or outperforms nearly all offline trackers except CoWTracker, while avoiding offline sequence-level processing. CoWTracker processes the entire video jointly using a VGGT backbone, which improves accuracy but is computationally expensive and inherently ill-suited for real-time causal tracking.

To evaluate generalization across a broader set of benchmarks spanning diverse scenes, Table 2 compares DeMeTer with AllTracker, a window-based dense tracker, on seven datasets using δ_{avg}. Despite operating causally, DeMeTer improves over AllTracker by 1.3% in average δ_{avg} while achieving a 2.2× inference speedup. DeMeTer outperforms AllTracker on 5 of 7 datasets. In particular on general, driving, egocentric and robotic scenes (TAPVid-DAVIS, TAPVid-Kinetics, DriveTrack, EgoPoints, RoboTAP), while delivering on-par performance on surveillance and synthetic scenes (CroHD, RGB-Stacking). These results indicate that DeMeTer generalizes better across diverse benchmarks than a window-based dense tracker, while requiring no access to future frames and being more than twice as fast.

DeMeTer is further qualitatively compared in Figure 2 with two state-of-the-art online trackers, SPOT [36] and ReTracker [6], as well as the current best offline tracker, CoWTracker [8]. In addition, processing times are also reported for inference on the entire fox sequence, which contains 216 frames. Note, that the VGGT backbone in CoWTracker limits the number of frames, as it requires encoding them jointly. Therefore, the processing time reported for CoWTracker is based on the maximal number of frames

Table 1. Point tracking results on TAPVid-DAVIS [21], RGB-Stacking [43], TAPVid-Kinetics [21], and RoboTAP [33], including both offline and online trackers. DeMeTer achieves state-of-the-art performance among online methods across a wide range of datasets, with particularly strong improvements on Kinetics and RoboTAP.

	DAVIS			RGB-Stacking			Kinetics			RoboTAP			**Average**		
Method	AJ ↑	δ_{avg} ↑	OA ↑	AJ ↑	δ_{avg} ↑	OA ↑	AJ ↑	δ_{avg} ↑	OA ↑	AJ ↑	δ_{avg} ↑	OA ↑	AJ ↑	δ_{avg} ↑	OA ↑
Offline															
LocoTrack [3]	62.9	75.3	87.2	69.7	83.2	89.5	52.9	66.8	85.3	52.9	66.8	85.3	59.6	73.0	86.8
BootsTAPIR [28]	61.4	73.6	88.7	70.8	83.0	89.9	54.6	68.4	86.5	54.6	68.4	86.5	60.4	73.4	87.9
CoTracker3 [5]	64.4	76.9	91.2	74.3	85.2	92.4	54.7	67.8	87.4	54.7	67.8	87.4	62.0	74.4	89.6
DOT [34]	60.1	74.5	89.0	77.1	87.7	93.3	48.4	63.8	85.2	–	–	–	61.9	75.3	89.2
DELTA [20]	60.8	75.2	87.6	72.2	83.0	91.4	51.0	66.6	84.0	60.2	73.4	85.6	61.1	74.6	87.2
AllTracker [7]	61.9	75.2	87.8	80.7	90.1	91.7	59.3	71.3	89.3	70.1	82.2	92.2	68.0	79.7	90.3
CoWTracker [8]	**65.5**	**78.0**	**92.1**	**85.4**	**92.8**	**94.9**	**60.9**	**73.1**	**91.5**	**73.2**	**83.4**	**94.7**	**71.3**	**81.8**	**93.3**
Online															
TAP-Net [21]	33.0	48.6	78.8	53.5	68.1	86.3	38.5	54.4	80.6	45.1	62.1	82.9	42.5	58.3	82.2
MFT [17]	47.3	66.8	77.8	–	–	–	39.6	60.4	72.7	–	–	–	43.5	63.6	75.3
DOT† [34]	53.3	67.8	85.4	61.3	75.3	86.5	45.3	58.0	81.4	51.9	62.9	79.9	53.0	66.0	83.3
Online TAPIR [33]	56.7	70.2	85.7	67.7	78.6	89.5	51.5	64.4	85.2	59.1	69.9	86.1	58.8	70.8	86.6
CoTracker3$_{S=1}$ [5]	56.0	70.8	87.0	64.3	80.9	88.6	–	–	–	52.9	70.1	82.6	57.7	73.9	86.1
AllTracker$_{S=1}$ [7]	58.5	71.1	85.9	77.6	88.3	91.4	55.3	68.4	87.0	64.5	77.9	89.0	64.0	76.4	88.3
SPOT [36]	61.5	75.0	88.9	73.3	86.4	90.3	50.2	62.7	85.5	60.7	73.6	87.5	61.4	74.4	88.1
ReTracker [6]	63.7	**77.5**	88.2	68.2	80.7	85.8	54.5	67.0	86.4	65.5	78.1	89.9	63.0	75.8	87.6
DeMeTer	**64.0**	76.7	**90.3**	**79.4**	**88.6**	**93.8**	**59.2**	**72.5**	**90.5**	**71.1**	**82.8**	**93.3**	**68.4**	**80.2**	**92.0**

Table 2. Comparison with AllTracker on a broader evaluation set using δ_{avg}. AllTracker is evaluated offline, while DeMeTer is fully online. Despite this constraint, DeMeTer achieves better average performance.

Method	Cro.	Dav.	Dri.	Ego.	Kin.	Rgb.	Rob.	**Avg.**
AllTracker (offline)	**42.3**	75.2	66.1	60.3	71.3	**90.1**	82.2	69.6
DeMeTer	42.1	**76.7**	**68.4**	**62.1**	**72.5**	88.6	**82.8**	**70.5**

(180) that could fit on an 80GB GPU. SPOT is relatively efficient (19.3 seconds) but gradually loses point coverage during large motions, with many tracks disappearing from the fox by the end of the sequence. ReTracker, a sparse tracker, preserves more point identities but is substantially slower (40 minutes) and produces increasingly fragmented tracks over time. CoWTracker produces high-quality dense tracks; however, without approximations and implementation optimizations, its 79 GB VRAM requirement and restriction to sequences shorter than approximately 1.5 minutes represent significant practical limitations. By contrast, DeMeTer maintains dense and spatially coherent tracks over the fox throughout the sequence, while completing full inference in 12.3 seconds (17.6 FPS) using only 4.2 GB of VRAM. Despite operating fully causally, the visual quality of DeMeTer is comparable to that of the strongest offline method, while being substantially more efficient and suitable for real-time deployment.

3.3. Optical Flow. DeMeTer is further evaluated on optical flow estimation by treating each frame pair as a two-frame video. This evaluation is zero-shot: we use the same model trained for point tracking without any flow-specific fine-tuning. Results are reported on Sintel [45], KITTI [46], and Spring [47] using standard optical flow metrics. On Sintel, *Clean* denotes simpler rendering, while *Final* includes realistic effects such as motion blur, defocus blur, and atmospheric effects. End-Point Error (EPE) measures the average distance between predicted and ground-truth flow vectors; Fl-all reports the percentage of erroneous pixels on KITTI; and 1px error measures the percentage of pixels whose flow error is greater than one pixel on the Spring dataset.

Table 3 reports the optical flow results, including the full AllTracker model trained on Sintel and KITTI [7]. Without any flow-specific fine-tuning, DeMeTer transfers reasonably well to dense two-frame correspondence on visible regions but still lags behind specialized optical flow models trained directly for flow estimation. Compared with dense point trackers, DeMeTer outperforms the Kubric-only AllTracker variant, showing that global appearance matching provides a stronger correspondence signal for both sparse and dense tasks.

The largest gap appears on KITTI, where the prediction error is evaluated also on the invisible (occluded) points. This is expected because DeMeTer is trained to report locations only for visible points. Consistent with this, performance improves substantially on the non-occluded KITTI subset, where DeMeTer obtains 2.16 EPE and 9.79 Fl-noc. Fine-tuning with optical-flow supervision would likely reduce this gap, as observed in

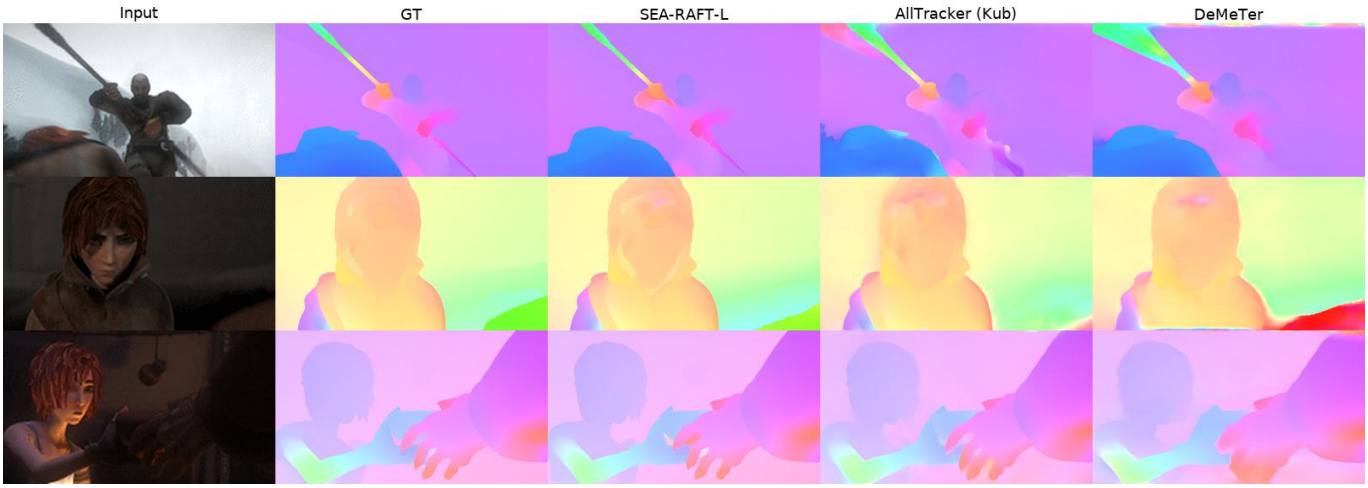

Figure 3. Qualitative comparison of optical flow predictions on Sintel Final [45]. DeMeTer recovers coherent large-scale motion but produces noisier estimates near boundaries and occlusion regions.

Table 3. Zero-shot optical flow results on Sintel [45], KITTI [46], and Spring [47]. We evaluate Sintel on matched/visible correspondences, and KITTI and Spring on their training splits. DeMeTer is trained only with point-tracking supervision and evaluated without flow-specific fine-tuning. It is competitive on visible correspondences but weaker on KITTI, which evaluates occluded points.

	Sintel (*matched*)		KITTI (*train*)		Spring (*train*)	
Method	Clean↓	Final↓	EPE↓	Fl-all↓	EPE↓	1px↓
Optical Flow models						
RAFT [14]	0.62	1.40	1.53	7.81	0.45	4.79
SEA-RAFT-L [16]	0.44	1.20	1.60	8.26	0.41	4.06
Dense Point Trackers						
AllTracker [7]	0.65	1.74	–	–	–	–
AllTracker (Kub) [7]	2.38	3.36	20.15	43.93	1.14	9.20
CoWTracker [8]	0.78	1.48	1.04	4.87	–	–
DeMeTer	1.32	2.41	10.72	21.68	0.96	14.97

prior work [7].

Figure 3 visually compares optical flow predictions on Sintel Final. DeMeTer captures the dominant motion structure but produces less sharp flow near object boundaries and occlusions than specialized optical flow models. A training objective that supervises occluded correspondences would likely improve optical-flow transfer. However, tracking occluded points does not naturally fit the global matching setup used in our model, and our point-tracking results suggest that explicit occluded-point supervision is not necessary for strong causal tracking performance.

Table 4. Architectural ablation results on our evaluation suite. Global matching is the most important component, while appearance and motion memories provide complementary temporal information.

Ablation	$\delta_{\text{avg}} \uparrow$
Default	**66.1**
w/o global matching	53.5
Learned queries	58.1
FIFO appearance memory	65.8
w/o appearance memory	65.4
w/o motion memory	65.3
w/o both memories	63.6

3.4. Ablations. We ablate the main architectural choices of DeMeTer by evaluating each variant on a combined benchmark suite and reporting δ_{avg}, averaged across datasets with weights proportional to the number of sequences in each dataset. Specifically, we use CroHD [41], DriveTrack [42], TAPVid-DAVIS [21], TAPVid-Kinetics [21], RGB-Stacking [43], RoboTAP [33], and EgoPoints [44]. The ablated models are trained for 100k steps on two GPUs with a learning rate of 1.5×10^{-4} and a reduced sequence length

of 16. We use this reduced setup to compare design choices under the same training regime and evaluate the final model separately in the main experiments.

Individual components. Table 4 ablates the main components of DeMeTer. Removing global matching causes the largest degradation, reducing δ_{avg} by 12.6%, showing that explicit dense matching provides a strong initialization signal for refinement. Replacing reference-frame features with learned queries also reduces performance by 8.0%, suggesting that query-frame appearance is important for preserving point identities. Removing appearance memory or motion memory individually leads to smaller drops of 0.7% and 0.8%, respectively. Removing both memories results in a larger drop of 2.5%, suggesting that appearance and motion memories provide complementary temporal information.

Table 5. Effect of iterative refinement on our benchmark set. Global matching alone provides a coarse but meaningful initialization, while refinement improves localization and temporal consistency. DeMeTer achieves its best performance after four refinement steps.

	$\delta_{\text{avg}} \uparrow$	
Step	AllTracker	DeMeTer
Global Matching	–	46.5
Refiner Step 1	66.2	60.8
Refiner Step 2	68.9	61.8
Refiner Step 3	69.1	68.3
Refiner Step 4	69.6	70.2

Refinement steps. Table 5 analyzes the effect of global matching and iterative refinement. The global matching stage alone provides a useful dense initialization, and performance improves substantially after the first refinement step. Additional steps yield incremental gains, with performance saturating after three to four iterations. DeMeTer uses four refinement steps in the final model. For time-critical applications, fewer refinement steps can be used with only a moderate loss in accuracy. Unlike AllTracker, DeMeTer starts from an explicit global match and then refines locally, rather than relying primarily on refinement from a propagated estimate, resulting in a higher final average accuracy.

Table 6. Memory size ablation results. Appearance memory is robust to moderate changes in size, while motion memory requires sufficient temporal history; Reducing motion memory to a single slot causes a large drop because at least two recent states are needed to reconstruct a local motion trend.

Ablation	Appearance mem.	Motion mem.	$\delta_{\text{avg}} \uparrow$
Default	6	16	66.1
Appearance memory size			
Appearance mem. = 4	4	16	66.1
Appearance mem. = 2	2	16	66.4
Appearance mem. = 1	1	16	66.4
Motion memory size			
Motion mem. = 8	6	8	66.1
Motion mem. = 4	6	4	66.0
Motion mem. = 2	6	2	65.9
Motion mem. = 1	6	1	56.8

Memory size. Table 6 examines the effect of appearance and motion memory size. Reducing the appearance memory does not degrade performance, suggesting that the static reference appearance and a small set of recent reliable appearances are sufficient for matching. In contrast, reducing the motion memory has a stronger effect: performance decreases gradually with fewer memory slots and drops sharply when only one slot is used. We interpret this drop as evidence that the refiner benefits from multiple recent motion states, which provide a short-term motion prior for regularizing predictions when visual information is weak.

4. Conclusion

We introduced DeMeTer, a dense online point tracker that decomposes point location estimation into two stages: (i) global visual matching for robust re-detection and (ii) local motion-based refinement. Both stages exploit a novel memory-based approach, preserving long-term point identities while leveraging recent motion context to refine ambiguous

predictions. Across seven point-tracking benchmarks and three flow-estimation datasets, DeMeTer demonstrates strong online performance, achieving state-of-the-art results on point tracking and improving average δ_{avg} by 4.1% over the strongest online dense tracker. The ablations show that appearance memory enables robust initial localization, while motion memory provides a short-term motion prior that acts as regularization when visual evidence is weak. DeMeTer also remains competitive with offline trackers while operating online, which is crucial for long sequences and reactive systems, and provides strong dense flow estimates despite never being trained on optical flow. We hope this work encourages future long-term trackers to preserve explicit re-detection as they scale to dense, efficient, and accurate online tracking.

Bibliography

1. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *IJCV* 60(2):91–110.
2. Doersch C et al. (2023) TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement in *ICCV*. pp. 10061–10072.
3. Cho S et al. (2024) Local All-Pair Correspondence for Point Tracking in *ECCV*. (Springer), pp. 306–325.
4. Karaev N et al. (2024) CoTracker: It is Better to Track Together in *ECCV*. (Springer), pp. 18–35.
5. Karaev N et al. (2025) CoTracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos in *ICCV*. pp. 6013–6022.
6. Tan D et al. (2025) ReTracker: Exploring Image Matching for Robust Online Any Point Tracking in *ICCV*. pp. 4306–4316.
7. Harley AW et al. (2025) AllTracker: Efficient Dense Point Tracking at High Resolution in *ICCV*. pp. 5253–5262.
8. Lai Z, Insafutdinov E, Sucar E, Vedaldi A (2026) CoWTracker: Tracking by Warping instead of Correlation. *arXiv preprint arXiv:2602.04877*.
9. Horn BK, Schunck BG (1981) Determining Optical Flow. *Artificial intelligence* 17(1-3):185–203.
10. Brox T, Bruhn A, Papenberg N, Weickert J (2004) High Accuracy Optical Flow Estimation Based on a Theory for Warping in *ECCV*. (Springer), pp. 25–36.
11. Lucas BD, Kanade T (1981) An Iterative Image Registration Technique with an Application to Stereo Vision in *IJCAI*. Vol. 2, pp. 674–679.
12. Dosovitskiy A et al. (2015) FlowNet: Learning Optical Flow with Convolutional Networks in *ICCV*. pp. 2758–2766.
13. Sun D, Yang X, Liu MY, Kautz J (2018) PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume in *CVPR*. pp. 8934–8943.
14. Teed Z, Deng J (2020) RAFT: Recurrent All-Pairs Field Transforms for Optical Flow in *ECCV*. (Springer), pp. 402–419.
15. Huang Z et al. (2022) FlowFormer: A Transformer Architecture for Optical Flow in *ECCV*. (Springer), pp. 668–685.
16. Wang Y, Lipson L, Deng J (2024) SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow in *ECCV*. (Springer), pp. 36–54.
17. Neoral M, Šerỳch J, Matas J (2024) MFT: Long-Term Tracking of Every Pixel in *WACV*. pp. 6837–6847.
18. Wu G et al. (2023) AccFlow: Backward Accumulation for Long-Range Optical Flow in *ICCV*. pp. 12119–12128.
19. Tournadre M, Soladié C, Stoiber N, Richard PY (2024) Dense Trajectory Fields: Consistent and Efficient Spatio-Temporal Pixel Tracking in *ACCV*. pp. 2212–2230.
20. Ngo T et al. (2025) DELTA: Dense Efficient Long-range 3D Tracking for any video in *ICLR*. Vol. 2025, pp. 44620–44641.
21. Doersch C et al. (2022) TAP-Vid: A Benchmark for Tracking Any Point in a Video. *NIPS* 35:13610–13626.
22. Harley AW, Fang Z, Fragkiadaki K (2022) Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories in *ECCV*. (Springer), pp. 59–75.
23. Zheng Y, Harley AW, Shen B, Wetzstein G, Guibas LJ (2023) PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking in *ICCV*. pp. 19855–19865.
24. Jung S et al. (2026) TAPNext++: What's Next for Tracking Any Point (TAP)? in *CVPR*. pp. 8429–8438.
25. Li H et al. (2024) TAPTR: Tracking Any Point with Transformers as Detection in *ECCV*. (Springer), pp. 57–75.
26. Li H et al. (2024) TAPTRv2: Attention-based Position Update Improves Tracking Any Point. *NeurIPS* 37:101074–101095.
27. Zholus A et al. (2025) TAPNext: Tracking Any Point (TAP) as Next Token Prediction in *ICCV*. pp. 9693–9703.
28. Doersch C et al. (2024) BootsTAP: Bootstrapped Training for Tracking-Any-Point in *ACCV*. pp. 3257–3274.
29. Zhang B, Ke L, Harley AW, Fragkiadaki K (2025) TAPIP3D: Tracking Any Point in Persistent 3D Geometry. *arXiv preprint arXiv:2504.14717*.
30. Feng H et al. (2025) St4RTrack: Simultaneous 4D Reconstruction and Tracking in the World in *ICCV*. pp. 8503–8513.
31. Rajič F et al. (2025) Multi-View 3D Point Tracking in *ICCV*. pp. 59–68.
32. Koo J et al. (2025) MV-TAP: Tracking Any Point in Multi-View Videos. *arXiv preprint arXiv:2512.02006*.
33. Vecerik M et al. (2024) RoboTAP: Tracking Arbitrary Points for Few-Shot Visual Imitation in *ICRA*. (IEEE), pp. 5397–5403.
34. Le Moing G, Ponce J, Schmid C (2024) Dense Optical Tracking: Connecting the Dots in *CVPR*. pp. 19187–19197.
35. Kim IH et al. (2025) Exploring Temporally-Aware Features for Point Tracking in *CVPR*. pp. 1962–1972.
36. Dong Q, Fu Y (2025) Online Dense Point Tracking with Streaming Memory in *ICCV*. pp. 8710–8720.
37. Liu Z et al. (2022) A ConvNet for the 2020s in *CVPR*. pp. 11976–11986.
38. Shi W et al. (2016) Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network in *CVPR*. pp. 1874–1883.
39. Greff K et al. (2022) Kubric: A scalable dataset generator in *CVPR*. pp. 3749–3761.
40. Loshchilov I, Hutter F (2019) Decoupled Weight Decay Regularization in *ICLR*.
41. Sundararaman R, De Almeida Braga C, Marchand E, Pettre J (2021) Tracking Pedestrian Heads in Dense Crowd in *CVPR*. pp. 3865–3875.
42. Balasingam A, Chandler J, Li C, Zhang Z, Balakrishnan H (2024) DriveTrack: A Benchmark for Long-Range Point Tracking in Real-World Videos in *CVPR*. pp. 22488–22497.
43. Lee AX et al. (2021) Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes in *CoRL*.
44. Darkhalil A, Guerrier R, Harley AW, Damen D (2025) EgoPoints: Advancing Point Tracking for Egocentric Videos in *WACV*.
45. Butler DJ, Wulff J, Stanley GB, Black MJ (2012) A Naturalistic Open Source Movie for Optical Flow Evaluation in *ECCV*. (Springer), pp. 611–625.
46. Geiger A, Lenz P, Urtasun R (2012) Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite in *CVPR*. (IEEE), pp. 3354–3361.
47. Mehl L, Schmalfuss J, Jahedi A, Nalivayko Y, Bruhn A (2023) Spring: A High-Resolution High-Detail Dataset and Benchmark for Scene Flow, Optical Flow and Stereo in *CVPR*. pp. 4981–4991.