Online Long-term Point Tracking in the Foundation Model Era

by

Görkay Aydemir

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

July 9, 2025

Online Long-term Point Tracking in the Foundation Model Era

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Görkay Aydemir

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Fatma Güney (Advisor)

_____

Assoc. Prof. Christian Rupprecht

_____

Prof. Ross Goroshin

Date: _____

To free and independent academia.

ABSTRACT

Online Long-term Point Tracking in the Foundation Model Era
Görkay Aydemir
Master of Science in Computer Science and Engineering
July 9, 2025

Point tracking aims to identify the same physical point across video frames and serves as a geometry-aware representation of motion. This representation supports a wide range of applications, from robotics to augmented reality, by enabling accurate modeling of dynamic environments. Most existing long-term tracking approaches operate in an offline setting, where future frames are available to refine predictions and recover from occlusions. However, real-world scenarios often demand online predictions: the model must operate causally, using only current and past frames. This constraint is critical in streaming video and embodied AI, where decisions must be made immediately based on past observations. Under such constraints, viewpoint invariance becomes essential. Visual foundation models, trained on diverse large-scale datasets, offer the potential for robust geometric representations. While they lack temporal reasoning on their own, they can be integrated into tracking pipelines to enrich spatial features and improve robustness. In this thesis, we address the problem of long-term point tracking in an online setting, where frames are processed sequentially without access to future information or sliding windows. We begin by evaluating the suitability of visual foundation models for this task and find that they can serve as useful initializations and be integrated into tracking pipelines. However, to enable long-term tracking in an online setting, a dedicated design is still required. In particular, maintaining coherence over time in this causal regime requires memory to propagate appearance and context across frames. To address this, we introduce **Track-On**, a transformer-based model that treats each tracked point as a query and processes video frames one at a time. It predicts correspondences through patch classification followed by local refinement. To ensure temporal consistency, memory modules are incorporated to carry relevant information across frames. Track-On sets a new state of the art across seven public benchmarks, demonstrating the feasibility of long-term tracking without future access.

ÖZETÇE

Temel Modeller Çağında Uzun Vadeli Çevrimiçi Nokta İzleme
Görkay Aydemir
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
9 Temmuz 2025

Nokta izleme, aynı fiziksel noktanın video çerçeveleri boyunca tanımlanmasını hedefler ve hareketin geometriye duyarlı bir temsili olarak işlev görür. Bu temsil, dinamik ortamların doğru modellenmesini sağlayarak robotikten artırılmış gerçekliğe kadar çeşitli uygulamalara olanak tanır. Mevcut uzun vadeli izleme yöntemlerinin çoğu, gelecekteki çerçevelerin erişilebilir olduğu çevrimdışı senaryolarda çalışır. Oysa gerçek dünyada, modelin yalnızca geçmiş ve mevcut çerçeveleri kullanarak nedensel şekilde çalışması gereken çevrimiçi tahminler gereklidir. Bu durumda, bakış açısına karşı değişimsizlik kritik hâle gelir. Büyük ve çeşitli veri kümeleri üzerinde eğitilen görsel temel modeller, sağlam geometrik temsiller sunabilir. Zamanla ilgili çıkarım yetenekleri sınırlı olsa da, bu modeller izleme sistemlerine entegre edilerek mekansal özellikleri zenginleştirebilir. Bu tezde, geleceğe erişim olmadan çerçevelerin sırayla işlendiği çevrimiçi ortamda uzun vadeli nokta izleme problemini ele alıyoruz. Görsel temel modellerin bu görevde yararlı başlangıç noktaları sunduğunu gösteriyoruz; ancak uzun vadeli izleme için özel bir tasarım gereklidir. Bu amaçla her noktayı bir sorgu olarak işleyen ve video çerçevelerini sırasıyla değerlendiren transformatör tabanlı bir model olan Track-On'u sunuyoruz. Yama sınıflandırması ve yerel iyileştirme ile eşleşmeleri tahmin ederken, bellek modülleriyle zaman boyunca tutarlılığı korur. Track-On, yedi denektaşında geleceğe erişim olmadan uzun vadeli izlemenin mümkün olduğunu göstermektedir.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Asst. Professor Fatma Güney, for her continuous support, encouragement, and invaluable guidance throughout the years. I am also sincerely thankful to Assoc. Professor Weidi Xie, whose long-term collaboration and insightful mentorship have played a crucial role in shaping the direction and execution of my research.

I would like to thank Assoc. Professor Christian Rupprecht and Professor Ross Goroshin for kindly agreeing to serve on my thesis committee and for their time and thoughtful evaluation of my work.

I am also grateful to all members of the AVG group for fostering a collaborative research environment, with special thanks to Shadi for his generous help and support.

Beyond the academic journey, I owe everything to my family. Their constant encouragement, quiet sacrifices, and steady belief in me have been a source of strength through every step of this process. I am especially thankful to my mother, Özlem, and my father, Hakan, for their endless support, which has shaped who I am. In a world where everything shifts, their presence has been a quiet constant.

Among my friends, I would like to thank Akın, for being like an older brother, even in academic matters; Mustafa, whose presence transformed my undergraduate years and who became the main figure of my two-year-long Istanbul chapter; and Serdar, with whom I have built not just a friendship but a shared worldview. Together, as "Serdar and Görkay," we have created a space against the world, where, in the face of absurdity, we embrace the absence of meaning.

Finally, I want to thank Duygu, whose presence brought clarity to a process often clouded by uncertainty. Her patience, warmth, and belief in me never wavered, even during the most difficult periods. I am deeply grateful for her companionship, and for always helping me see beyond the immediate, to what truly matters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
AJ	Average Jaccard
CNN	Convolutional Neural Network
FoMo	Foundation Model
IME	Inference-Time Memory Extension
LoRA	Low-Rank Adaptation
MLP	Multi-Layer Perceptron
OA	Occlusion Accuracy
SD	Stable Diffusion
TAP	Tracking-Any-Point
ViT	Vision Transformer

Chapter 1

INTRODUCTION

Motion estimation is one of the core challenges in computer vision, with applications spanning video compression [Jasinschi et al., 1998], video stabilization [Battiato et al., 2007, Lee et al., 2009], and augmented reality [Marchand et al., 2015]. The objective is to track physical points across video frames accurately. A widely used solution is optical flow, which estimates pixel-level correspondences between adjacent frames. In principle, long-term motion estimation can be achieved by chaining together these frame-by-frame estimations.

Recent advances in optical flow techniques, such as PWC-Net [Sun et al., 2018] and RAFT [Teed and Deng, 2020], have significantly improved short-term accuracy. However, chaining flow predictions over long time horizons remains problematic due to error accumulation and challenges in handling occlusions.

For example, in Figure 1.1, long-term correspondences are obtained by chaining short-term optical flow predictions. While this works in unobstructed regions, the chain fails to recover occluded points: all predictions are mistakenly aligned with the tree trunk, even though the actual points lie on the cycling child.

To address this, [Sand and Teller, 2008] proposed pixel tracking, a paradigm shift that focuses on tracking individual points through time. This idea, revisited in PIPs [Harley et al., 2022] and TAPIR [Doersch et al., 2023], leverages deep features and cost volumes to maintain correspondence, a task commonly referred to as tracking-any-point (TAP), or simply, point tracking.

Point tracking extends the correspondence problem into a long-term setting [Harley et al., 2022, Doersch et al., 2022], where the objective is to estimate the 2D projection, *i.e.* the location in each video frame, of the same physical point throughout the

Figure 1.1: **Tracking with Optical Flow.** When short-term optical flow predictions are chained to achieve long-term tracking, they fail to recover from occlusions. In this example, all predictions (red) collapse onto the tree trunk, while the correct locations (green) are on the cycling child.

video. This requires overcoming significant challenges such as appearance changes, occlusions, and complex motion. The task demands a high level of geometric awareness, as it goes beyond pairwise frame matching and requires consistent reasoning over time. Importantly, long-term point tracking plays a fundamental role in many real-world applications: in robotics, it enables precise object manipulation [Vecerik et al., 2023]; in augmented reality, it allows stable anchoring of virtual content; in autonomous navigation, it supports robust motion estimation and scene understanding; and in 3D vision, it benefits tasks like structure-from-motion and SLAM by providing reliable feature trajectories.

A key requirement in this setting is viewpoint invariance: the ability to recognize and follow a physical point despite changes in camera pose, perspective, and lighting. As the visual appearance of a point may vary drastically under motion, trackers must rely on geometric features rather than raw appearance. This makes geometry-aware feature representations, particularly those that are invariant to viewpoint transformations, highly desirable.

1.1 Foundation Models for Point Tracking

Visual foundation models (FoMos), pretrained on large-scale and diverse datasets, have demonstrated strong generalization across a variety of downstream tasks, in-

cluding classification [Radford et al., 2021], segmentation [Wang et al., 2023b], and object localization [Melas-Kyriazi et al., 2022]. Their ability to extract semantically and geometrically meaningful features makes them attractive candidates for point tracking. Specifically, their potential for viewpoint-invariant representations may help mitigate appearance variations and occlusion-induced drift in long-term tracking.

Matching and correspondence estimation provide a natural way to probe the geometric capacity of such models. If a model can reliably associate pixels corresponding to the same physical point across views or time, this indicates a strong inductive bias toward geometric consistency. Indeed, several recent studies [Tang et al., 2023, Zhang et al., 2024b, El Banani et al., 2024] have explored the correspondence capabilities of foundation models, primarily focusing on two-view matching.

However, point tracking goes beyond static matching: it extends correspondence estimation into a dynamic and long-term context. Here, the model must handle ongoing changes in appearance, scene dynamics, and visibility. Successfully tracking a single point through an entire video requires consistent and temporally coherent representations, something that simple two-view matching does not guarantee.

This raises a natural question: do the powerful representations learned by visual foundation models contain sufficient geometric and temporal structure to support long-term point tracking, without any task-specific supervision? Unlike models trained explicitly for correspondence, FoMos are optimized for general visual understanding, often via contrastive or masked pretraining objectives. Despite this, their dense features often encode surprising spatial regularities and object-level consistency.

To answer this, we evaluate whether these models can serve as a basis for long-term point tracking. Specifically, we investigate their capabilities in a zero-shot setting without any task-specific training, and analyze how well they can be adapted when supervision is available. This allows us to probe the geometric inductive biases of foundation models and understand their potential and limitations as generic backbones for temporally consistent tracking.

In the first part of this thesis, we systematically assess whether foundation models

can be repurposed for point tracking, and to what extent they can represent and maintain long-term geometric correspondences.

While foundation models offer strong geometric representations, they must be integrated into full tracking pipelines to realize their potential in real-world settings. In particular, long-term point tracking requires not only spatially rich features but also temporal reasoning, a component missing from most frozen backbones.

1.2 Online Point Tracking

Even when correspondence-aware features are used within existing methods, already achieving strong performance, most models remain fundamentally offline. They process entire videos or large frame windows, leveraging both past and future information to refine predictions (Figure 1.2, left). While effective, this design limits their applicability in real-time scenarios [Karaev et al., 2024b, Harley et al., 2022], where predictions must be made immediately without access to future frames. For instance, in streaming video analysis or robotics, the environment evolves in response to the agent's current actions, requiring the model to operate causally and produce predictions on the fly.

Furthermore, many offline methods rely on full spatiotemporal attention across video sequences, leading to significant memory overhead and poor scalability to longer sequences. These limitations motivate the need for online models that can reason about motion and visibility in a streaming setting, without sacrificing tracking accuracy or robustness.

Prior work on online point tracking remains limited. Online-TAPIR [Vecerik et al., 2023] adapts an offline model (TAPIR [Doersch et al., 2023]) by applying a causal attention mask and retraining it for sequential inference. While this enables causal operation, the model architecture remains offline in its original design, it does not incorporate components specifically tailored for online tracking, such as mechanisms for maintaining and updating memory over time. In contrast, tasks like video object segmentation [Ravi et al., 2025] and action recognition [Xu et al., 2021] have demonstrated the effectiveness of purpose-built memory modules in capturing

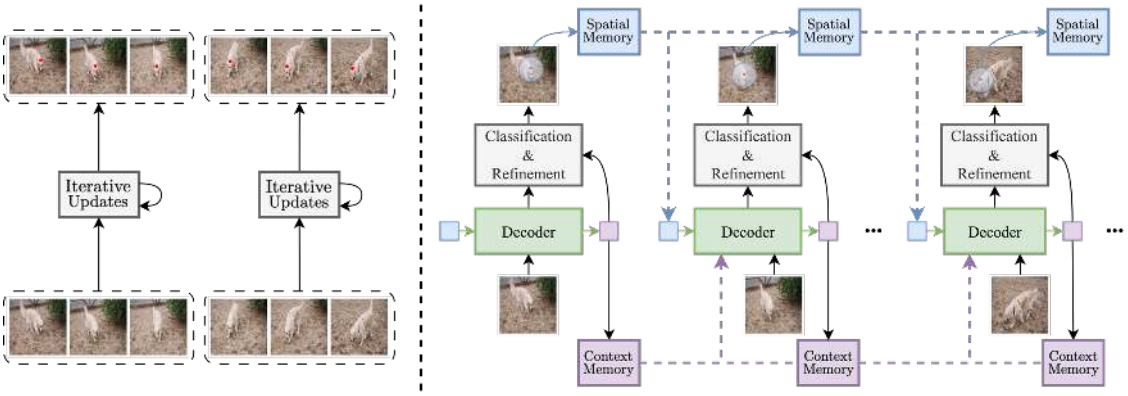

Figure 1.2: **Offline vs. Online Point Tracking.** We propose an online model, tracking points frame-by-frame (**right**), unlike the dominant offline paradigm where models require access to all frames within a sliding window or the entire video (**left**). In contrast, our approach allows for frame-by-frame tracking in videos of any length. To capture temporal information, we introduce two memory modules: spatial memory, which tracks changes in the target point, and context memory, which stores broader contextual information from previous states of the point.

long-term temporal dependencies. These insights suggest that achieving reliable online point tracking requires architectural choices that go beyond causal masking, including dedicated memory structures that support temporal continuity and robust appearance adaptation.

In the second part of this thesis, we address the challenge of long-term point tracking in an online setting (Figure 1.2, right), where the model processes video frames sequentially without access to future frames. We propose a simple transformer-based model, **Track-On**, in which points of interest are treated as queries in a transformer decoder that attends to the current frame to update their representations. Rather than relying on full temporal modeling, our approach maintains temporal continuity through two specialized memory modules: *spatial memory* and *context memory*. This enables the model to perform reliable long-term tracking while avoiding the high computational and memory costs associated with dense video-wide attention.

Specifically, spatial and context memory play distinct but complementary roles. The former reduces drift by updating the query representation using recent visual content around the tracked point. This ensures that the query reflects the most recent appearance, rather than the initial one. The latter maintains a broader summary of the track's history by storing past query embeddings, capturing long-term evolution and occlusion events. Together, these modules allow the model to reason over time while operating causally and efficiently.

At training time, the queries identify the most likely location by computing embedding similarity with patches in the current frame and are supervised using similarity-based classification, akin to contrastive learning. The prediction is further refined through a local offset regression to determine precise coordinates. We show through extensive experiments that our patch-classification and refinement approach offers a compelling alternative to dominant iterative update schemes [Karaev et al., 2024b, Harley et al., 2022, Doersch et al., 2023]. Our model sets a new state-of-the-art among online methods and achieves performance on par with or surpassing offline models across seven benchmark datasets, including TAP-Vid [Doersch et al., 2022].

1.3 Findings

In this thesis, we demonstrated that visual foundation models can provide powerful initializations for long-term point tracking. Although not explicitly trained for correspondence, their representations offer strong spatial regularities that can be leveraged for tracking tasks. Our systematic evaluation showed that, with minimal adaptation, these models encode geometric structure that supports temporally consistent predictions.

Building on this insight, we introduced **Track-On**, a simple yet effective transformer-based model for online point tracking. Unlike dominant iterative update approaches optimized with regression, Track-On formulates point tracking as a patch classification problem. Our results show that this formulation, combined with a lightweight refinement step via local offset prediction, can achieve high-precision correspon-

dences while avoiding the complexity of iterative updates.

A central component of our model is its memory design. Our spatial memory module reduces the effects of drift by continually updating the query representation based on recent visual context. In parallel, the context memory provides a longer-term view by accumulating query embeddings over time, enabling the model to reason about appearance changes and occlusions. Importantly, the memory size can be fixed during training and flexibly extended at inference, allowing the model to generalize to arbitrarily long videos.

Track-On achieves state-of-the-art performance among online trackers and matches or surpasses offline baselines across a diverse set of benchmarks: from high-quality object-centric datasets like DAVIS, to large-scale internet videos such as Kinetics, to synthetic environments like Dynamic Replica, and very-long sequences in Point Odyssey. Crucially, our model remains highly efficient: each frame is processed independently with a single backbone forward pass, while relevant past information is compactly stored and accessed through our memory design.

These results underscore the potential of online point tracking as a scalable and practical alternative to traditional offline approaches. By combining causal processing with effective memory mechanisms, Track-On closes the gap between real-time operation and long-term correspondence. We believe that online point tracking will play an increasingly central role in dynamic vision systems, enabling deployment in real-world settings such as robotics, augmented reality, and embodied AI.

1.4 Thesis Outline and Contributions

This thesis is organized as follows: Chapter 2 reviews related work on optical flow, long-term point tracking, visual foundation models, and online tracking architectures with memory-based designs.

Chapter 3 presents an analysis [Aydemir et al., 2024][1] of visual foundation models in the context of long-term point tracking under zero-shot settings. We show

[1]Project page: https://kuis-ai.github.io/fomo_pt

that certain models offer strong geometric representations that serve as effective initializations for temporal correspondence tasks, and that lightweight adaptation can match or exceed the performance of fully supervised baselines.

Chapter 4 introduces **Track-On** [Aydemir et al., 2025][2], a transformer-based model for online point tracking that treats target points as queries and performs correspondence estimation via patch classification and refinement. Through extensive experiments and ablations, we demonstrate that Track-On achieves state-of-the-art performance among online trackers and performs competitively with offline methods.

Finally, Chapter 5 concludes the thesis and outlines several directions for future work to further advance the capabilities of long-term and online tracking.

In addition to the two main contributions presented in this thesis, two other research projects were completed during the same period but are not included in the core chapters: (i) **SOLV** [Aydemir et al., 2023b][3], the first fully unsupervised method for segmenting multiple objects in real-world videos using an object-centric approach. By combining a novel masking strategy with slot merging based on similarity, SOLV effectively segments diverse object classes in challenging YouTube videos. (ii) **ADAPT** [Aydemir et al., 2023a][4], a method for predicting the trajectories of all agents in complex traffic scenes. Leveraging dynamic weight learning and an adaptive prediction head, ADAPT achieves superior performance and efficiency compared to prior multi-agent forecasting methods.

Taken together, these works reflect a unified research centered on modeling temporal dynamics, spanning object-centric reasoning, multi-agent prediction, and point-level tracking. This thesis contributes to the broader goal of building models that understand and act upon time-varying information across diverse domains, from autonomous driving to general-purpose video understanding.

[2]Track-On project page: https://kuis-ai.github.io/track_on

[3]SOLV project page: https://kuis-ai.github.io/solv

[4]ADAPT project page: https://kuis-ai.github.io/adapt

Chapter 2

RELATED WORK

2.1 Optical Flow

Optical flow estimation has long been a fundamental problem in computer vision, addressing the pixel-wise correspondence between consecutive frames. Classical methods [Horn and Schunck, 1981, Black and Anandan, 1993, Bruhn et al., 2005] laid the foundation through energy minimization and variational approaches. With the advent of deep learning, FlowNet [Dosovitskiy et al., 2015] introduced the first end-to-end trainable neural architectures for optical flow. Subsequently, methods incorporating explicit cost volume computations, such as DCFlow [Xu et al., 2017] and PWC-Net [Sun et al., 2018], significantly improved local correspondence matching. RAFT [Teed and Deng, 2020], refined this further by iteratively updating flow predictions via recurrent processing of cost volumes (Figure 2.1), achieving notable accuracy and inspiring subsequent methods [Zhang et al., 2021a, Shi et al., 2023]. Despite these advancements, optical flow methods inherently operate between pairs of consecutive frames, limiting their ability to handle long-term occlusions and significant appearance changes [Neoral et al., 2024, Doersch et al., 2022]. Recent approaches aiming to address these limitations typically integrate additional temporal context or explicit tracking mechanisms [Neoral et al., 2024].

2.2 Vision Foundation Models

In recent years, the availability of large-scale datasets and increased computing power has led to the development of deep learning models that can handle various visual tasks. These models are trained on large amounts of data through methods like generative objectives [Rombach et al., 2022], self-supervision [Caron et al., 2021, Oquab et al., 2024], or supervised learning on extensive labeled datasets [Kir-

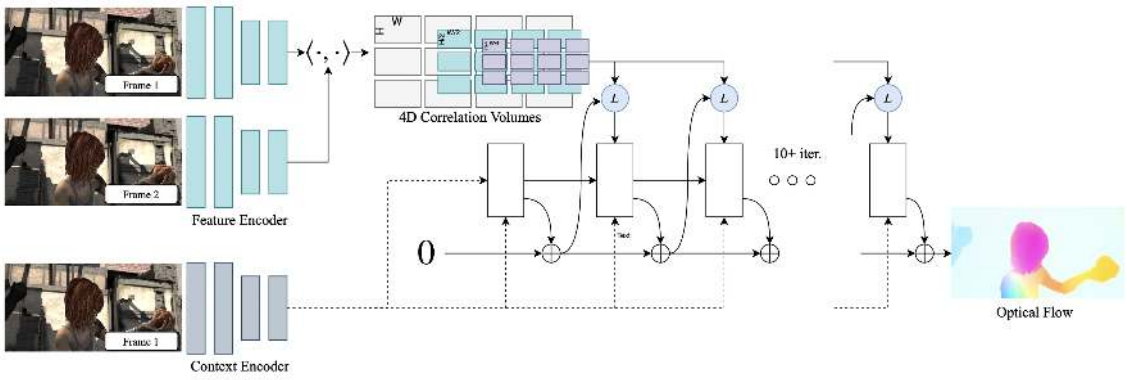

Figure 2.1: Iterative *update operation* introduced in **RAFT** [Teed and Deng, 2020].

illov et al., 2023]. The flexibility of these models allows them to perform different tasks, such as video object segmentation [Wang et al., 2023b, Wang et al., 2023c], estimating correspondence [Hedlin et al., 2023], object-centric learning [Aydemir et al., 2023b], discovering parts [Amir et al., 2021], autonomous driving [Barın et al., 2024], and open-vocabulary segmentation [Xu et al., 2023].

2.3 Point Tracking

Point tracking, presents significant challenges, particularly for long-term tracking where maintaining consistent tracking through occlusions is difficult. PIPs [Harley et al., 2022] was one of the first approaches to address this by predicting motion through iterative updates within temporal windows, as illustrated in Figure 2.2. TAP-Vid [Doersch et al., 2022] initiated a benchmark for evaluation. TAPIR [Doersch et al., 2023] improved upon PIPs by refining initialization and incorporating depthwise convolutions to enhance temporal accuracy. BootsTAPIR [Doersch et al., 2024] further advanced TAPIR by utilizing student-teacher distillation on a large corpus of real-world videos. In contrast, CoTracker [Karaev et al., 2024b] introduced a novel approach by jointly tracking multiple points, exploiting spatial correlations between points via factorized transformers. Differently, TAPTR [Li et al., 2024b] adopted a design inspired by DETR [Carion et al., 2020, Zhu et al., 2021], drawing parallels between object detection and point tracking. DINO-Tracker [Tumanyan

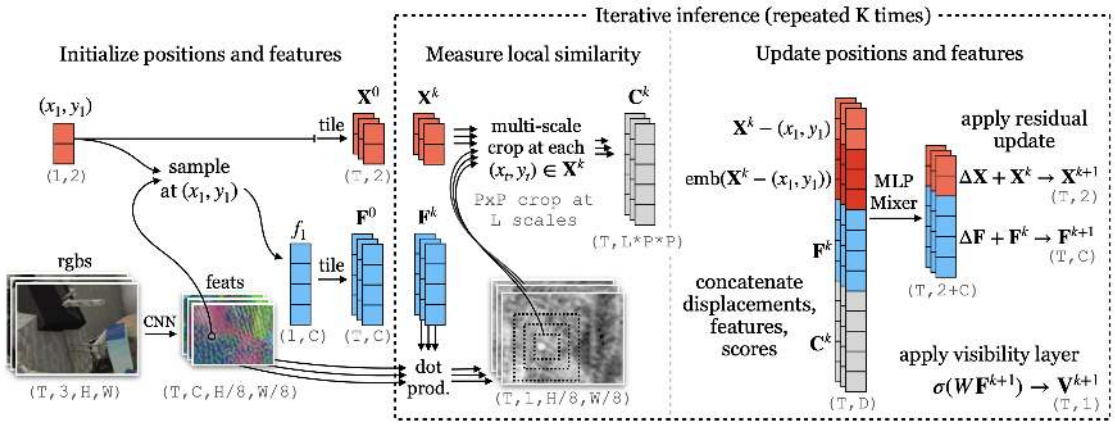

Figure 2.2: Overview of **PIPs** [Harley et al., 2022], iteratively updating the track feature and correspondence prediction in the temporal window.

et al., 2024] took a different route, using DINO as a foundation for test-time optimization. TAPTRv2 [Li et al., 2024a], the successor to TAPTR, builds on its predecessor by incorporating offsets predicted by the deformable attention module. While these models calculate point-to-region similarity for correlation, LocoTrack [Cho et al., 2024] introduced a region-to-region similarity approach to address ambiguities in matching. Recently, CoTracker3 [Karaev et al., 2024a] combined the region-to-region similarity method from LocoTrack with the original CoTracker architecture and utilized pseudo-labeled real-world data during training to further enhance performance.

However, all these models are designed for offline tracking, assuming access to all frames within a sliding window [Karaev et al., 2024b] or the entire video [Doersch et al., 2023, Doersch et al., 2024]. Conversely, MFT [Neoral et al., 2024], which extends optical flow to long-term scenarios, can be adapted for online point tracking tasks, although it does not belong to the point tracking family. Among point tracking approaches, models with online variants [Doersch et al., 2024, Doersch et al., 2023] are re-trained with a temporally causal mask to process frames sequentially on a frame-by-frame basis, despite being originally designed for offline tracking. In contrast, we explicitly focus on online point tracking by design, enabled by novel memory modules to capture temporal information. Additionally, many of

these models use a regression objective, originally developed for optical flow [Teed and Deng, 2020], while we introduce a new paradigm based on patch classification and refinement.

Another line of research, orthogonal to ours, explores leveraging scene geometry for point tracking. SpatialTracker [Xiao et al., 2024] extends CoTracker to the 3D domain by tracking points in three-dimensional space, while OmniMotion [Wang et al., 2023a] employs test-time optimization to learn a canonical representation of the scene. Concurrent work DynOMO [Seidenschwarz et al., 2025] also uses test-time optimization, utilizing Gaussian splats for online point tracking.

2.4 Causal Processing in Videos

Online, or temporally causal models rely solely on current and past frames without assuming access to future frames. This is in contrast to current practice in point tracking with clip-based models, processing frames together. Causal models are particularly advantageous for streaming video understanding [Yang et al., 2022a, Zhou et al., 2024], embodied perception [Yao et al., 2019], and processing long videos [Zhang et al., 2024a, Xu et al., 2021], as they process frames sequentially, making them well-suited for activation caching. Due to its potential, online processing has been studied across various tasks in computer vision, such as pose estimation [Fan et al., 2021, Nie et al., 2019], action detection [Xu et al., 2019, De Geest et al., 2016, Kondratyuk et al., 2021, Eun et al., 2020, Yang et al., 2022b, Wang et al., 2021, Zhao and Krähenbühl, 2022, Xu et al., 2021, Chen et al., 2022a], temporal action localization [Buch et al., 2017, Singh et al., 2017], object tracking [He et al., 2018, Wang et al., 2020], video captioning [Zhou et al., 2024], and video object segmentation [Cheng and Schwing, 2022, Liang et al., 2020].

In causal models, information from past context is commonly propagated using either sequential models [De Geest et al., 2016], which are inherently causal, or transformers with causal attention masks [Wang et al., 2021]. However, these models often struggle to retain information over long contexts or face expanded memory requirements when handling extended past contexts. To address this, some approaches

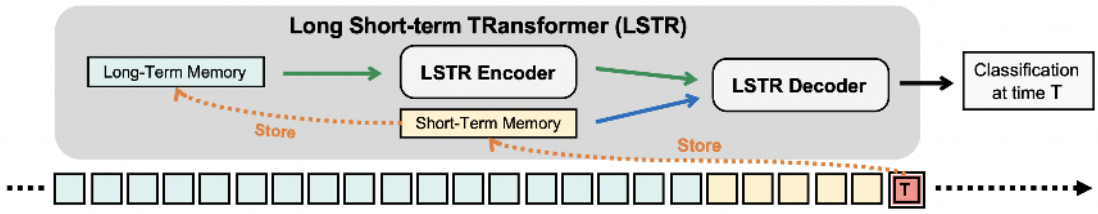

Figure 2.3: **LSTR** [Xu et al., 2021] maintains causal representations by combining short-term and long-term memory modules to store and update information in a streaming video setting.

introduce memory modules for more effective and efficient handling of complex tasks. For example, LSTR [Xu et al., 2021] separates past context as long-term and short-term memories for action detection (Figure 2.3), while XMem [Cheng and Schwing, 2022] incorporates a sensory memory module for fine-grained information in video object segmentation. Long-term memory-based modeling is also applied beyond video understanding [Balazevic et al., 2024], including tasks like long-sequence text processing and video question answering [Zhang et al., 2021b]. We also employ an attention-based memory mechanism, which is specialized in point tracking with two types of memory; one focusing on spatial local regions around points, and another on broader context.

Chapter 3

CAN VISUAL FOUNDATION MODELS ACHIEVE LONG-TERM POINT TRACKING?

Vision foundation models (FoMos) trained on large-scale data have demonstrated strong generalization across diverse visual tasks [Radford et al., 2021, Oquab et al., 2024, Rombach et al., 2022], yet their capacity to support long-term correspondence estimation remains underexplored. In this chapter, we investigate whether these models can be effectively repurposed for long-term point tracking. Our goal is to assess their geometric awareness and determine whether tracking can emerge through simple mechanisms like feature similarity or lightweight adaptation.

To this end, we formulate a point tracking pipeline that leverages correlation maps computed from FoMo features, and we evaluate models under three distinct regimes: (i) a *zero-shot* setting, where we use the frozen model to retrieve the most similar location based on cosine similarity [El Banani et al., 2024, Zhan et al., 2023]; (ii) a *probing* setup, where low-capacity convolutional heads are trained to decode point positions and occlusions from the correlation map [Doersch et al., 2022]; and (iii) an *adaptation* setup, where we apply Low-Rank Adaptation (LoRA)[Hu et al., 2022] to fine-tune attention layers.

3.1 Methodology

In this section, we first explain the models we consider (Section 3.1.1) and how we leverage them for point tracking under different settings (Section 3.1.2).

3.1.1 Models

We consider a range of visual models, each trained with different objectives on diverse datasets, inspired by previous work [El Banani et al., 2024]. Specifically, we

evaluate models trained with self-supervision: Masked Autoencoders (MAE) [He et al., 2022], DINO [Caron et al., 2021], DINOv2 [Oquab et al., 2024], and DINOv2 with registers (DINOv2-Reg) [Darcet et al., 2023]; language supervision: CLIP [Radford et al., 2021]; image generation: Stable Diffusion (SD) [Rombach et al., 2022]; and direct supervision for classification and segmentation respectively: DeIT III [Touvron et al., 2022] and Segment Anything (SAM) [Kirillov et al., 2023]. For each model, we use the available pre-trained checkpoints.

All models, except SD, utilize Vision Transformer (ViT) architectures, while SD employs a U-Net based model. For the ViTs, we use the Base architecture (ViT-B) with a patch size of either 14 (for the DINO family) or 16 (for the remaining ViTs), unless explicitly stated otherwise. We use the output of the last block as our feature map, discarding the `[CLS]` token.

For SD, we follow the approach outlined in DIFT [Tang et al., 2023] and use the outputs from an intermediate layer of the U-Net encoder. Specifically, we utilize the outputs of the upsample block at $n = 2$, after noising the input frame with a time step of $t = 51$, corresponding to $1/8^{\text{th}}$ of the input image resolution. To reduce the stochasticity of the noising process, we calculate the average feature over 8 runs, each with different noisy versions of the same input image.

3.1.2 FoMos for Point Tracking

In this section, we explain how we leverage foundation models to estimate correspondence. For all settings, we utilize correlation maps to achieve this. Formally, given an encoded video, *i.e.* feature maps, represented by a selected foundation model, $\mathbf{F} \in \mathbb{R}^{T \times D \times H \times W}$, and a query prompted at frame t_q at location $\mathbf{p}_q$, we extract the query feature $\mathbf{q} = \mathbf{F}_{t_q}(\mathbf{p}_q) \in \mathbb{R}^{D \times 1 \times 1}$ by bilinear interpolation. The correlation map at any time t, $\mathbf{C}_t \in \mathbb{R}^{H \times W}$, is then calculated using the cosine similarity between $\mathbf{F}_t$ and $\mathbf{q}$ (see Figure 3.1):

$$\mathbf{C}_t = \frac{\mathbf{q} \cdot \mathbf{F}_t}{\|\mathbf{q}\| \cdot \|\mathbf{F}_t\|} \tag{3.1}$$

As different models have varying stride values (8 for SD, 14 for the DINO family, and 16 for others), we ensure a fair evaluation by maintaining the same size of the

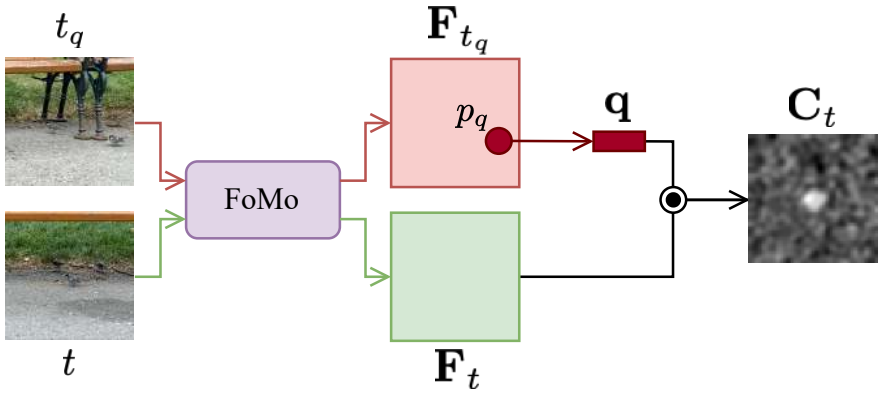

Figure 3.1: **Correlation Map.** We compute a dense correlation map $\mathbf{C}_t$ by measuring cosine similarity between a query feature $\mathbf{q}$ and all spatial locations in the frame feature map $\mathbf{F}_t$. The resulting map encodes the similarity distribution for a single point across the frame at time t. High similarity regions (highlighted in warmer colors) indicate likely positions for the tracked point. This formulation enables correspondence estimation without explicit supervision and serves as the foundation for both zero-shot retrieval and learning-based decoding.

correlation map, *i.e.* the resolution of the final feature map, across different models. This is achieved by resizing the input images so that the feature maps have a consistent 32×32 resolution.

Zero-Shot Evaluation: In the zero-shot setting, we directly consider the most similar feature location to the query as the prediction for the corresponding time step, t. In other words, the predicted location $\hat{\mathbf{p}}_t$ is the index of the highest value in $\mathbf{C}_t$:

$$\hat{\mathbf{p}}_t = \underset{\mathbf{p}}{\operatorname{argmax}}\, \mathbf{C}_t(\mathbf{p}) \tag{3.2}$$

However, this formulation does not provide information about the visibility of the point and is valid only if the point is visible. Therefore, we evaluate only the visible points.

Probing: In TAPNet [Doersch et al., 2022], correlation maps are computed from

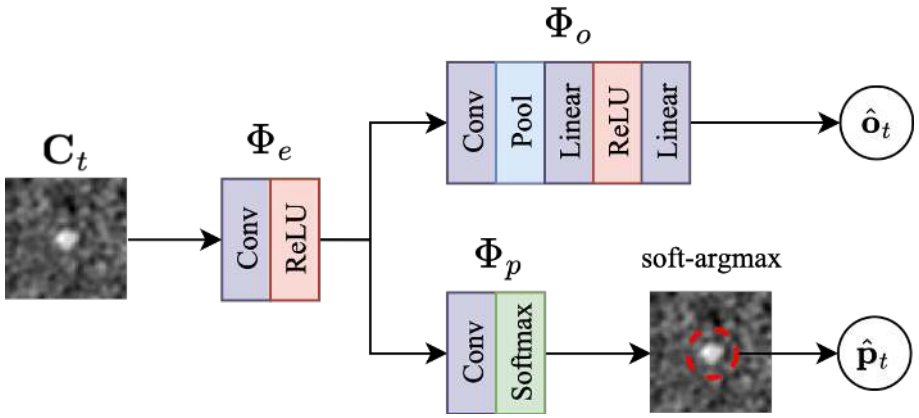

Figure 3.2: **Probing the Correlation Map.** We adopt lightweight convolutional heads, inspired by TAPNet [Doersch et al., 2022], to decode correlation maps into point predictions $\hat{\mathbf{p}}_t$ and occlusion logits $\hat{\mathbf{o}}_t$. A shared encoder Φ_e extracts compact features, which are processed by two specialized heads: one for occlusion classification, and one for localization via soft-argmax.

backbone features and processed through a small network to decode visibility and point location. The simplicity of this setup makes it ideal for probing whether geometric correspondence is inherently encoded in a frozen feature space.

Inspired by this, we attach low-capacity convolutional branches—comprising only 5.5K parameters—on top of the correlation map $\mathbf{C}_t$, computed from a frozen foundation model. Formally, the point prediction $\hat{\mathbf{p}}_t$ and occlusion probability $\hat{\mathbf{o}}_t$ are computed as:

$$\begin{aligned} \hat{\mathbf{p}}_t &= \text{soft-argmax}(\Phi_p \circ \Phi_e(\mathbf{C}_t)) \\ \hat{\mathbf{o}}_t &= \sigma(\Phi_o \circ \Phi_e(\mathbf{C}_t)) \end{aligned} \tag{3.3}$$

Here, Φ_e denotes a shared convolutional layer applied to the correlation map. The outputs are then passed to two task-specific heads: Φ_p for localizing the target point, and Φ_o for occlusion classification. The point head uses a convolution followed by a spatial softmax and a soft-argmax to produce a 2D coordinate, while the occlusion head includes a convolution, pooling, and two linear layers with ReLU activation to predict an occlusion logit. We illustrate this architecture in Figure 3.2.

The overall design ensures minimal overhead while allowing the model to decode

geometric signals, if present, from frozen backbone features. The training objectives include a Huber loss for point localization and a binary cross-entropy loss for occlusion.

Adaptation: In the adaptation phase, we go beyond mere probing and update the model to examine whether these models provide a good starting point for learning correspondence in a constrained learning setup. Rather than fine-tuning the entire model, we employ Low Rank Adaptation (LoRA) [Hu et al., 2022]. Formally, given a pretrained weight $\mathbf{W}_{\{Q,V\}} \in \mathbb{R}^{d\times d}$, we use the adapted weight $\mathbf{W}'$:

$$\mathbf{W}'_{\{Q,V\}} = \mathbf{W}_{\{Q,V\}} + \mathbf{BA} \tag{3.4}$$

where $\mathbf{A} \in \mathbb{R}^{r\times d}$, $\mathbf{B} \in \mathbb{R}^{d\times r}$, r is the rank, and d is the feature dimension. We train only the $\mathbf{B}$ and $\mathbf{A}$ matrices while keeping the original $\mathbf{W}$ matrices frozen, for each attention layer.

This approach maintains the integrity of the pretrained model's information while enabling adaptation with significantly fewer parameters compared to full model fine-tuning. Specifically, we apply LoRA to the query and value projections within the attention layers of the Vision Transformer (ViT), following the methodology in MeLo [Zhu et al., 2023].

3.2 Experiments

3.2.1 Experimental Setup

Datasets: We evaluate the models on the TAP-Vid benchmark, utilizing three datasets: (i) **TAP-Vid DAVIS** includes 30 real-world videos, each with around 100 frames, featuring intricate motions; (ii) **TAP-Vid RGB-Stacking** is a synthetic dataset containing 50 videos of robotic manipulation tasks; (iii) **TAPVid-Kinetics** consists of over 1000 online videos with various actions. For training in probing and adaptation, we use **TAP-Vid Kubric**, a synthetic simulation dataset of 11K videos, each with a fixed length of 24 frames.

Metrics: Consistent with prior work [Doersch et al., 2023, Doersch et al., 2024, Karaev et al., 2024b], we evaluate tracking performance using multiple metrics.

Table 3.1: **Zero-Shot Evaluation.** These results show the zero-shot evaluation results on the TAP-Vid datasets. δ^x_{avg} is reported.

Model	DAVIS	RGB-St.	Kinetics	**Avg.**
MAE [He et al., 2022]	23.5	43.2	27.6	31.4
DeIT III [Touvron et al., 2022]	24.0	23.3	22.4	23.2
CLIP [Radford et al., 2021]	25.4	33.8	25.0	28.1
SAM [Kirillov et al., 2023]	29.5	<u>44.7</u>	31.4	35.2
SD [Rombach et al., 2022]	33.9	**46.2**	**37.2**	**39.1**
DINO [Caron et al., 2021]	34.5	39.3	34.4	36.1
DINOv2-Reg [Darcet et al., 2023]	<u>37.4</u>	35.9	33.6	35.6
DINOv2 [Oquab et al., 2024]	**38.0**	37.8	<u>34.5</u>	<u>36.8</u>
TAPNet [Doersch et al., 2022]	48.6	68.1	54.4	57.0

Occlusion Accuracy (OA) evaluates the correctness of occlusion predictions. δ^x_{avg} indicates the proportion of visible points tracked within 1, 2, 4, 8, and 16 pixels, averaged across them. Average Jaccard (AJ) combines both occlusion and prediction accuracy. Evaluations are performed in a queried-first mode, where the first visible point for each trajectory serves as the query.

3.2.2 Zero-Shot Evaluation

Different Models: The results of different foundation models in the zero-shot setting are presented in Table 3.1. We also report the performance of a supervised model, TAPNet [Doersch et al., 2022], as it is the only supervised model with a final feature map resolution of 32×32. The DINO family demonstrates superior performance on DAVIS. Conversely, SD and SAM outperform the DINO models on RGB-Stacking. One possible reason for this is that RGB-Stacking is a synthetic dataset, giving SD an advantage due to its more diverse training set, which includes both real-world and synthetic data. Additionally, SD performs the best on Kinetics.

On average, SD achieves the highest performance, followed by DINOv2. We visualize correlation maps of SD, DINOv2, DINOv2-Reg, and SAM in Figure 3.3. SD appears to have greater geometric sensitivity, while DINOv2 demonstrates higher semantic capability. Overall, the superior performance of these two models aligns with prior work [El Banani et al., 2024, Zhan et al., 2023], which shows that DINOv2 and SD have a better 3D understanding compared to other models.

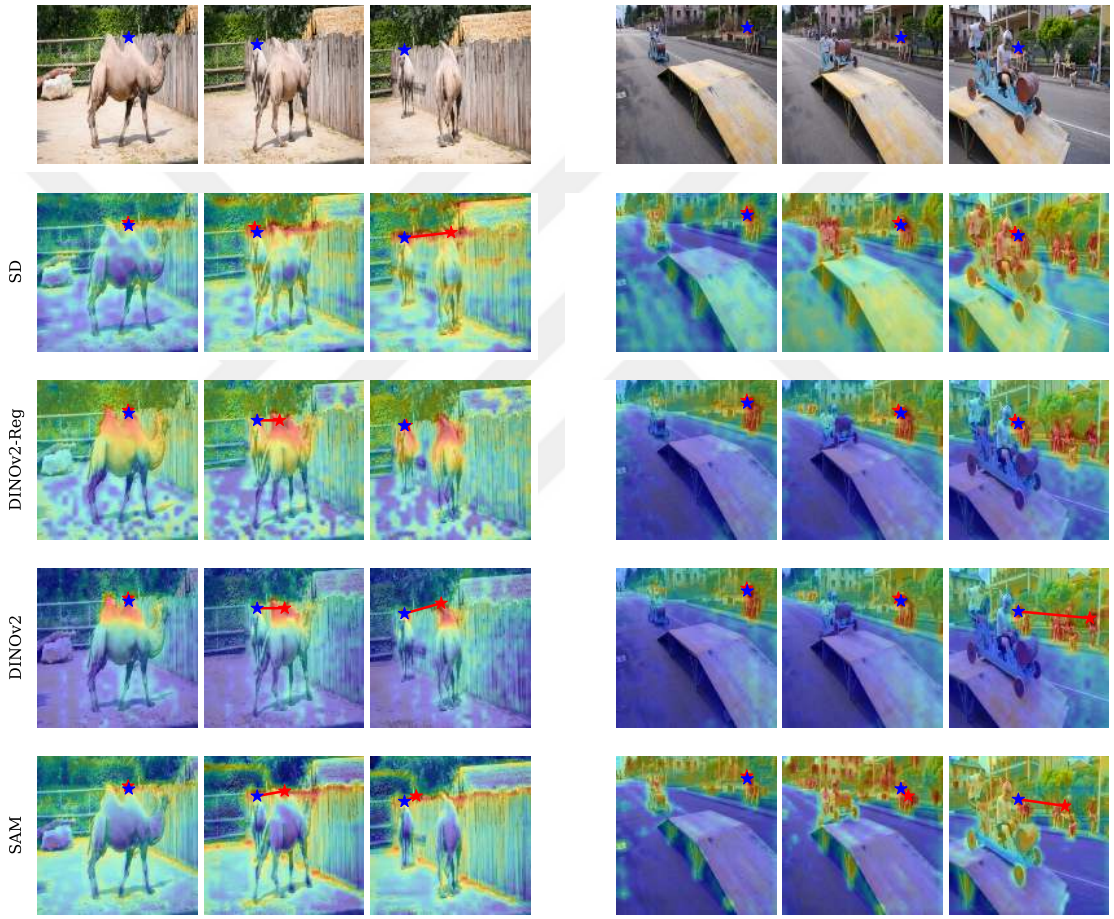

Figure 3.3: **Zero-Shot Tracking on TAP-Vid DAVIS.** We show correlation maps $\mathbf{C}_t$ for query points from two videos using Stable Diffusion [Rombach et al., 2022], DINOv2 [Oquab et al., 2024], DINOv2-Reg [Darcet et al., 2023], and SAM [Kirillov et al., 2023]. We compute correlation maps between a sampled query feature and frame features at later timesteps. Predictions (red stars) and ground truth (blue stars) are connected by red lines to indicate tracking error. Warmer colors in the correlation maps indicate stronger feature similarity.

Effect of Architectural Variations: We conducted an ablation study on: (i) architectural differences to understand how the backbone capacity of the same pre-training method affects correspondence awareness, and (ii) input resolution to examine the effect of resolution, a key factor in correspondence [Doersch et al., 2023], in Table 3.2 using DINOv2. The results indicate that using larger architectures consistently improves performance, with an increase from 37.1 (ViT-S) to 40.0 (ViT-G). Additionally, the resolution of the final feature map has a significant impact. Higher resolutions lead to a performance gain of 10.5 (64×64), while lower resolutions result in a reduction of 13.1 (16×16).

Table 3.2: **Effect of Architecture for Zero-Shot Evaluation.** This table shows the zero-shot evaluation results for different architectures of DINOv2 by varying the resolution on TAP-Vid DAVIS.

Architecture	**Final Resolution**	δ^x_{avg}
ViT-**S**/14	32×32	37.1 (-0.9)
ViT-**B**/14		38.0
ViT-**L**/14		39.1 (+1.1)
ViT-**G**/14		40.0 (+2.0)
ViT-**B**/14	16×16	24.9 (-13.1)
	48×48	45.1 (+7.1)
	64×64	48.5 (+10.5)

3.2.3 Probing and Adaptation

We chose DINOv2 for probing and adaptation due to its top-2 performance in zero-shot evaluation and its resource efficiency compared to SD. For computational efficiency, we selected the ViT-S/14 architecture. For both settings, we used AdamW [Loshchilov and Hutter, 2019], a batch size of 16 (1/4 of TAPNet), a learning rate of 1×10^{-3}, linear warm-up, and cosine decay. We trained for 20 epochs

Table 3.3: **Probing and Adapting DINOv2.** This table shows different setups for DINOv2 ViT-S/14 on the TAP-Vid DAVIS, including the number of learnable parameters. The setups include zero-shot, probing, and adaptation with various LoRA ranks.

Model	**Setup**	**Rank**	**#L.P.**	AJ	δ^x_{avg}	OA
DINOv2	Zero-shot	-	0	-	37.1	-
	Probing	-	5.5K	27.1	42.3	79.4
	Adaptation	16	0.3M	33.4	49.0	80.1
	Adaptation	32	0.6M	33.9	49.7	**80.4**
	Adaptation	64	1.2M	**35.0**	**51.3**	80.2
TAPNet	Supervised	-	12.0M	33.0	48.6	78.8

(~1/4 iterations of TAPNet) for probing and 40 epochs (~1/2 iterations of TAPNet) for adaptation. The applied weight decays were 1×10^{-3} for probing and 1×10^{-5} for adaptation.

We compared probing and adaptation results with different ranks, as shown in Table 3.3. By only probing correlation maps, DINOv2 can surpass TAPNet in OA. Moreover, adaptation of any rank performs better than TAPNet across all metrics. Surpassing supervised models even in a significantly constrained training setup, where the number of learnable parameters is substantially lower (2.5% for rank 16), demonstrates that DINOv2 could serve as a strong initialization for correspondence learning.

3.2.4 Discussion

We explored the geometric awareness of vision foundation models for long-term point tracking under zero-shot settings and showed that Stable Diffusion and DINOv2 have better geometric understanding than other models. Moreover, our experiments demonstrate that DINOv2 can surpass the performance of supervised models despite

a significantly lighter training setup, indicating that these models inherently possess the notion of correspondence.

Chapter 4

TRACK-ON: TRANSFORMER-BASED ONLINE POINT TRACKING WITH MEMORY

In Chapter 3, we showed that visual foundation models encode strong geometric priors and can be repurposed for point tracking via simple correlation-based formulations. However, despite their success in zero-shot and lightly supervised settings, these models lack temporal reasoning and operate without memory, limiting their performance in online tracking scenarios. More broadly, existing point tracking methods, such as TAPIR [Doersch et al., 2023] and CoTracker [Karaev et al., 2024b], are designed for offline use. They process multiple frames jointly, often relying on future context, making them unsuitable for streaming video where predictions must be made sequentially.

While FoMos can be integrated into these models to improve accuracy, this does not resolve the fundamental challenge: tracking in an online setting. Even with strong features, existing models struggle to maintain consistency over time without access to future frames or full spatiotemporal attention. This motivates the need for an architecture explicitly designed for online long-term tracking.

In this chapter, we introduce **Track-On**, a transformer-based model that tracks points causally, frame by frame. Each point is represented as a query in the decoder, and updated through two specialized memory modules. The spatial memory stores localized features around past predictions to reduce drift, while the context memory aggregates point-wise embeddings over time to maintain continuity. This design enables accurate and efficient long-term tracking under strict online constraints.

4.1 Methodology

4.1.1 Problem Scenario

Given an RGB video of T frames, $\mathcal{V} = \{\mathbf{I}_1,\ \mathbf{I}_2,\ \ldots,\ \mathbf{I}_T\} \in \mathbb{R}^{T\times H\times W\times 3}$, and a set of N predefined queries, $\mathcal{Q} = \{(t^1, \mathbf{p}^1),\ (t^2, \mathbf{p}^2),\ \ldots,\ \ (t^N, \mathbf{p}^N)\} \in \mathbb{R}^{N\times 3}$, where each query point is specified by the start time and pixel's spatial location, our goal is to predict the correspondences $\hat{\mathbf{p}}_t \in \mathbb{R}^{N\times 2}$ and visibility $\hat{\mathbf{v}}_t \in \{0,1\}^N$ for all query points in an online manner, *i.e.* using only frames up to the current target frame t. To address this problem, we propose a transformer-based point tracking model, that tracks points **frame-by-frame**, with dynamic memories $\mathbf{M}$ to propagate temporal information along the video sequence:

$$\hat{\mathbf{p}}_t,\ \hat{\mathbf{v}}_t,\ \mathbf{M}_t = \Phi\left(\mathbf{I}_t,\ \mathcal{Q},\ \mathbf{M}_{t-1};\ \Theta\right) \tag{4.1}$$

In the following sections, we start by describing the basic transformer architecture for point tracking in Section 4.1.2, then introduce the two memory modules and their update mechanisms in Section 4.1.3.

4.1.2 Track-On: Point Tracking with a Transformer

Our model is based on transformer, consisting of three components, as illustrated in Figure 4.1: **Visual Encoder** is tasked to extract visual features of the video frame, and initialize the query points; **Query Decoder** enables the queried points to attend the target frame to update their features; and **Point Prediction**, to predict the positions of corresponding queried points in a coarse-to-fine manner.

Visual Encoder

We adopt a Vision Transformer (ViT) as our visual backbone, specifically DINOv2 [Oquab et al., 2024], due to its strong viewpoint invariance and adaptation flexibility, as demonstrated in Chapter 3. However, standard ViTs process image patches with relatively large stride, which limits spatial resolution and poses challenges for dense prediction tasks. To address this, we employ ViT-Adapter [Chen

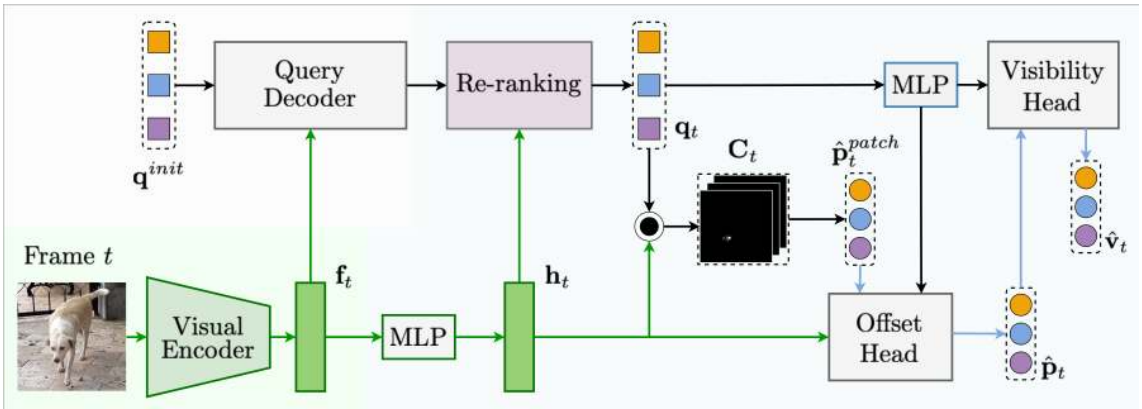

Figure 4.1: **Overview.** We introduce Track-On, a simple transformer-based method for online, frame-by-frame point tracking. The process involves three steps: (i) **Visual Encoder**, which extracts features from the given frame; (ii) **Query Decoder**, which decodes interest point queries using the frame's features; (iii) **Point Prediction** (highlighted in light blue), where correspondences are estimated in a coarse-to-fine manner, first through patch classification based on similarity, then followed by refinement through offset prediction from a few most likely patches. Note that the squares refer to point queries, while the circles represent predictions, either as point coordinates or visibility.

et al., 2022b] to obtain higher-resolution feature maps. Finally, we add learnable spatial positional embeddings γ^s to the per-frame features:

$$\mathbf{f}_t = \Phi_{\text{vis-enc}}(\mathbf{I}_t) + \gamma^s \in \mathbb{R}^{\frac{H}{S} \times \frac{W}{S} \times D} \tag{4.2}$$

where D denotes the feature dimension, and S refers to the stride. We use a single-scale feature map from ViT-Adapter for memory efficiency, specifically with a stride of $S = 4$.

Query Initialization: To initialize the query features ($\mathbf{q}^{init}$), we apply bilinear sampling to the feature map at the query position ($\mathbf{p}^i$):

$$\mathbf{q}^{init} = \left\{\text{sample}(\mathbf{f}_{t^i},\ \mathbf{p}^i)\right\}_{i=1}^{N} \in \mathbb{R}^{N \times D} \tag{4.3}$$

In practice, we initialize the query based on the features of the start frame t^i for i-th query, assuming they can start from different time point, and propagate them

to the subsequent frames.

Query Decoder

After extracting the visual features for the frame and query points, we adopt a variant of transformer decoder [Vaswani et al., 2017], with 3 blocks, *i.e.* cross-attention followed by a self-attention, with an additional feed forward layer between attentions:

$$\mathbf{q}_t^{dec} = \Phi_{\text{q-dec}}\left(\mathbf{q}^{init},\ \mathbf{f}_t\right) \in \mathbb{R}^{N\times D} \tag{4.4}$$

The points of interest are treated as queries, which update their features by iteratively attending to visual features of the current frame with cross attention. These updated queries are then used to search for the best match within the current frame, as explained in the following section.

Point Prediction

Unlike previous work that regresses the exact location of the points, we formulate the tracking as a matching problem to one of the patches, that provides a coarse estimate of the correspondence. For exact correspondence with higher precision, we further predict offsets to the patch center. Additionally, we also infer the visibility $\hat{\mathbf{v}}_t \in [0,1]^N$ and uncertainty $\hat{\mathbf{u}}_t \in [0,1]^N$ for the points of interest.

Patch Classification: We first pass the visual features into 4-layer MLPs, and downsample the resulting features into multiple scales, *i.e.*, $\mathbf{f}_t \rightarrow \mathbf{h}_t \in \mathbb{R}^{\frac{H}{S}\times\frac{W}{S}\times D} \rightarrow \mathbf{h}_t^l \in \mathbb{R}^{\frac{H}{2^l.S}\times\frac{W}{2^l.S}\times D}$. We compute the cosine similarity between the decoded queries and patch embeddings in four scales, and the similarity map $\mathbf{C}_t^{dec}$ is obtained as the weighted average of multi-scale similarity maps with learned coefficients. We then apply a temperature to scale the similarity map and take softmax spatially over the patches within the current frame. The resulting $\mathbf{C}_t^{dec}$ provides a measure of similarity for each query across the patches in the frame.

We train the model with a classification objective, where the ground-truth class is the patch with the point of interest in it. In other words, we perform a P-class classification, P is the total number of patches in the frame.

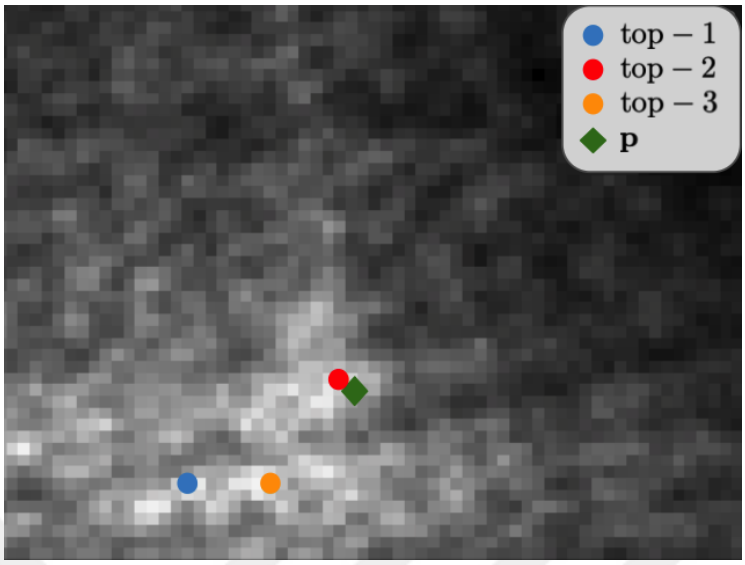

Figure 4.2: **Top-k Points.** In certain cases, a patch with high similarity, though not the most similar, is closer to the ground-truth patch. The top-3 patch centers, ranked by similarity, are marked with dots, while the ground-truth is represented by a diamond.

Re-ranking: We observed that the true target patch might not always have the highest similarity on $\mathbf{C}_t^{dec}$, however, it is usually among the top-k patches. For example, in Figure 4.2, the patch with the second-highest similarity (top-2) is closer to the true correspondence than the most similar patch (top-1). To rectify such cases, we introduce a re-ranking module $\Phi_{\text{re-rank}}$:

$$\mathbf{q}_t = \Phi_{\text{re-rank}}\left(\mathbf{q}_t^{dec},\ \mathbf{h}_t,\ \mathbf{C}_t^{dec}\right) \in \mathbb{R}^{N\times D} \tag{4.5}$$

where $\mathbf{q}_t$ denotes the refined queries after ranking.

In the re-ranking module (Figure 4.3), we identify the top-k patches with the highest similarities and retrieve their corresponding features with a deformable attention decoder. Then, we integrate them into the original query features via a transformer decoder to produce refined queries. Using these refined queries, we calculate the final similarity map $\mathbf{C}_t$ and apply a classification loss. Finally, we select the center of the patch with the highest similarity ($\hat{\mathbf{p}}_t^{patch} \in \mathbb{R}^{N\times 2}$) as our coarse prediction. Additionally, we compute an uncertainty score for each top-k location,

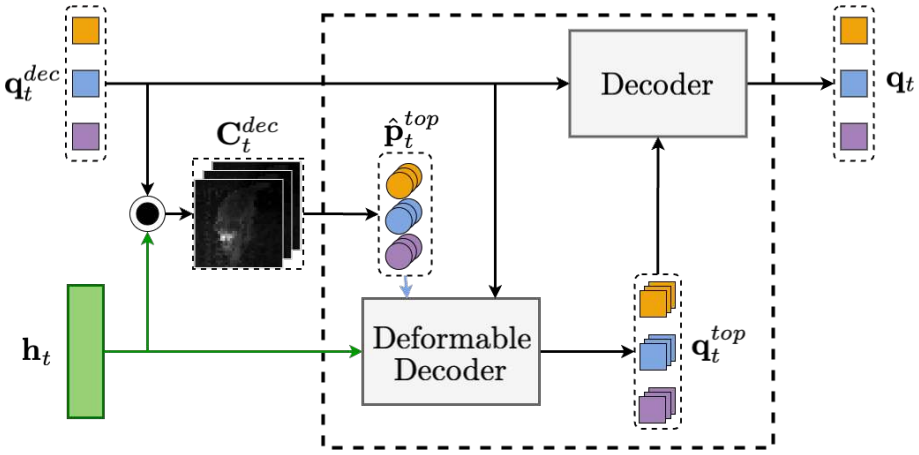

Figure 4.3: **Re-ranking Module.** The features around the top-k points ($\hat{\mathbf{p}}_t^{top}$) with the highest similarity are decoded using deformable attention to extract the corresponding top-k features ($\mathbf{q}_t^{top}$). These features are then fused with the decoded query $\mathbf{q}_t^{dec}$ using a transformer decoder.

i.e. $\hat{\mathbf{u}}_t^{top} \in \mathbb{R}^{N\times k}$, by processing their corresponding features with a linear layer.

Offset Prediction: For the exact correspondence ($\hat{\mathbf{p}}_t \in \mathbb{R}^{N\times 2}$), we further predict an offset $\hat{\mathbf{o}}_t \in \mathbb{R}^{N\times 2}$ to the patch center by incorporating features from the local region around the inferred patch, as shown in Figure 4.4:

$$\hat{\mathbf{o}}_t = \Phi_{\text{off}}(\mathbf{q}_t,\ \mathbf{h}_t,\ \hat{\mathbf{p}}_t^{patch}), \qquad \hat{\mathbf{p}}_t = \hat{\mathbf{p}}_t^{patch} + \hat{\mathbf{o}}_t \tag{4.6}$$

Here, Φ_{off} is a deformable transformer decoder [Zhu et al., 2021] block with 3 layers, excluding self-attention. In this decoder, the query $\mathbf{q}_t$ is processed using the key-value pairs $\mathbf{h}_t$, with the reference point set to $\hat{\mathbf{p}}_t^{patch}$. To limit the refinement to the local region, the offsets are constrained by the S (stride) and mapped to the range $[-S, S]$ using a tanh activation.

In addition, we predict the visibility $\hat{\mathbf{v}}_t$ and uncertainty $\hat{\mathbf{u}}_t$, using visibility head Φ_{vis}. We first decode the region around the predicted location $\hat{\mathbf{p}}_t$ (Eq. 4.6) using a deformable decoder layer. Then, we predict visibility and uncertainty by applying a linear layer to the decoded queries. At training time, we define a prediction to be uncertain if the prediction error exceeds a threshold ($\delta_u = 8$ pixels) or if the point is occluded. During inference, we classify a point as visible if its probability exceeds

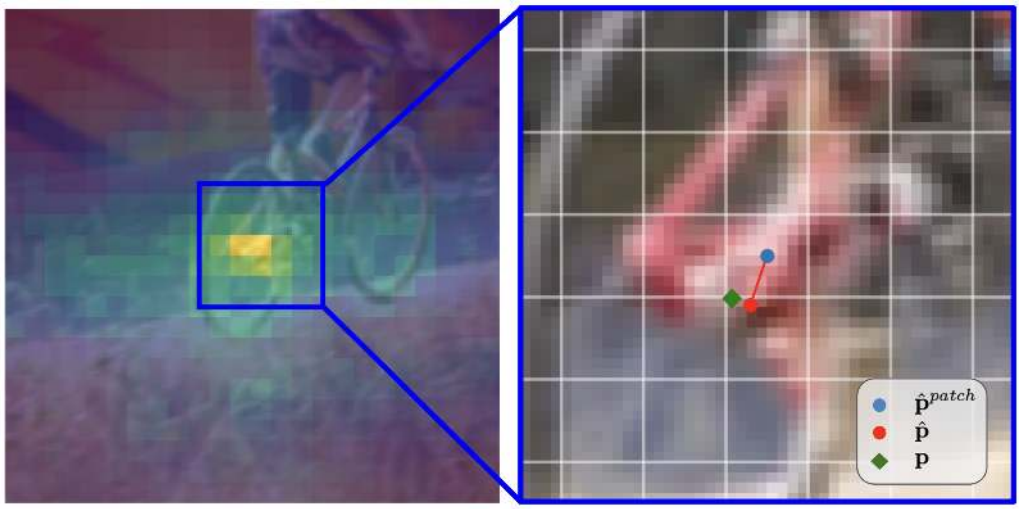

Figure 4.4: **Offset Head**. Starting with a rough estimation from patch classification (**left**), where lighter colors indicate higher correlation, we refine the prediction using the offset head (**right**). The selected patch center and the final prediction are marked by a blue dot and a red dot, respectively, with the ground-truth represented by a diamond.

a threshold δ_v. Although we do not directly utilize uncertainty in our predictions during inference, we found predicting uncertainty to be beneficial for training.

Training: We train our model using the ground-truth trajectories $\mathbf{p}_t \in \mathbb{R}^{N\times 2}$ and visibility information $\mathbf{v}_t \in \{0,1\}^N$. For patch classification, we apply cross-entropy loss based on the ground-truth class, patch $\mathbf{c}^{patch}$. For offset prediction $\hat{\mathbf{o}}_t$, we minimize the ℓ_1 distance between the predicted offset and the actual offset. We supervise the visibility $\hat{\mathbf{v}}_t$ and uncertainty $\hat{\mathbf{u}}_t$ using binary cross-entropy loss. Additionally, we supervise the uncertainties of the top-k points, $\hat{\mathbf{u}}_t^{top}$, at re-ranking. The total loss is a weighted combination of them:

$$\begin{aligned}\mathcal{L} = \lambda \underbrace{\left(\mathcal{L}_{\text{CE}}\left(\mathbf{C}_t,\ \mathbf{c}^{patch}\right) + \mathcal{L}_{\text{CE}}\left(\mathbf{C}_t^{dec},\ \mathbf{c}^{patch}\right)\right)}_{\text{Patch Classification Loss}} \cdot \mathbf{v}_t \\ + \underbrace{\mathcal{L}_{\ell_1}\left(\hat{\mathbf{o}}_t,\ \mathbf{o}_t\right)}_{\text{Offset Loss}} \cdot \mathbf{v}_t + \underbrace{\mathcal{L}_{\text{CE}}(\hat{\mathbf{v}}_t, \mathbf{v}_t)}_{\text{Visibility Loss}} + \underbrace{\mathcal{L}_{\text{CE}}(\hat{\mathbf{u}}_t, \mathbf{u}_t)}_{\text{Uncertainty Loss}} + \underbrace{\mathcal{L}_{\text{CE}}(\hat{\mathbf{u}}_t^{top}, \mathbf{u}_t^{top})}_{\text{Top-}k\text{ Uncertainty Loss}}\end{aligned} \tag{4.7}$$

Discussion: Till this point, our model has exclusively considered relocating the queried points within the current frame. However, as the appearance of points

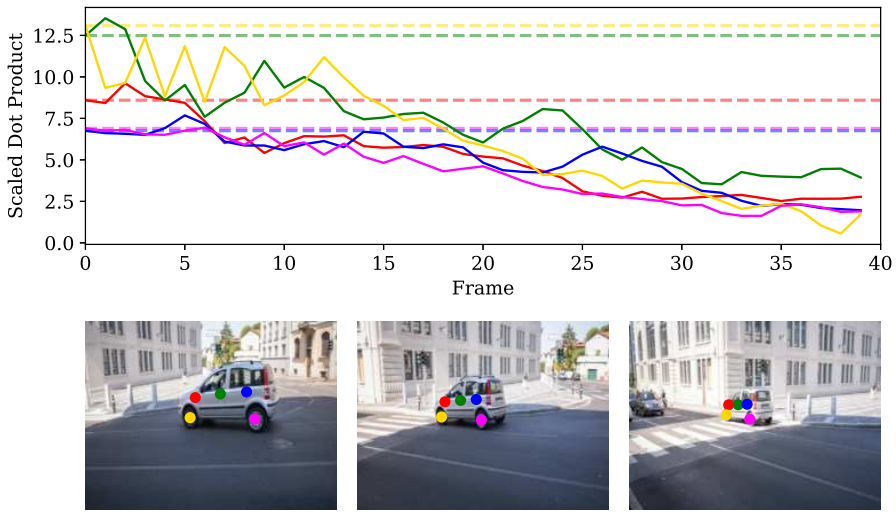

Figure 4.5: **Feature Drift.** For the tracks shown below (start, middle, and final frames), the plot above illustrates the decreasing similarity between the features of the initial query and its correspondences over time, with the initial similarity indicated by horizontal dashed lines.

consequently changes over time, the embedding similarity between the initial query point and future correspondences tends to decrease gradually (Figure 4.5). This problem, known as feature drift, leads to inaccurate predictions, when solely relying on the feature similarity with the initial point.

4.1.3 Track-On with Memory

Here, we introduce two types of memories: **spatial memory** and **context memory**, as illustrated in Figure 4.6. Spatial memory stores information around the predicted locations, allowing us to update the initial queries based on the latest predictions. Context memory preserves the track's history states by storing previously decoded queries, ensuring continuity over time and preventing inconsistencies. This design enables our model to effectively capture temporal progressions in long-term videos, while also adapting to changes in the target's features to address feature drift.

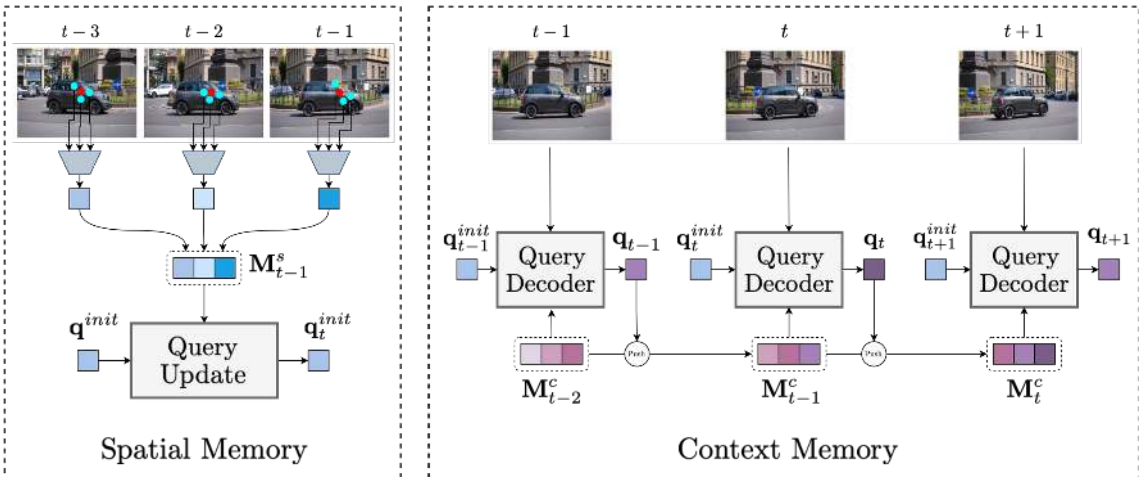

Figure 4.6: **Memory Modules.** Spatial memory $\mathbf{M}^s_{t-1}$ (**left**) is used to update the initial query $\mathbf{q}^{init}$ from the first frame to $\mathbf{q}^{init}_t$ on the current frame. The goal is to resolve feature drift by storing the content around the model's predictions in previous frames. Context memory $\mathbf{M}^c_{t-1}$ (**right**) is input to the query decoder which updates $\mathbf{q}^{init}_t$ to $\mathbf{q}_t$. It provides a broader view of the track's history with appearance changes and occlusion status by storing the point's embeddings from past frames.

We store the past features for each of the N queries independently, with up to K embeddings per query in each memory module. Once fully filled, the earliest entry from the memory will be obsoleted as a new entry arrives, operating as a First-In First-Out (FIFO) queue.

Spatial Memory

Here, we introduce the spatial memory module that stores fine-grained local information from previous frames, enabling continuous updates to the initial query points. This adaptation to appearance changes helps mitigate feature drift.

Memory Construction: We zero-initialize the memory, $\mathbf{M}^s_0$, update its content with each frame. For the first frame, we make a prediction using initial query $\mathbf{q}^{init}$ without memory.

Memory Write ($\Phi_{\text{q-wr}}$): To update the memory with the new prediction, $\mathbf{M}^s_{t-1} \rightarrow \mathbf{M}^s_t$, we extract a feature vector around the predicted point $\hat{\mathbf{p}}_t$ on the current feature

map $\mathbf{f}_t$, and add it to the memory:

$$\mathbf{M}_t^s = [\mathbf{M}_{t-1}^s,\ \Phi_{\text{q-wr}}\left([\mathbf{q}^{init}, \mathbf{q}_t],\ \mathbf{f}_t,\ \hat{\mathbf{p}}_t\right)] \tag{4.8}$$

$\Phi_{\text{q-wr}}$ is a 3-layer deformable transformer decoder without self-attention, using the concatenated $\mathbf{q}^{init}$ and $\mathbf{q}_t$ as the query, attending a local neighborhood of predicted point for update. Utilizing deformable attention for the local summarization process helps prevent error propagation over time, as the query can flexibly select relevant features from any range.

Query Update ($\Phi_{\text{q-up}}$): In such scenario, before passing into the query decoder to estimate the correspondence, the initial query points first visit the spatial memory $\mathbf{M}_{t-1}^s$ for an update:

$$\mathbf{q}_t^{init} = \Phi_{\text{q-up}}\left(\mathbf{q}^{init},\ \mathbf{M}_{t-1}^s\right) = \mathbf{q}^{init} + \phi_{\text{qqm}}\left(\mathbf{q}^{init},\ \phi_{\text{mm}}(\mathbf{M}_{t-1}^s + \gamma^s)\right) \tag{4.9}$$

ϕ_{mm} is a transformer encoder layer that captures dependencies within the memory, and ϕ_{qqm} is a transformer decoder layer without initial self-attention, where $\mathbf{q}^{init}$ attends to updated memory, followed by a linear layer, and $\gamma^s \in \mathbb{R}^{K\times D}$ is learnable position embeddings. Instead of sequentially updating the query embeddings at each time step, *e.g.* extracting $\mathbf{q}_t^{init}$ using $\mathbf{q}_{t-1}^{init}$, we update them with respect to the initial query $\mathbf{q}^{init}$, conditioned on all previous predictions stored in the memory. This prevents error propagation by taking into account the entire history of predictions.

Context Memory

In addition to spatial memory, we introduce a context memory that incorporates historical information of the queried points from a broader context, enabling the model to capture past occlusions and visual changes. Specifically, we store the decoded query features from previous time steps in context memory, $\mathbf{M}_{t-1}^c$. We then integrate it by extending the query decoder with an additional transformer decoder layer without self-attention, where queries attend to memory with added learnable position embeddings ($\gamma^c \in \mathbb{R}^{K\times D}$):

$$\mathbf{q}_t = \Phi_{\text{q-dec}}\left(\mathbf{q}_t^{init},\ \mathbf{f}_t,\ \mathbf{M}_{t-1}^c + \gamma^c\right) \tag{4.10}$$

Changes to the query decoder with memory are shown in red. For the writing operation, we add the most recent $\mathbf{q}_t$ to $\mathbf{M}^c_{t-1}$ and remove the oldest item, following the same procedure as in the spatial memory. Our experiments demonstrate that incorporating past content temporally with context memory enables more consistent tracking with additional benefits over spatial memory, especially in visibility prediction, since spatial memory focuses only on the regional content where the point is currently visible.

Inference-Time Memory Extension

Although the memory size K is fixed at training time, the number of video frames at inference can be different from the training frame limit. To address this, we extend the memory size during inference by linearly interpolating the temporal positional embeddings, γ^s and γ^c, to a larger size K_i. In particular, we train our model with memory size $K = 12$, and extend it to $K_i \in \{16, \ldots, 96\}$ at inference time.

4.2 Experiments

4.2.1 Experimental Setup

Datasets: We evaluate our model on seven datasets with diverse characteristics. For both training and evaluation, we use TAP-Vid [Doersch et al., 2022], consistent with prior work. Specifically, we train on TAP-Vid Kubric, a synthetic dataset containing 11K video sequences, each with 24 frames. For evaluation, we use three subsets from the TAP-Vid benchmark: **TAP-Vid DAVIS**, consisting of 30 real-world videos from the DAVIS dataset; **TAP-Vid RGB-Stacking**, a synthetic dataset of 50 videos focused on robotic manipulation with mostly textureless objects; **TAP-Vid Kinetics**, which includes over 1,000 real-world videos from the Kinetics dataset.Beyond TAP-Vid, we evaluate generalization to the following datasets: **RoboTAP** [Vecerik et al., 2023], consisting of 265 real-world robotic sequences, each averaging over 250 frames; **Dynamic Replica** [Karaev et al., 2023], a benchmark for 3D reconstruction with 20 sequences of 300 frames; **BADJA** [Biggs et al., 2019], a dataset for animal joint tracking, containing 7 sequences with sparse trajectories;

Point Odyssey (PO) [Zheng et al., 2023], which contains 12 long-form videos of up to 4325 frames, designed to test extreme tracking persistence.

Evaluation Details: We follow the standard protocol of TAP-Vid benchmark by first downsampling the videos to 256×256. We evaluate models in the queried first protocol, which is the natural setting for causal tracking. In this mode, the first visible point in each trajectory serves as the query, and the goal is to track that point in subsequent frames.

4.2.2 Results

TAP-Vid Benchmark

As shown in Table 4.1, we categorize models into online and offline settings. Offline models, with bidirectional information flow, use either a fixed-size window—where half of the window spans past frames and the other half future frames—or the entire video, granting access to any frame regardless of video length and providing a clear advantage. In contrast, online models process one frame at a time, enabling frame-by-frame inference. In the following discussion, we mainly focus on the setting using similar training set to ours, *i.e.* the models without using real-world videos.

We evaluate tracking performance with the following metrics of TAP-Vid benchmark, as in Chapter 3: Occlusion Accuracy (OA), which measures the accuracy of visibility prediction; δ^x_{avg}, the average proportion of visible points tracked within 1, 2, 4, 8, and 16 pixels; Average Jaccard (AJ), which jointly assesses visibility and localization precision.

Comparison on DAVIS: Our model outperforms all existing online models across every evaluation metric, achieving an 8.3 AJ improvement over the closest competitor, Online TAPIR. Additionally, it surpasses all offline models in both AJ (65.0 vs. 64.5) and δ^x_{avg}(78.0 vs. 76.7), outperforming even the concurrent CoTracker3, which was trained on longer videos (24 vs. 64 frames). Notably, our model also outperforms models fine-tuned on real-world videos by a significant margin. These results are particularly impressive because our model is an online approach, processing the video frame by frame, yet it exceeds the performance of offline models that process

Table 4.1: **Quantitative Results on TAP-Vid Benchmark.** This table shows results in comparison to the previous work on TAP-Vid under queried first setting, in terms of AJ, δ^x_{avg}, and OA. The models are categorized into online and offline schemes, the former setting grants access to any frame regardless of video length, thus providing a clear advantage. While online models process one frame at a time, enable frame-by-frame inference. For training datasets, Kub and Kub-L(ong), refer to the TAP-Vid Kubric dataset with 24-frame and 64-frame videos, respectively; and R indicates the inclusion of a large number of real-world videos, we highlight these models in gray. MFT is a long-term optical flow method trained on a combination of Sintel [Butler et al., 2012], FlyingThings [Mayer et al., 2016], and Kubric datasets.

Model	Input	Train	DAVIS			RGB-Stacking			Kinetics		
			AJ ↑	δ^x_{avg}↑	OA ↑	AJ ↑	δ^x_{avg}↑	OA ↑	AJ ↑	δ^x_{avg}↑	OA ↑
Offline											
TAPIR	Video	Kub	56.2	70.0	86.5	55.5	69.7	88.0	49.6	64.2	85.0
TAPTR	Window	Kub	63.0	76.1	<u>91.1</u>	60.8	76.2	87.0	49.0	64.4	85.2
TAPTRv2	Window	Kub	63.5	75.9	**91.4**	53.4	70.5	81.2	49.7	64.2	85.7
SpatialTracker	Window	Kub	61.1	76.3	89.5	63.5	77.6	88.2	50.1	65.9	86.9
LocoTrack	Video	Kub	62.9	75.3	87.2	69.7	83.2	89.5	52.9	<u>66.8</u>	85.3
CoTracker3	Window	Kub-L	64.5	76.7	89.7	71.1	81.9	90.3	**54.1**	66.6	<u>87.1</u>
CoTracker3	Video	Kub-L	63.3	76.2	88.0	**74.0**	<u>84.9</u>	<u>90.5</u>	53.5	66.5	86.4
BootsTAPIR	Video	Kub + R	61.4	73.6	88.7	70.8	83.0	89.9	54.6	68.4	86.5
CoTracker3	Window	Kub-L + R	63.8	76.3	90.2	71.7	83.6	91.1	55.8	68.5	88.3
CoTracker3	Video	Kub-L + R	64.4	76.9	91.2	74.3	85.2	92.4	54.7	67.8	87.4
Online											
DynOMo	Frame	-	45.8	63.1	81.1	-	-	-	-	-	-
MFT	Frame	SFK	47.3	66.8	77.8	-	-	-	39.6	60.4	72.7
Online TAPIR	Frame	Kub	56.7	70.2	85.7	67.7	-	-	51.5	64.4	85.2
Track-On (*Ours*)	Frame	Kub	**65.0**	**78.0**	90.8	<u>71.4</u>	**85.2**	**91.7**	<u>53.9</u>	**67.3**	**87.8**

the entire video at once.

Comparison on RGB-Stacking: The dataset consists of long video sequences, with lengths of up to 250 frames, making it ideal for evaluating models' long-term processing capabilities. Our model surpasses Online TAPIR by 3.7 AJ and outperforms offline competitors, achieving improvements of 0.3 in $\delta^x_{avg}$and 1.2 in OA compared to CoTracker3, which utilizes video-level input. The results of offline models on this dataset highlight a significant limitation of the windowed inference approach, which struggles with long video sequences due to restricted temporal coverage. In contrast, models with full video input perform considerably better. By effectively extending the temporal span through our memory mechanisms, our model achieves comparable or superior performance on long videos using only frame-by-frame inputs, despite the inherent disadvantage of not having bidirectional connections across the entire video sequence.

Comparison on Kinetics: The dataset comprises a variety of long internet videos. Our model outperforms Online TAPIR across all metrics by a considerable margin, while also surpassing offline models in $\delta^x_{avg}$and OA. Specifically, it achieves a 0.5 improvement in δ^x_{avg}over LocoTrack (with video inputs) and a 0.7 improvement in OA over CoTracker3 (with window inputs). Despite the significant difference in training data between CoTracker3 and our model, ours ranks second in AJ, with only a small gap of 0.2. Additionally, models fine-tuned on real-world data demonstrate superior performance, underscoring the potential benefits of training on large-scale real-world datasets, which seem particularly advantageous for datasets like Kinetics compared to others.

RoboTAP

We evaluate our model on the RoboTAP dataset [Vecerik et al., 2023], which consists of 265 real-world robotic sequences with an average length of over 250 frames, as shown in Table 4.2. We use the same metrics as the TAP-Vid benchmark: AJ, δ_{avg}, and OA, with the memory size K_i set to 48. Our model consistently surpasses existing online and offline models across all metrics. Specifically, in AJ and δ_{avg}, our

Table 4.2: **Quantitative Results on RoboTAP, Dynamic Replica, and BADJA** This table shows results in comparison to the previous work on RoboTAP, Dynamic Replica, and BADJA under queried first setting. The models are categorized into online and offline schemes, the former setting grants access to any frame regardless of video length, thus providing a clear advantage. While online models process one frame at a time, enable frame-by-frame inference. For training datasets, Kub and Kub-L(ong), refer to the TAP-Vid Kubric dataset with 24-frame and 64-frame videos, respectively; and R indicates the inclusion of a large number of real-world videos, we highlight these models in gray.

Model	Input	Train	RoboTAP			Dynamic Replica	BADJA	
			AJ ↑	δ^x_{avg} ↑	OA ↑	δ^{vis} ↑	δ^{seg} ↑	δ^{3px} ↑
Offline								
TAPIR	Video	Kub	59.6	73.4	87.0	66.1	66.9	15.2
TAPTR	Window	Kub	60.1	75.3	86.9	69.5	64.0	18.2
TAPTRv2	Window	Kub	60.9	74.6	87.7	-	-	-
SpatialTracker	Window	Kub	-	-	-	-	69.2	17.1
LocoTrack	Video	Kub	62.3	76.2	87.1	71.4	-	-
CoTracker3	Window	Kub-L	60.8	73.7	87.1	72.9	-	-
CoTracker3	Video	Kub-L	59.9	73.4	87.1	69.8	-	-
BootsTAPIR	Video	Kub + R	64.9	80.1	86.3	69.0	-	-
CoTracker3	Window	Kub-L + R	66.4	78.8	90.8	73.3	-	-
CoTracker3	Video	Kub-L + R	64.7	78.0	89.4	72.2	-	-
Online								
Online TAPIR	Frame	Kub	59.1	-	-	-	-	-
Track-On (*Ours*)	Frame	Kub	**63.5**	**76.4**	**89.4**	**73.6**	**71.9**	**20.2**

model outperforms the closest competitor, LocoTrack (which processes the entire video), by 1.2 and 0.2 points, respectively. Additionally, it exceeds the nearest competitor (TAPTRv2) in OA by 1.7 points. This demonstrates that our causal memory modules, which enable online tracking, are capable of effectively capturing the dynamics of long video sequences despite lacking bidirectional information flow across

all frames. It is worth noting that this dataset, which features textureless objects, presents a significant challenge. Fine-tuning on real-world videos provides substantial improvements, as learning to track points on textureless objects is particularly difficult, as highlighted by models tuned on real-world datasets.

Dynamic Replica

We compare to previous work on the Dynamic Replica dataset [Karaev et al., 2023], a benchmark designed for 3D reconstruction with 20 sequences, each consisting of 300 frames, as shown in Table 4.2. Following prior work [Karaev et al., 2024b], we evaluate models using δ^{vis}, consistent with the TAP-Vid benchmark. Unlike previous work, we do not report δ^{occ}, as our model is not supervised for occluded points. The memory size is set to $K_i = 48$. Despite being an online model, our model outperforms offline competitors, including those trained on longer sequences (CoTracker3, 73.6 vs. 72.9) and versions fine-tuned on real-world videos (73.6 vs. 73.3). This highlights the robustness of our model, particularly in handling longer video sequences effectively.

BADJA

We compare to previous work on the BADJA challenge [Biggs et al., 2019], a dataset for animal joint tracking comprising 7 sequences, as shown in Table 4.2. Two metrics are used for evaluation: δ^{seg}, which measures the proportion of points within a threshold relative to the segmentation mask size (specifically, points within $0.2\sqrt{A}$, where A is the area of the mask); and δ^{3px}, the ratio of points tracked within a 3-pixel range. Given the dataset's low FPS nature, we kept the memory size at the original value of 12. Our model achieves state-of-the-art results by a significant margin, with a 2.7-point improvement in δ^{seg} over SpatialTracker and a 2.0-point improvement in δ^{3px} over TAPTR. These results highlight the flexibility of our inference-time memory extension, enabling the model to adapt effectively to data with varying characteristics.

Table 4.3: **Quantitative Results on PointOdyssey.** This table shows results in comparison to the previous work on PointOdyssey under queried first setting.

Model	Input	Train	PointOdyssey			
			δ^{vis}_{avg} ↑	δ^{all}_{avg} ↑	MTE ↓	Survival ↑
TAP-Net	Frame	Kub	-	23.8	92.0	17.0
TAP-Net	Frame	PO	-	28.4	63.5	18.3
PIPs	Window	Kub	-	16.5	147.5	32.9
PIPs	Window	PO	-	27.3	64.0	42.3
PIPs++	Window	PO	32.4	29.0	-	47.0
CoTracker	Window	PO	<u>32.7</u>	<u>30.2</u>	-	**55.2**
Track-On (*Ours*)	Frame	Kub	**38.1**	**34.2**	**28.8**	<u>49.5</u>

PointOdyssey

We evaluated our model, trained on TAP-Vid Kubric, on the PointOdyssey (PO) [Zheng et al., 2023] dataset, which consists of 12 long videos with thousands of frames (up to 4325). The results are presented in Table 4.3. We adopted four evaluation metrics proposed in Point Odyssey: δ^{vis}_{avg}, which measures the δ_{avg} metric from the TAP-Vid benchmark for visible points; δ^{all}_{avg}, which calculates δ_{avg} for all points, including both visible and occluded ones; MTE (Median Trajectory Error), computed for all points; and Survival Rate, defined as the average number of frames until tracking failure (set to 50 pixels). The memory size K_i was set to 96. From the results, we observe that PIPs trained on Kubric achieves a δ^{vis}_{avg} of 16.5, while the same model trained on PO with a larger window size achieves 27.3 ($\sim$ 65% improvement). Notably, CoTracker does not report the performance of its model trained on Kubric but instead reports results for a model trained with sequences of length 56 on PO. These findings highlight the importance of training on PO to achieve higher performance across models. Our model, trained on Kubric, outperforms CoTracker and PIPs++ trained on PO in both δ^{vis}_{avg} and δ^{all}_{avg}. Interestingly, while training on PO is critical for other models to achieve strong performance, our model demonstrates

robustness by surpassing them even when trained on a different data distribution. Moreover, despite not being explicitly supervised for occluded points, our model still achieves superior δ^{all}_{avg}. In terms of the Survival Rate, our model falls behind CoTracker trained on PO, despite its superior δ metrics. This further emphasizes the importance of training on PO to excel in this specific metric.

4.2.3 Ablation Study

Components: We conducted an experiment to examine the impact of each proposed component in the correspondence estimation section (Section 4.1.2), we remove them one at a time while keeping other modules unchanged. First, we removed the re-ranking module Φ_{rank}. Second, we removed the offset head Φ_{off}, eliminating the calculation of additional offsets. Instead, we used the coarse prediction, *i.e.* the selected patch center, as the final prediction. Lastly, we replaced the additional deformable attention layer in the visibility head Φ_{vis} with a 2-layer MLP. Note that, we do not apply inference-time memory extension to models in this comparison.

From the results in Table 4.4, we can make the following observations: (i) The re-ranking module improves all metrics, notably increasing AJ by 2.1, as it introduces specialized queries for identifying correspondences. Errors larger than 16 pixels

Table 4.4: **Model Components.** Removing individual components of our model without (inference-time memory extension)—namely, the re-ranking module (Φ_{rank}), offset head (Φ_{off}), and visibility head (Φ_{vis}) one at a time. All metrics are higher-is-better.

Model	δ^{1px}	δ^{16px}	AJ	δ^{x}_{avg}	OA
Full Model (without IME)	45.5	95.9	64.9	77.7	90.6
- No re-ranking ($\Phi_{\text{re-rank}}$)	43.8	95.5	62.8	76.3	89.7
- No offset head (Φ_{off})	27.6	96.1	60.1	73.0	90.5
- No visibility head (Φ_{vis})	45.4	96.1	64.0	77.4	90.6

are also more frequent without it, showing its role in reducing large errors. (ii) The offset head is crucial for fine-grained predictions. While δ^{16px} values remain similar without the offset head, lower error thresholds (*i.e.* less than 1 pixel) show a significant difference (45.5 vs. 27.6), highlighting the importance of predicted offsets for fine-grained localization. (iii) Replacing the deformable attention layer in Φ_{vis} with an MLP does not affect OA but reduces AJ. The deformable head ensures more consistent visibility predictions by conditioning them on accurate point predictions, leading to higher AJ. Despite this, OA remains robust even when an MLP is used for visibility prediction.

Offset Head and Stride: The offset head is essential for refining patch classification outputs, enabling more precise localization. Specifically, the offset head allows for precision beyond the patch size S (stride). In Table 4.5, we examine the impact of removing the offset head (Φ_{off}) for two stride values, $S = 4$ and $S = 8$, without utilizing inference-time memory extension. For both values, the addition of the offset head significantly enhances AJ and $\delta^x_{avg}$by refining predictions within the local region. With stride 4, the offset head notably improves δ^{2px}, while for stride 8, it improves both δ^{2px} and δ^{4px}. This demonstrates that while patch classification offers coarse localization, the offset head provides further refinement, achieving pixel-level precision.

Larger stride values risk losing important details necessary for accurate tracking.

Table 4.5: **Offset Head.** The effect of removing the offset head (Φ_{off}) on models with varying strides. All metrics are higher-is-better.

Φ_{off}	Stride	δ^{2px}	δ^{4px}	δ^{8px}	AJ	δ^x_{avg}	OA
✗	8	37.4	79.0	91.1	51.3	62.9	91.0
✓		66.1	84.0	91.7	62.5	75.8	90.6
✗	4	64.3	84.4	92.4	60.1	73.0	90.5
✓		69.3	85.5	92.5	64.9	77.7	90.6

For instance, increasing the stride from 4 to 8 results in AJ drops of 12% for TAPIR and 16% for CoTracker, as reported in their ablation studies. However, our coarse-to-fine approach mitigates the negative effects of stride 8, leading to only a minimal decline of 4%, highlighting the robustness of our model to larger stride values.

Note that the model with $S = 8$ and no offset head (first row) has a higher occlusion accuracy (OA). A possible reason is the imbalance in the loss, where the visibility loss has a relatively higher impact compared to the model with an additional offset loss (second row), leading to improved occlusion accuracy.

Memory Modules: To demonstrate the effectiveness of our proposed memory modules, we conduct an ablation study, as shown in Table 4.6. We start by evaluating the model without memory (Model-A), which corresponds to the vanilla model described in Section 4.1.2. As expected, due to the model's lack of temporal processing, Model-A performs poorly, particularly in OA. Introducing temporal information through either spatial memory (Model-B) or context memory (Model-C) leads to significant performance improvements. Model-C, in particular, achieves higher OA by providing a more comprehensive view of the track's history, including occlusions. Combining both memory types (Model-D) further boosts performance, highlighting the complementary strengths of the two memory modules. Lastly, incorporating the memory extension at inference time yields slight improvements in all metrics,

Table 4.6: **Memory Components.** The effect of spatial memory ($\mathbf{M}^s$), context memory ($\mathbf{M}^c$), and inference-time memory extension (IME). All metrics are higher-is-better.

Model	$\mathbf{M}^s$	$\mathbf{M}^c$	IME	AJ	δ^x_{avg}	OA
A	✗	✗	✗	52.0	67.6	78.1
B	✓	✗	✗	63.5	77.0	89.0
C	✗	✓	✗	64.3	77.8	90.3
D	✓	✓	✗	64.9	77.7	90.6
E	✓	✓	✓	65.0	78.0	90.8

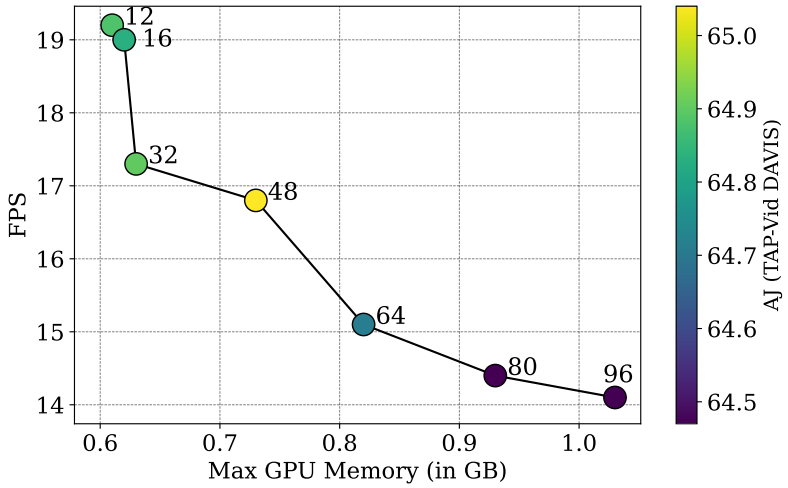

Figure 4.7: **Efficiency.** Inference speed (frames per second, FPS) vs. maximum GPU memory usage (in GB) where color represents the performance in AJ for different memory sizes (indicated near the nodes), while tracking approximately 400 points on the DAVIS dataset.

leading to an overall enhancement in performance.

Efficiency: We plot the inference speed (frames per second, FPS), maximum GPU memory usage during video processing, and AJ performance on the TAP-Vid DAVIS dataset as a function of memory size K_i (indicated near the plot nodes) in Figure 4.7. The results are based on tracking approximately 400 points on a single NVIDIA A100 GPU. Unlike offline methods, our approach does not utilize temporal parallelization in the visual encoder, processing frames sequentially in an online setting. As the memory size K increases, the model's inference speed decreases due to the higher computational cost of temporal attention in memory operations, correspondingly increasing GPU memory usage. For instance, the FPS decreases from 19.2 with $K = 12$ to 16.8 with $K = 48$, and further down to 14.1 with $K = 96$.

Additionally, our model demonstrates high memory efficiency, with GPU memory usage ranging from 0.61 GB ($K = 12$) to a maximum of 1.03 GB ($K = 96$). At the default memory size of $K = 48$, where our model performs best on this dataset, it achieves 16.8 FPS with a maximum GPU memory usage of 0.73 GB. This highlights the efficiency of our frame-by-frame tracking approach, making it well-suited for

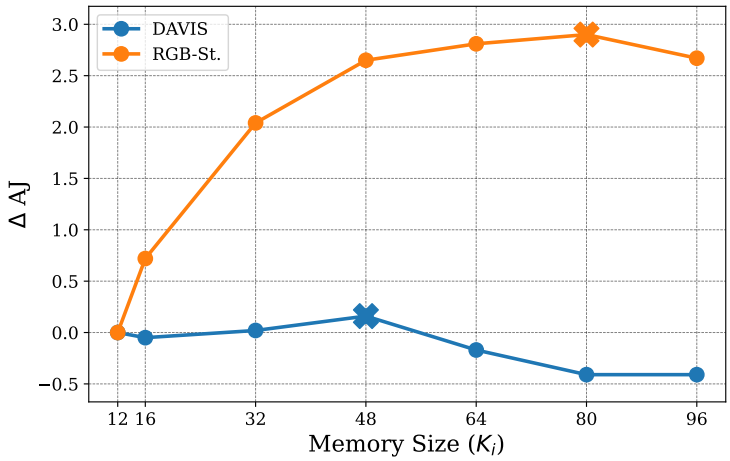

Figure 4.8: **Memory Size.** The effect of varying extended memory sizes during inference, on TAP-Vid DAVIS and TAP-Vid RGB-Stacking.

consumer GPUs and real-time applications. Moreover, we observe that performance improves as the memory size increases up to $K = 48$, but declines beyond this point. This suggests that excessively large memory sizes can hurt performance by storing unnecessary information.

Memory Size: We experimented with varying memory sizes, trained with $K = 12$, and extended them to different values $K = \{16, 32, 48, 64, 80, 96\}$ during inference on TAP-Vid DAVIS and TAP-Vid RGB-Stacking, as shown in Figure 4.8. The plot shows the change in AJ compared to the default training memory size of 12 after extension. Memory sizes reported in Table 4.1 are marked with crosses. For DAVIS (blue), performance slightly increases up to a memory size of 48 (64.88 AJ $\rightarrow$ 65.01 AJ) but declines beyond that, indicating that excessive memory can negatively impact the model. In contrast, for RGB-Stacking (orange), memory size plays a more critical role due to the disparity in video frame counts between training (24 frames) and inference (250 frames), as well as the high FPS nature of the dataset. Performance consistently improves up to $K = 80$, yielding a 2.9 AJ increase. These results highlight that, although the model is trained with a fixed and relatively small memory size, extending memory during inference is possible to adapt the varying characteristics of different datasets.

Table 4.7: **Spatial Memory.** Comparison of the model's performance with and without spatial memory ($\mathbf{M}^s$), evaluated using the AJ metric across different datasets, with inference-time memory extension (IME) applied.

Model	DAVIS	RGB-Stacking	Kinetics	RoboTAP
Full Model	65.0	71.4	53.9	63.5
- Without Spatial Memory ($\mathbf{M}^s$)	64.6	70.2	53.3	62.1

Spatial memory: To evaluate the effect of spatial memory in the presence of feature drift and inference-time memory extension, we conduct an experiment across different datasets using a model trained without spatial memory (Model-C in Table 4.6), as shown in Table 4.7. The results indicate that spatial memory consistently improves AJ across four datasets: DAVIS, RGB-Stacking, Kinetics, and RoboTAP. The impact is particularly notable for RGB-Stacking (+1.2 AJ) and RoboTAP (+1.4 AJ), where objects are less descriptive and often textureless, as both datasets originate from robotics scenarios. This suggests that spatial memory, which retains information around the local region of previous predictions, helps mitigate drift and enhances generalization across different scene characteristics.

Additionally, to directly assess the impact of spatial memory (Section 4.1.3) in mitigating feature drift, we conducted an analysis comparing the tracking performance of the initial feature sampled from the query frame, $\mathbf{q}^{init}$, with the query feature updated using spatial memory at frame t, denoted as $\mathbf{q}_t^{init}$. For this evaluation, we introduced the new metric of similarity ratio score (s_{sr}), which measures how well the updated query features align with the feature at the target point compared to the initial query.

Ideally, $\mathbf{q}_t^{init}$ should provide a better starting point for detecting correspondences compared to $\mathbf{q}^{init}$, particularly when the object's appearance changes significantly. To assess whether $\mathbf{q}_t^{init}$ is more similar to the feature at the ground-truth correspondence location than $\mathbf{q}^{init}$, we calculate the ratio of their similarity to ground-truth,

as a way of quantifying the increase in the similarity after the update:

$$s_{sr}(t) = \frac{\mathbf{q}_t^{init} \cdot \text{sample}(\mathbf{f}_t,\ \mathbf{p}_t)}{\mathbf{q}^{init} \cdot \text{sample}(\mathbf{f}_t,\ \mathbf{p}_t)} \tag{4.11}$$

Here, $\mathbf{p}_t$ represents the location of the ground-truth correspondence point, and $\mathbf{f}_t$ is the feature map of the target frame. On the DAVIS dataset, we calculated s_{sr} for visible points, achieving a score of 1.24, indicating that spatial memory introduces a 24% increase in similarity compared to the initial feature. In Figure 4.9, we visualize the similarity scores for different tracks over time for two videos from the DAVIS dataset. The plot highlights that the similarity increases more significantly toward the end of the video, where appearance changes are more severe. Moreover, the score is consistently greater than 1, showing that $\mathbf{q}_t^{init}$ always provides better initialization than $\mathbf{q}^{init}$ in these videos.

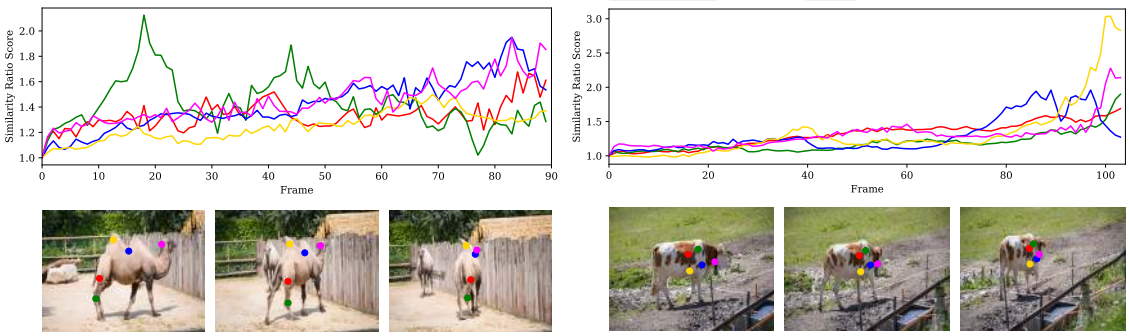

Figure 4.9: **Similarity Ratio Score.** The similarity ratio score $s_{sr} > 1$ over frames for different tracks, demonstrates increased similarity with ground-truth location on the target frame when utilizing spatial memory.

Chapter 5

CONCLUSION

5.1 Discussion

In this thesis, we investigated the task of long-term point tracking with a focus on online settings, where the model must process video frames sequentially and cannot rely on future information. We approached this challenge from two complementary perspectives: evaluating the geometric capabilities of visual foundation models for point tracking, and designing a purpose-built transformer architecture that enables causal, efficient, and scalable tracking over long video sequences.

In Chapter 3, we analyzed whether FoMos trained on large-scale data can support point tracking through simple geometric similarity. Our findings revealed that certain FoMos exhibit strong geometric awareness, especially Stable Diffusion in zero-shot settings, and DINOv2 under lightweight adaptation. In particular, we showed that DINOv2 can match and even surpass the performance of fully supervised models, despite using fewer parameters and significantly less task-specific supervision. This demonstrates that these models encode meaningful correspondence priors and can serve as effective backbones for temporal reasoning tasks.

However, our evaluation also highlighted a key gap: while FoMos can localize points accurately across individual frames, they lack the temporal modeling capabilities required for consistent long-term tracking, especially under occlusion and appearance changes. This motivated the second part of the thesis, presented in Chapter 4, where we introduced **Track-On**, a transformer-based model explicitly designed for online point tracking.

Track-On treats each point as a query token in a transformer decoder and estimates correspondences via patch classification followed by local offset refinement. To handle temporal consistency without requiring full video access, the model main-

tains two complementary memory modules: a spatial memory that stores localized features around previous predictions to reduce feature drift, and a context memory that aggregates the trajectory's history to improve robustness and visibility estimation. This design enables the model to operate in a fully causal setting, tracking points frame-by-frame, while still capturing long-term dependencies.

Through extensive experiments on seven datasets, we demonstrated that Track-On achieves state-of-the-art performance among online models. It also competes closely with, and in some cases surpasses, offline models that have access to full videos or sliding windows of frames. We also show that Track-On is highly efficient, requiring minimal GPU memory and enabling fast inference, making it suitable for deployment in real-time applications.

Our ablation studies further validate the effectiveness of each component in our architecture. The patch classification and re-ranking modules improve robustness in coarse matching, while the offset head significantly enhances fine-grained localization. The memory modules play a crucial role in maintaining temporal coherence, particularly in long sequences. We also demonstrate that inference-time memory extension allows the model to scale beyond the fixed training window, adapting to varying sequence lengths without retraining.

In summary, this thesis presents two complementary contributions toward the goal of long-term online point tracking. First, we show that foundation models offer strong geometric cues that can be exploited with minimal supervision. Second, we propose a transformer-based architecture that translates these insights into an efficient and accurate online tracking model. Together, these components provide a practical and scalable solution to the problem of causal motion understanding in videos.

5.2 Limitations and Future Work

A key limitation of this work, shared by most point tracking approaches, is the domain gap between training and test data. Current models are primarily trained on synthetic datasets such as TAP-Vid Kubric [Doersch et al., 2022], where dense

ground-truth correspondences can be generated automatically. In contrast, real-world datasets remain scarce and limited in diversity, as collecting ground-truth 2D trajectories across long videos is extremely labor-intensive and difficult to scale. As a result, available real-world datasets, such as TAP-Vid DAVIS or RoboTAP, are typically small in size and only used for benchmarking, not for training.

To address this, recent efforts like BootsTAP [Doersch et al., 2024] and CoTracker3 [Karaev et al., 2024a] explore pseudo-labeling strategies, bootstrapping real-world videos using predictions from existing models. While these methods offer improvements, they inherit the limitations of the teacher models. For instance, CoTracker3 is trained on pseudo-labels that are direct outputs of TAPIR and CoTracker, sharing the biases and failure modes of those models.

Emerging large-scale datasets such as Stereo4D [Jin et al., 2025] and DynPose-100K [Rockwell et al., 2025] open new possibilities for bridging this gap. These datasets use a combination of 3D scene reconstruction tools, like structure-from-motion, stereo depth estimation, and feature matching, to annotate massive collections of real-world internet videos with accurate camera poses and dynamic 3D point trajectories. For example, DynPose-100K achieves robust camera pose estimation for 100K dynamic videos using a fusion of advanced masking, tracking, and global bundle adjustment.

One promising direction is to leverage these reconstructions as a basis for extracting high-quality 2D point tracks from real-world 4D scenes. By projecting the stable 3D trajectories back to 2D views across time, we can construct a dataset of real-world point correspondences that covers a wide range of scene types, motions, and camera behaviors. Such a dataset would enable supervised training or self-training of point trackers with significantly improved realism and diversity, helping to close the domain gap between synthetic and real-world data. If adopted widely, this approach could benefit the broader point tracking community by establishing a scalable pipeline for collecting reliable training annotations from casually captured internet video.

Another important research direction involves improving point tracker design by leveraging scene-level geometry estimation. Recent models such as DUST3R [Wang

et al., 2024], VGGT [Wang et al., 2025] and MonST3R [Zhang et al., 2025] predict dense 3D structure from images. Particularly, MonST3R generalizes to dynamic scenes; and produces per-frame dynamic point clouds and camera poses. This opens up a new tracking formulation: instead of relying solely on 2D appearance features, we can lift the query point into the 3D space, track its motion in the world coordinate system, and reproject it into future frames using the estimated camera motion. This geometric perspective has the potential to resolve ambiguities in textureless regions, improve robustness to occlusion, and ensure physical consistency in the predicted trajectories. Integrating such geometric priors with our transformer-based tracking architecture is an exciting next step toward accurate and causally consistent tracking in the wild.

BIBLIOGRAPHY

[Amir et al., 2021] Amir, S., Gandelsman, Y., Bagon, S., and Dekel, T. (2021). Deep ViT features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*.

[Aydemir et al., 2023a] Aydemir, G., Akan, A. K., and Güney, F. (2023a). ADAPT: Efficient multi-agent trajectory prediction with adaptation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Aydemir et al., 2025] Aydemir, G., Cai, X., Xie, W., and Güney, F. (2025). Track-On: Transformer-based online point tracking with memory. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Aydemir et al., 2023b] Aydemir, G., Xie, W., and Guney, F. (2023b). Self-supervised object-centric learning for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Aydemir et al., 2024] Aydemir, G., Xie, W., and Güney, F. (2024). Can visual foundation models achieve long-term point tracking? In *Proc. of the European Conf. on Computer Vision (ECCV) Workshops*.

[Balazevic et al., 2024] Balazevic, I., Shi, Y., Papalampidi, P., Chaabouni, R., Koppula, S., and Henaff, O. J. (2024). Memory consolidation enables long-context video understanding. In *Proc. of the International Conf. on Machine learning (ICML)*.

[Barın et al., 2024] Barın, M. R., Aydemir, G., and Güney, F. (2024). Robust bird's eye view segmentation by adapting dinov2. In *Proc. of the European Conf. on Computer Vision (ECCV) Workshops*.

[Battiato et al., 2007] Battiato, S., Gallo, G., Puglisi, G., and Scellato, S. (2007). SIFT features tracking for video stabilization. In *Proc. of the International Conference on Image Analysis and Processing (ICIAP)*.

[Biggs et al., 2019] Biggs, B., Roddick, T., Fitzgibbon, A., and Cipolla, R. (2019). Creatures great and SMAL: Recovering the shape and motion of animals from video. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*.

[Black and Anandan, 1993] Black, M. and Anandan, P. (1993). A framework for the robust estimation of optical flow. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Bruhn et al., 2005] Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision (IJCV)*.

[Buch et al., 2017] Buch, S., Escorcia, V., Shen, C., Ghanem, B., and Carlos Niebles, J. (2017). SST: Single-stream temporal action proposals. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Chen et al., 2022a] Chen, J., Mittal, G., Yu, Y., Kong, Y., and Chen, M. (2022a). GateHUB: Gated history unit with background suppression for online action detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Chen et al., 2022b] Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., and Qiao, Y. (2022b). Vision transformer adapter for dense predictions. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Cheng and Schwing, 2022] Cheng, H. K. and Schwing, A. G. (2022). XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Cho et al., 2024] Cho, S., Huang, J., Nam, J., An, H., Kim, S., and Lee, J.-Y. (2024). Local all-pair correspondence for point tracking. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Darcet et al., 2023] Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. (2023). Vision transformers need registers. *arXiv preprint arXiv:2309.16588*.

[De Geest et al., 2016] De Geest, R., Gavves, E., Ghodrati, A., Li, Z., Snoek, C., and Tuytelaars, T. (2016). Online action detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Doersch et al., 2022] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Doersch et al., 2024] Doersch, C., Luc, P., Yang, Y., Gokay, D., Koppula, S., Gupta, A., Heyward, J., Rocco, I., Goroshin, R., Carreira, J., and Zisserman, A. (2024). BootsTAP: Bootstrapped training for tracking-any-point. *Proc. of the Asian Conf. on Computer Vision (ACCV)*.

[Doersch et al., 2023] 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 *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P. v. d., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[El Banani et al., 2024] El Banani, M., Raj, A., Maninis, K.-K., Kar, A., Li, Y., Rubinstein, M., Sun, D., Guibas, L., Johnson, J., and Jampani, V. (2024). Probing the 3D awareness of visual foundation models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Eun et al., 2020] Eun, H., Moon, J., Park, J., Jung, C., and Kim, C. (2020). Learning to discriminate information for online action detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Fan et al., 2021] Fan, Z., Liu, J., and Wang, Y. (2021). Motion adaptive pose estimation from compressed videos. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Harley et al., 2022] Harley, A. W., Fang, Z., and Fragkiadaki, K. (2022). Particle video revisited: Tracking through occlusions using point trajectories. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[He et al., 2018] He, A., Luo, C., Tian, X., and Zeng, W. (2018). A twofold siamese network for real-time object tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[He et al., 2022] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Hedlin et al., 2023] Hedlin, E., Sharma, G., Mahajan, S., Isack, H., Kar, A., Tagliasacchi, A., and Yi, K. M. (2023). Unsupervised semantic correspondence using stable diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. In *Artificial Intelligence (AI)*.

[Hu et al., 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Jasinschi et al., 1998] Jasinschi, R. S., Veen, T. N., et al. (1998). Motion estimation methods for video compression—a review. *Journal of the Franklin Institute*.

[Jin et al., 2025] Jin, L., Tucker, R., Li, Z., Fouhey, D., Snavely, N., and Holynski, A. (2025). Stereo4d: Learning how things move in 3d from internet stereo videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Karaev et al., 2024a] Karaev, N., Makarov, I., Wang, J., Neverova, N., Vedaldi, A., and Rupprecht, C. (2024a). CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*.

[Karaev et al., 2023] Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., and Rupprecht, C. (2023). DynamicStereo: Consistent dynamic depth from stereo videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Karaev et al., 2024b] Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., and Rupprecht, C. (2024b). CoTracker: It is better to track together. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Kirillov et al., 2023] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023).

Segment anything. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Kondratyuk et al., 2021] Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., and Gong, B. (2021). MoViNets: Mobile video networks for efficient video recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Lee et al., 2009] Lee, K.-Y., Chuang, Y.-Y., Chen, B.-Y., and Ouhyoung, M. (2009). Video stabilization using robust feature trajectories. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Li et al., 2024a] Li, H., Zhang, H., Liu, S., Zeng, Z., Li, F., Ren, T., Li, B., and Zhang, L. (2024a). TAPTRv2: Attention-based position update improves tracking any point. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Li et al., 2024b] Li, H., Zhang, H., Liu, S., Zeng, Z., Ren, T., Li, F., and Zhang, L. (2024b). TAPTR: Tracking any point with transformers as detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Liang et al., 2020] Liang, Y., Li, X., Jafari, N., and Chen, J. (2020). Video object segmentation with adaptive feature bank and uncertain-region refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Loshchilov and Hutter, 2019] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Marchand et al., 2015] Marchand, E., Uchiyama, H., and Spindler, F. (2015). Pose estimation for augmented reality: a hands-on survey. In *IEEE Trans. on Visualization and Computer Graphics (VCG)*.

[Mayer et al., 2016] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks

for disparity, optical flow, and scene flow estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Melas-Kyriazi et al., 2022] Melas-Kyriazi, L., Rupprecht, C., Laina, I., and Vedaldi, A. (2022). Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Neoral et al., 2024] Neoral, M., Šerỳch, J., and Matas, J. (2024). MFT: Long-term tracking of every pixel. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*.

[Nie et al., 2019] Nie, X., Li, Y., Luo, L., Zhang, N., and Feng, J. (2019). Dynamic kernel distillation for efficient pose estimation in videos. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Oquab et al., 2024] Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.-Y., Xu, H., Sharma, V., Li, S.-W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. (2024). DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*.

[Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *Proc. of the International Conf. on Machine learning (ICML)*.

[Ravi et al., 2025] Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., and Feichtenhofer, C. (2025). SAM 2: Segment anything in images and videos. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Rockwell et al., 2025] Rockwell, C., Tung, J., Lin, T.-Y., Liu, M.-Y., Fouhey, D. F., and Lin, C.-H. (2025). Dynamic camera poses and where to find them. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Sand and Teller, 2008] Sand, P. and Teller, S. (2008). Particle video: Long-range motion estimation using point trajectories. In *International Journal of Computer Vision (IJCV)*.

[Seidenschwarz et al., 2025] Seidenschwarz, J., Zhou, Q., Duisterhof, B., Ramanan, D., and Leal-Taixé, L. (2025). DynOMo: Online point tracking by dynamic online monocular gaussian reconstruction. In *Proc. of the International Conf. on 3D Vision (3DV)*.

[Shi et al., 2023] Shi, X., Huang, Z., Li, D., Zhang, M., Cheung, K. C., See, S., Qin, H., Dai, J., and Li, H. (2023). Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Singh et al., 2017] Singh, G., Saha, S., Sapienza, M., Torr, P. H., and Cuzzolin, F. (2017). Online real-time multiple spatiotemporal action localisation and prediction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Sun et al., 2018] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Tang et al., 2023] 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 (NeurIPS)*.

[Teed and Deng, 2020] Teed, Z. and Deng, J. (2020). RAFT: Recurrent all-pairs field transforms for optical flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Touvron et al., 2022] Touvron, H., Cord, M., and Jégou, H. (2022). DeiT III: Revenge of the ViT. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Tumanyan et al., 2024] Tumanyan, N., Singer, A., Bagon, S., and Dekel, T. (2024). DINO-Tracker: Taming DINO for self-supervised point tracking in a single video. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Vecerik et al., 2023] Vecerik, M., Doersch, C., Yang, Y., Davchev, T., Aytar, Y., Zhou, G., Hadsell, R., Agapito, L., and Scholz, J. (2023). RoboTAP: Tracking arbitrary points for few-shot visual imitation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*.

[Wang et al., 2025] Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., and Novotny, D. (2025). Vggt: Visual geometry grounded transformer. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Wang et al., 2023a] Wang, Q., Chang, Y.-Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., and Snavely, N. (2023a). Tracking everything everywhere all at once. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Wang et al., 2024] Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., and Revaud, J. (2024). Dust3r: Geometric 3d vision made easy. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Wang et al., 2023b] Wang, X., Girdhar, R., Yu, S. X., and Misra, I. (2023b). Cut

and learn for unsupervised object detection and instance segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Wang et al., 2023c] Wang, X., Misra, I., Zeng, Z., Girdhar, R., and Darrell, T. (2023c). VideoCutLER: Surprisingly simple unsupervised video instance segmentation. *arXiv preprint arXiv:2308.14710*.

[Wang et al., 2021] Wang, X., Zhang, S., Qing, Z., Shao, Y., Zuo, Z., Gao, C., and Sang, N. (2021). OadTR: Online action detection with transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Wang et al., 2020] Wang, Z., Zheng, L., Liu, Y., Li, Y., and Wang, S. (2020). Towards real-time multi-object tracking. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Xiao et al., 2024] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Xu et al., 2023] Xu, J., Liu, S., Vahdat, A., Byeon, W., Wang, X., and De Mello, S. (2023). Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Xu et al., 2017] Xu, J., Ranftl, R., and Koltun, V. (2017). Accurate optical flow via direct cost volume processing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Xu et al., 2019] Xu, M., Gao, M., Chen, Y.-T., Davis, L. S., and Crandall, D. J. (2019). Temporal recurrent networks for online action detection. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Xu et al., 2021] Xu, M., Xiong, Y., Chen, H., Li, X., Xia, W., Tu, Z., and Soatto,

S. (2021). Long short-term transformer for online action detection. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Yang et al., 2022a] Yang, J., Liu, S., Li, Z., Li, X., and Sun, J. (2022a). Real-time object detection for streaming perception. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Yang et al., 2022b] Yang, L., Han, J., and Zhang, D. (2022b). Colar: Effective and efficient online action detection by consulting exemplars. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Yao et al., 2019] Yao, Y., Xu, M., Wang, Y., Crandall, D. J., and Atkins, E. M. (2019). Unsupervised traffic accident detection in first-person videos. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*.

[Zhan et al., 2023] Zhan, G., Zheng, C., Xie, W., and Zisserman, A. (2023). What does stable diffusion know about the 3D scene? *arXiv preprint arXiv:2310.06836*.

[Zhang et al., 2021a] Zhang, F., Woodford, O. J., Prisacariu, V., and Torr, P. H. (2021a). Separable flow: Learning motion cost volumes for optical flow estimation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Zhang et al., 2024a] Zhang, H., Wang, Y., Tang, Y., Liu, Y., Feng, J., Dai, J., and Jin, X. (2024a). Flash-VStream: memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*.

[Zhang et al., 2024b] Zhang, J., Herrmann, C., Hur, J., Chen, E., Jampani, V., Sun, D., and Yang, M.-H. (2024b). Telling left from right: Identifying geometry-aware semantic correspondence. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Zhang et al., 2025] Zhang, J., Herrmann, C., Hur, J., Jampani, V., Darrell, T., Cole, F., Sun, D., and Yang, M.-H. (2025). Monst3r: A simple approach for

estimating geometry in the presence of motion. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Zhang et al., 2021b] Zhang, Z., Zhou, C., Ma, J., Lin, Z., Zhou, J., Yang, H., and Zhao, Z. (2021b). Learning to rehearse in long sequence memorization. In *Proc. of the International Conf. on Machine learning (ICML)*.

[Zhao and Krähenbühl, 2022] Zhao, Y. and Krähenbühl, P. (2022). Real-time online video detection with temporal smoothing transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

[Zheng et al., 2023] Zheng, Y., Harley, A. W., Shen, B., Wetzstein, G., and Guibas, L. J. (2023). PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*.

[Zhou et al., 2024] Zhou, X., Arnab, A., Buch, S., Yan, S., Myers, A., Xiong, X., Nagrani, A., and Schmid, C. (2024). Streaming dense video captioning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Zhu et al., 2021] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. In *Proc. of the International Conf. on Learning Representations (ICLR)*.

[Zhu et al., 2023] Zhu, Y., Shen, Z., Zhao, Z., Wang, S., Wang, X., Zhao, X., Shen, D., and Wang, Q. (2023). MeLo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. *arXiv preprint arXiv:2311.08236*.