TAPTR: Tracking Any Point with Transformers as Detection

Hongyang Li[1,2], Hao Zhang[2,3], Shilong Liu[2,4], Zhaoyang Zeng[2], Tianhe Ren[2], Feng Li[2,3], and Lei Zhang[1,2]

[1] South China University of Technology
[2] International Digital Economy Academy (IDEA)
[3] The Hong Kong University of Science and Technology
[4] Dept. of CST., BNRist Center, Institute for AI, Tsinghua University

Abstract. In this paper, we propose a simple yet effective approach for Tracking Any Point with TRansformers (TAPTR). Based on the observation that point tracking bears a great resemblance to object detection and tracking, we borrow designs from DETR-like algorithms to address the task of TAP. In TAPTR, in each video frame, each tracking point is represented as a point query, which consists of a positional part and a content part. As in DETR, each query (its position and content feature) is naturally updated layer by layer. Its visibility is predicted by its updated content feature. Queries belonging to the same tracking point can exchange information through self-attention along the temporal dimension. As all such operations are well-designed in DETR-like algorithms, the model is conceptually very simple. We also adopt some useful designs such as cost volume from optical flow models and develop simple designs to provide long temporal information while mitigating the feature drifting issue. TAPTR demonstrates strong performance with state-of-the-art performance on various datasets with faster inference speed.

1 Introduction

Understanding every pixel in a video and tracking their motions is a fundamental task in computer vision, which is of great importance to video object tracking, segmentation, action recognition, and physical world understanding. Previously, this task is normally simplified to optical flow estimation and has received much attention [14,16,40,42,45,49,55,63]. Since optical flow mainly solves the correspondence problem between two consecutive frames, the lack of long-range temporal information makes it ineffective in handling the situation when a tracking point is occluded. The works [10,34,43,48,50,57] dedicated to semantic key point tracking can handle the problem of occlusion. However, the semantics of tracking targets are limited to only a small range such as the joints of humans. To break the limitation of optical-flow estimation and key-point tracking, some recent works [5,7,11,18,33,64] propose to track any arbitrary point specified by a user in the whole video and formalize the task as Tracking Any Point (TAP).

Compared with optical flow estimation, the most important problem that TAP needs to address is to model long-range point motion which may include

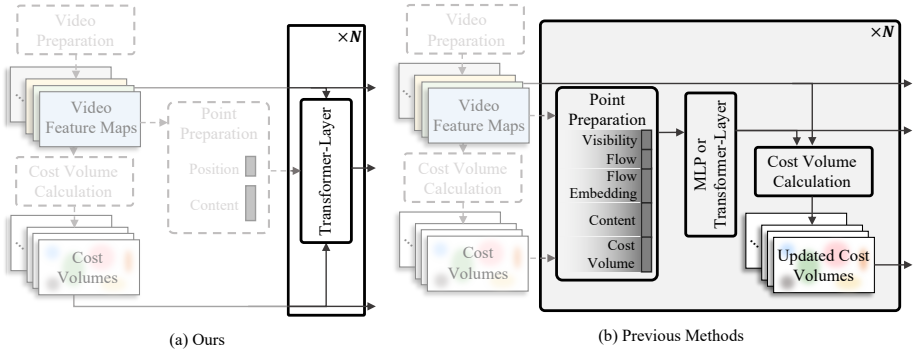

Fig. 1: Comparison of our well-designed DETR-like simple framework with meaning-clear point modeling and previous framework with redundant designs and blackbox point modeling. The operation within the dashed box will execute only once.

occlusions along the temporal axis. MFT [33] addresses this problem by extending an off-the-shelf optical flow estimation method RAFT [45] with the ability of visibility estimating and chaining the optical flow results to obtain the trajectory of a tracking point in the whole video. Although MFT has demonstrated impressive results, such an extension still lacks the capability of long-range temporal information modeling and falls short in more challenging point tracking tasks. To tackle this issue, several works [5, 7, 11, 64] utilize a sliding window-based approach and let the points in different frames in the same window exchange information along the temporal axis.

However, such works usually treat each tracking point independently and ignore the correlation between points, which is an inappropriate assumption when points, for example, belong to the same object, can provide contextual information for each other based on some physical laws [4, 30, 35, 41, 58]. To account for the correlation between tracking points, CoTracker [18] proposes to track all points simultaneously using a Transformer-based architecture.

A crucial problem in such works is how to model tracking points. As shown in Fig. 1 (b), in previous methods with iterative refinement [11, 18, 64], a tracking point is modeled as a concatenation of several features, including point flow vector as local movement, point flow embedding, point visibility, point content feature, and local correlation as cost volume. These features are normally well-designed in optical flow estimation algorithms and have clear physical meanings. However, previous methods [11, 18, 64] simply concatenate all features and send them as a blackbox vector to MLPs [11, 64] or Transformers [18] and expect MLPs or Transformers to decipher and utilize the features. We suggest that the previous methods may not fully achieve a clean model or facilitate ease of understanding. Therefore, a conceptually simple and effective approach might be beneficial for the task.

Inspired by DEtection TRansformer (DETR) [3] and its follow-ups [22,23,26–28,36–38,61,65], we find that point tracking bears a great resemblance to object

detection and tracking [31,44,56,59]. In particular, in each video frame, tracking points can be essentially regarded as queries, which have been extensively studied in DETR-like algorithms [21,23,26,32,52,61,65]. With this motivation, we follow DETR-like algorithms to design a simple model for tracking any point with Transformers.

Our pipeline is illustrated in Fig. 1 (a). In our method, in every frame, each tracking point is represented as a point query, which has a clear meaning. Each query consists of two parts, a positional part (point coordinate) and a content part, and will be refined layer by layer. Its visibility is predicted by its updated content feature. For multiple frames in a sliding window, queries belonging to the same tracking point can exchange information through self-attention operation along the temporal dimension. All such operations are common and well-designed in object detection and make the model conceptually simple yet performance-wise strong.

We introduce some designs based on the DETR-like model to further boost performance. To address the difference between point tracking and object detection/tracking, we take into account the well-established cost volume [45] into the Transformer decoder. This is driven by the fact that, compared with object detection/tracking, point tracking requires more local and low-level features to precisely locate and track desired points. Furthermore, we propose a simple yet effective design for updating content features within the decoder and between the windows to convey longer temporal information while mitigating the drifting issues in the context of TAP.

We conduct experiments on several challenging TAP datasets and demonstrate superior performance over prior works. Our model surpasses the current state of the art (CoTracker) on the DAVIS dataset under the same setting (63.0 vs. 60.7), and achieves this with 1.3 times faster speed. Remarkably, TAPTR outperforms CoTracker even when CoTracker deliberately tracks each single point at a time (63.0 vs. 62.2), while maintaining a 25 times faster speed.

2 Related Work

Optical Flow. Optical flow estimation is a long-standing fundamental computer vision task. Extensive research has been conducted in the past few decades [1, 2, 13]. In the last ten years, deep learning-based methods [8, 14–16, 40, 42, 49, 54, 55, 60, 63] have dominated this task. In particular, thanks to the regularity of image grid features, DCFlow [55] firstly utilizes cost volume for optical flow estimation. Inspired by DCFlow, many follow-up works [42, 45, 49] are built upon the cost volume, validating the effectiveness and robustness of cost volume in optical flow estimation. Among the follow-ups, the paradigm of recurrently looking up the cost volume and iteratively updating the optical flow estimation proposed by RAFT [45] obtained a remarkable performance and inspired many subsequent works [54, 60]. However, optical flow only addresses the correspondence problem between two consecutive frames, which is incapable of handling the occlusion issue when a tracking point is occluded in a long-time video sequence.

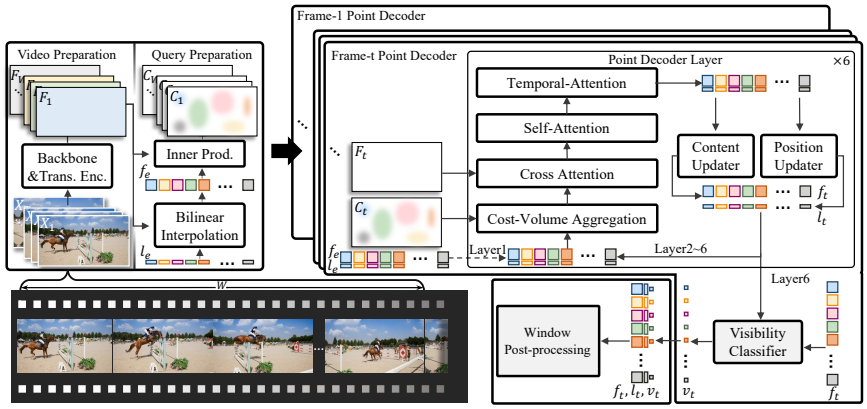

Fig. 2: The overview of TAPTR. The video preparation and query preparation parts provide the multi-scale feature map, point queries, and the cost volumes for the point decoder. The point decoder takes these elements as input and processes all frames in parallel. The outputs of the point decoder are sent to our window post-processing module to update the states of the point queries to their belonging tracking points.

Tracking Any Point. Compared to optical flow, each point to be tracked in the TAP task is arbitrarily selected in a video and is required to be tracked across the entire video. This task is more general and challenging. TAP-Vid [5] first formalizes the task and develops a challenging benchmark for the research community. MFT [33] proposes to track points by selecting the most reliable optical flow chain. However, the inherent limitations of optical flow estimation make MFT hard to handle more challenging point tracking tasks. OmniMotion [51] proposes to tackle this issue from the perspective of 3D. However, it requires a costly time-consuming test time optimization which prevents it from being applied in online scenarios. Recently, some general end-to-end point trackers have been proposed [5, 7, 11, 64], such as PIPs and TAP-Net. However, limited by the irregular data structure, they track all points in parallel independently. CoTracker [18] proposes to utilize the flexible Transformer architecture to construct the interaction between points and obtain remarkable performance.

3 TAPTR Model

3.1 Task Definition and Overview

Task Definition. Given a video, with T frames and any of the i-th tracking point in the video with an initial location $l_e^i = (x_e^i, y_e^i)$, our goal is to track the point across the video to obtain its trajectory, including its location sequence $L^i = \{l_t^i\}_{t=1}^T$, $l_t^i = (x_t^i, y_t^i)$, and its visibility sequence $V^i = \left\{v_t^i\right\}_{t=1}^T, v_t^i \in \{0, 1\}$. Here e is a special index indicating the time stamp when the tracking point first emerges or starts to be tracked.

Overview. Our model mainly consists of four parts as shown in Fig. 2. They are video preparation, query preparation, point decoder, and window post-processing.

Following previous works [11,18], we use a sliding-window strategy and process W frames once at a time. So for each window, video preparation is to extract feature maps for each frame with a backbone and a Transformer Encoder. Query preparation is to prepare initial locations l_e, content features f_e, and cost volumes $\{C_t\}_{t=1}^{W}$ for point queries in all frames of the window, where each point query in a frame has its unique belonging tracking point. The point decoder takes the point queries as input to detect their belonging tracking points' states as in DETR-like methods in all frames in parallel. Finally, there is a window post-processing part to update the ultimate states of each point query to the trajectory of its belonging tracking point.

3.2 Video Preparation

Following previous Transformer-based detectors and segmentors [23,26,61,65] we use a convolutional neural network as our backbone to get the multi-scale image feature maps and send the image feature maps into a Transformer-encoder to further improve the quality and receptive field of the image features. We define the final multi-scale image feature maps for the t-th frame as $F_t = \{F_{t,s}\}_{s=1}^{S}$, where S is the number of feature scales. The feature maps of all frames are obtained independently in parallel.

3.3 Query Preparation

In every frame, each tracking point will be assigned with a point query. The point query belonging to the i-th tracking point in the t-th frame is responsible for detecting the most matching point of its belonging point in the t-th frame.
Content Feature and Location. To obtain an initial content feature for a point query that accurately describes the point to be detected, we perform bilinear interpolation on the feature maps at the location where the belonging tracking point of the query point initially appears or starts to be tracked. Thus for the point queries that belong to the i-th point, their initial content feature can be obtained by

$$\begin{gathered} f_e^i = \texttt{MLP}\left(\texttt{Cat}\left(\texttt{Bili}\left(F_{e^i,1}, l_e^i\right), \texttt{Bili}\left(F_{e^i,2}, l_e^i\right), ..., \texttt{Bili}\left(F_{e^i,S}, l_e^i\right)\right)\right), \\ \forall 1 \le t \le T, \quad f_t^i \Leftarrow f_e^i \end{gathered} \tag{1}$$

where e^i is the timestamp when the i-th point first emerges or starts to be tracked. `Bili` and `Cat` stand for the bilinear interpolation and concatenation, respectively. `MLP` is a multi-layer perceptron, which works for the fusion of point features sampled on multi-scale feature maps. At the same time, the initial locations of the point queries that belong to the i-th point are initialized as l_e^i

$$\forall 1 \le t \le T, \quad l_t^i \Leftarrow l_e^i \tag{2}$$

For notation simplicity, in the following, we denote the query that belongs to the i-point in the t-th frame as a tuple $q_t^i = [f_t^i, l_t^i]$.

Cost Volume. The cost volume gives us an initial visual similarity between a point query and each pixel of the image. The effectiveness of cost volume for feature matching has been validated in many stereo-matching [20, 25, 39, 62] and optical flow estimation methods [16, 55]. Considering the characteristics of the TAP task, we further incorporate cost volume into TAPTR. However, different from previous works [18] that frequently recalculate the cost volume once the content feature of a query is updated, we only calculate cost volume once before the beginning of our decoder. This strategy keeps our decoder clean and the target of multi-layer refinement stable. In detail, to obtain the cost volume of the point query q_t^i we conduct inner product between the content feature of the point query and the image feature maps as in previous works [18, 55]

$$C_{t,s}^i = \texttt{InnerProd}(F_{t,s}, f_t^i), \tag{3}$$

where `InnerProd` indicates the inner product operation. Note that computing cost volume only once at the beginning does not mean that we do not update the cost volume anymore. For more details about the updating of cost volume and the effect of the updating on performance, please refer to Sec. 3.5 and Sec. 4.6.

3.4 Point Decoder

Given that when focusing on a single frame, the function of a point tracker essentially involves detecting the most matched point within the image. Therefore, the components of the original decoder align naturally with the TAP task, suggesting preserving these modules, particularly the self-attention and cross-attention.

Taking into consideration the characteristics of the TAP task, we further incorporate the well-validated cost volume into our decoder through a cost volume aggregation module. But as we have discussed, instead of updating cost volume frequently as in prior studies [18], to maintain the simplicity of the decoder, we treat the prepared cost volume as a static feature map similar to the original image feature maps. We directly reuse the cost volume across different layers of our decoder. To further utilize the temporal information, we adapt the attention mechanism along the temporal dimension through a temporal attention module.

Since each frame behaves the same in our point decoder, without loss of generality, in this section we take the t-th frame as an example to explain our decoder for simplicity.

Cost Volume Aggregation. Cost volume provides a basic visual similarity between the tracking point and a video frame. Instead of regressing the optical flow in a one-shot manner as in TAP-Net [5], thanks to the multi-layer design in our point decoder, we follow RAFT [45] to aggregate cost volume locally. More specifically, for the point query q_t^i, we conduct bilinear interpolation on the cost volume around its location in a grid form, this operation is commonly referred to as grid sampling `GridSample`. Then the sampled cost vector $c_t^i \in \mathbb{R}^{G \cdot G}$ is fused into the content feature of the point query to provide a basic perception of the image. The process can be formulated as

$$\begin{aligned} c_{t,s}^i &= \texttt{GridSample}\left(C_{t,s}^i, \texttt{Grid}\left(l_t^i, G\right)\right), \\ f_t^i &\Leftarrow \texttt{MLP}\left(\texttt{Cat}\left(c_{t,1}^i, c_{t,2}^i, \ldots, c_{t,S}^i, f_t^i\right)\right) \end{aligned} \tag{4}$$

where $C^i_{t,s}$ is the cost volume of q^i_t at the s-th scale, `Grid` is a function to generate a grid of sampling locations with grid size as G. Note that, like in RAFT, the sampling grid is shared in multi-scale cost volumes, resulting in a larger receptive field on smaller-scale feature maps.

Visual Feature Enhancer. As shown in previous feature-matching [29, 53] and optical flow estimation methods [16, 55], supplementing cost-volume with the original image features will provide more detailed geometrical information and thus result in a more robust feature to describe the points. In DETR-like architecture, the cross-attention naturally plays this role. Following [24, 61, 65], we use 2D deformable attention [65] to sample the multi-scale local image feature around the tracking point and fuse the sampled image feature into the point content feature. Thus for the point query q^i_t, this process can be formulated as

$$\begin{aligned} g^i_{t,s} &= \texttt{DFA2D}(F_{t,s}, l^i_t), \\ f^i_t &\Leftarrow \texttt{MLP}(\texttt{Cat}(g^i_{t,1}, g^i_{t,2}, \ldots, g^i_{t,S}, f^i_t)), \end{aligned} \tag{5}$$

where `DFA2D` indicates the 2D deformable attention operation and $g^i_{t,s}$ is the local geometrical feature for the i-th point in the s-th scale.

Interaction Among Point Queries. Limited by the irregularity of data structure, most previous works choose to process every point independently. Although CoTracker [18] first proposes to use the flexible attention mechanism to complete the interaction between points, their neglect of positional embedding limits the efficiency [26]. Following previous DETR-like methods, we add positional embedding to encourage the points queries to pay more attention to their neighboring queries, which can provide more useful contextual information. The process can be formulated as

$$p_t = \texttt{PE}\left(l_t, \tau\right), f_t \Leftarrow \texttt{Attention}\left(f_t + p_t, f_t + p_t\right). \tag{6}$$

In Eq. 6, $f_t \in \mathbb{R}^{N\times C}$ and $l_t \in \mathbb{R}^{N\times 2}$ indicate the content features and locations for all point queries in the t-th frame, where N and C indicate the number of points to be tracked and the number of feature channels, respectively. `Attention` and `PE` in Eq. 6 indicate the dense attention operation [47] and the sinusoidal positional encoding [3], respectively. As the key parameter of `PE`, $\tau \in \mathbb{R}$ represents the temperature for positional encoding [26], the lower τ is, the sharper positional embedding we get. Considering that point detection has a higher fine-grained requirement than object detection, we lower the default value of τ down.

Interaction Along Temporal Dimension. To better utilize the temporal information, we append temporal attention in our decoder. Temporal attention conducts dense attention along the temporal dimension independently for each tacking point. Thus for point queries that belong to the i-th point, the process can be formulated as

$$f^i \Leftarrow \texttt{Attention}(f^i, f^i), \tag{7}$$

where $f^i \in \mathbb{R}^{W\times C}$ indicates the content features of point queries that belong to the i-th point in all of the W frames.

Point Query Updating. Following the DETR-like methods, we update the location of each point query with the help of `Sigmoid`. As for the updating of content features, affected by the occlusion and drift, compared with the features that are updated through the above 4 blocks, the initial feature of each point query is the most reliable one. Inspired by the residual connection [12], we update the content feature of point queries in a residual mechanism. More specifically,

$$\begin{aligned} \Delta l_t^i = \texttt{MLP}(f_t^i), \Delta f_t^i = \texttt{MLP}(\texttt{Cat}(f_t^i, f_{e^i}^i)) \\ l_t^i \Leftarrow \texttt{Sigmoid}(\texttt{Sigmoid}^{-1}(l_t^i) + \Delta l_t^i), f_t^i \Leftarrow f_t^i + \Delta f_t^i. \end{aligned} \tag{8}$$

where $\texttt{Sigmoid}^{-1}$ is the inverse of sigmoid function. The updated point queries will be sent as inputs to the next layer to undergo the aforementioned blocks for D times for further optimization, where D indicates the number of point decoder layers. Note that, unlike the regression of location, since the classification of visibility can not be updated layer by layer and the wrong supervision of visibility may affect the prediction of location, we follow [18] to only predict the visibility at the last layer. So for the point query q_t^i, its visibility prediction

$$v_t^i = \texttt{VisibilityClassifier}(f_t^i), \tag{9}$$

where `VisibilityClassifier` is an `MLP` ended with a `Sigmoid` activate function.

Upon the completion of the final decoder layer, the updated content feature, the predicted location, and the predicted visibility of each point query will be sent to subsequent modules to update the trajectory preparing for next window.

3.5 Window Post-processing

Limited by the memory limitation, we can not process all the frames of an arbitrarily long video together in parallel. Following previous works [11, 18, 64], we utilize the sliding window strategy to mitigate this issue. However, as illustrated in Fig. 3, after obtaining the results of each window, our updating and padding extend beyond explicit states, including locations and visibilities, to include content features as well.

Updating and Padding Trajectory. After obtaining the query results of the point queries in each frame of the current window, similar to previous works [11, 18, 64], we update their explicit states into their belonging trajectory. Then based on the assumption that smaller temporal differences correspond to smaller positional differences, we pad the locations of the point queries in the last frame of the window to all corresponding point queries in the subsequent frames to reinitialize their location part.

Updating and Padding Content Features. Although the updating and padding of location transfer temporal information to subsequent windows, the lack of transferring more informative content features loses temporal information. However, transferring the content feature without any limitation will result in a drifting problem during inference because of the inconsistency of the video length during training and inference, which will be further discussed in Sec 4.5.

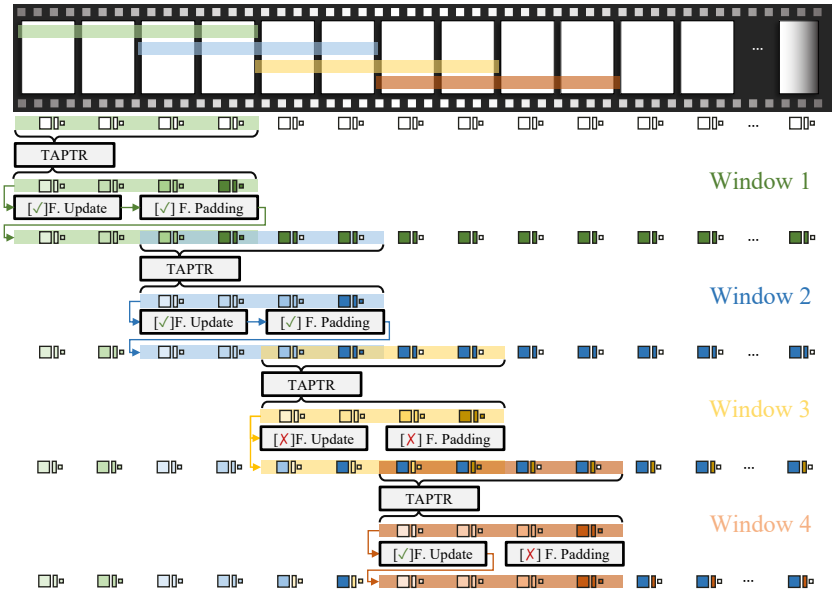

Fig. 3: The overview of sliding window and window updating and padding. "F. Update" indicates the updating of the content feature, and "F. Padding" indicates the padding of the updated feature to the subsequent frames. We use window size 4 and sliding stride 2 for illustration.

To mitigate the feature drifting issue, during training and inference, instead of updating the content feature every time, we employ a random drop strategy to mitigate feature drifting issues. More specifically, during training, as demonstrated in Window 3 of Fig. 3, we randomly disable feature updating, and feature padding is also disabled accordingly. The drop off of feature updating forces the network to handle the cases whether the content feature has been updated or not adaptively. During inference, as demonstrated in Window 4 of Fig. 3, we keep feature updating enabled so that the information from the last window can be at least transferred to the next one. But we drop off the feature padding in a dynamic frequency according to the length of the whole video, to mitigate the accumulation of drifting in the point queries' content feature.

Updating Cost Volume. To ensure the consistency of the optimization target during the multi-layer refinement of our point decoder, we do not update the cost volume in the decoder. This strategy also keeps our decoder clean and concise. Since there is an overlap between two adjacent windows, frames within the overlap will be processed twice by the network. If the content feature of a point query is updated after the current window, we will update its corresponding cost volumes at the beginning of the next window. Compared to previous methods that update the cost volume at each layer, the content features at this stage are more stable, resulting in higher-quality updated cost volumes.

3.6 Full Sequence Multi-Layer Loss

Instead of computing loss in every window, after obtaining the whole location sequence and visibility sequence of one point, we calculate the `L1` loss for the location sequence and the cross entropy loss for the visibility sequence without bells and whistles. Note that, similar to the auxiliary loss in the original DETR, we also maintain a location sequence for each layer of the point decoder. Since we only predict visibility at the last layer of the point decoder, the loss function can be formulated as

$$\texttt{Loss} = \left(\omega_V \texttt{CE}(V, \tilde{V}) + \sum_{d=1}^{D} \omega_L \texttt{L1}(L_d, \tilde{L}) \right) / N, \tag{10}$$

where `CE` shorts for the cross entropy loss, D and d are the number of decoder layers and the index of the point decoder layer respectively, L_d indicates the predicted location sequences of all tracking points from the d-th layer, V represents the predicted visibility sequence of all points from the final layer, N indicates the number of points. $\tilde{L}$ and $\tilde{V}$ are the ground truth location and visibility sequences of all tracking points, respectively, ω_L and ω_V are the weights for the supervision of location and visibility.

4 Experiments

We conduct extensive experiments on the challenging TAP-Vid benchmark [5] to verify the performance of TAPTR. Abundant ablation studies are also provided to analyze the effectiveness of each component in TAPTR, reflecting the adaptability of the DETR-like framework to the TAP task.

4.1 Datasets and Evaluation

Datasets. For training, we follow previous methods [18] to train our model on the TAP-Vid-Kubric dataset. TAP-Vid-Kubric is a synthetic dataset consisting of 11,000 videos with 24 frames. Each video in TAP-Vid-Kubric is generated by Kubric Engine [9] simulating the process of a set of rigid objects falling from the air and then dispersing upon impact. Each video contains annotations for tracking 2048 points. During training, we resize the resolution of the video to 512×512 and randomly sample 700-800 points. We evaluate our method on the TAP-Vid benchmark, which contains three subsets. The first is TAP-Vid-DAVIS, which contains 30 challenging videos captured from various real scenarios with complex motion and large changes in object scale. TAP-Vid-DAVIS has about 2000 frames and 650 annotations for point tracking in total. The second is TAP-Vid-RGB-Stacking. Although it is a synthetic dataset, the objects in this dataset are usually texture-less, making it hard to track. The third one is TAP-Vid-Kinetics, which contains over 1000 labeled videos collected from YouTube. Note that, for a fair comparison, when evaluating the benchmark, we will also downsample the video to 256×256 at first.

Evaluation Protocol and Metrics The TAP-Vid benchmark provides a comprehensive evaluation protocol along with metrics. To accommodate both the online tracker and offline trackers, TAP-Vid provides two evaluation modes. In

Method	PPS	DAVIS			DAVIS-S			RGB-Stacking			Kinetics		
		AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA
COTR [17]	–	–	–	–	35.4	51.3	80.2	–	–	–	–	–	–
Kubric-VFS-Like [9]	–	–	–	–	33.1	48.5	79.4	–	–	–	–	–	–
RAFT [45]	–	–	–	–	30.0	46.3	79.6	–	–	–	–	–	–
PIPs [11]	–	–	–	–	42.0	59.4	82.1	–	–	–	31.7	53.7	72.9
TAP-Net [5]	–	36.0	52.9	80.1	38.4	53.1	82.3	53.5	68.1	86.3	38.5	54.4	80.6
MFT [33]	–	47.3	66.8	77.8	56.1	70.8	86.9	–	–	–	39.6	60.4	72.7
TAPIR [7]	–	56.2	70.0	86.5	61.3	73.6	88.8	55.5	69.7	**88.0**	**49.6**	64.2	85.0
OmniMotion [51]	–	52.7	67.5	85.3	51.7	67.5	85.3	–	–	–	–	–	–
DINO-Tracker [46]	–	–	–	–	62.3	78.2	87.5	–	–	–	–	–	–
CoTracker-Single [18]	–	60.6	75.4	89.3	64.8	79.1	88.7	–	–	–	48.7	64.3	**86.5**
CoTracker2-All [18]	15.7	60.7	75.7	88.1	–	–	–	–	–	–	–	–	–
CoTracker2-Single [18]	0.8	62.2	75.7	89.3	65.9	**79.4**	89.9	–	–	–	–	–	–
BootsTAP† [6]	–	61.4	74.0	88.4	66.4	78.5	90.7	–	–	–	54.7	68.5	86.3
Ours	**20.4**	**63.0**	**76.1**	**91.1**	**66.3**	79.2	**91.0**	**60.8**	**76.2**	87.0	49.0	**64.4**	85.2

Table 1: Comparison of TAPTR with prior methods on TAP-Vid. Points-Per-Second (PPS) indicates how many points can be tracked across the whole video per second on DAVIS dataset on average. Note that, BootsTAP† introduces extra 15M video clips from publicly accessible videos for training.

the "First" mode, the tracking of a point starts from the first frame when it is visible. In the "Strided" mode, the tracking of a point starts from the 5-th, 10-th, 15-th, ... frame, as long as it is visible in these frames. The "Strided" mode requires the tracker to track along bi-directions. TAP-Vid benchmark provides three metrics to evaluate TAP methods. Occlusion Accuracy (OA) is used to evaluate the visibility prediction. $< \delta^x_{avg}$ is the average location precision for tracking points at thresholds of 1,2,4,8,16 pixels. Average Jaccard (AJ) is a comprehensive metric reflecting the accuracy of both location and visibility.

4.2 Implementation Details

We use ResNet50 [12] as our backbone and employ two layers of transformer encoder with deformable attention [65], and six layers of transformer decoder in default. We use the AdamW [66] optimizer and EMA [19] to train our model on 8 NVIDIA A100 for about 36,000 iterations with a learning rate of 2e-4. We accumulate gradients four times using the gradient accumulation to approximate the gradients similar to a batch size of 32. We follow previous works [18] to set the window size as 8 and the stride as 4 for window sliding.

4.3 Comparison with the State of the Arts

We evaluate TAPTR on the TAP-Vid [5] benchmark to show its superiority. As shown in Table 1, TAPTR shows significant superiority compared with previous SoTA methods across the majority of metrics. Note that, The concurrent work BootsTAP [6] is trained with additional real-world data. To evaluate the tracking speed of different methods fairly, we compare the Point Per Second (PPS), which is the average number of points that a tracker can track across the entire video per second on the DAVIS dataset in the "First" mode. The results show the speed advantage of TAPTR.

Row	Small Temp.	Trans. Enc.	Self Att.	Temp. Att.	C. V.	C. Attn.	Res. Up.	Win. Up.	AJ	$< \delta^x_{avg}$	OA
1	✓	✓	✓	✓	✓	✓	✓	✓	**63.0**	**76.1**	**91.1**
2	✗	✓	✓	✓	✓	✓	✓	✓	61.9	75.4	90.3
3	✗	✗	✓	✓	✓	✓	✓	✓	60.9	75.2	88.9
4	✗	✗	✗	✓	✓	✓	✓	✓	58.4	72.1	88.3
5	✗	✗	✗	✗	✓	✓	✓	✓	51.6	66.7	84.5
6	✗	✗	✗	✗	✗	✓	✓	✓	46.8	61.3	82.4
7	✗	✗	✗	✗	✓	✗	✓	✓	50.0	65.0	83.4
8	✗	✗	✗	✗	✗	✓	✗	✓	45.1	60.0	82.4
9	✗	✗	✗	✗	✗	✓	✗	✗	41.8	56.9	79.4

Table 2: Ablation study of the key components in our method on DAVIS dataset. "Small Temp.", "Trans. Enc", "Self. Att.", "Temp. Att.", "C. V.", and "C. Attn." are short for "Small Temperature", "Transformer Encoder", "Self Attention", "Temporal Attention", "Cost Volume", and "Cross Attention", respectively. "Res. Up." indicates the residual updating of the point queries' content feature within the decoder, "Win. Up." indicates the updating and padding of the content feature between windows.

4.4 Ablation of Key Components

As shown in Table 2, we provide extensive ablations to verify the effectiveness of each key component in TAPTR, providing references for future work.

Self Attention and Temperature in Positional Embedding. Compared to the default large temperature in the positional encoding [61] of self-attention within the context of object detection, reducing the temperature, especially 100 times, brings about 1.1 AJ improvement (Row 1 vs. Row 2). After further dropping the Self-Attention, there will also be a drop of 2.5 AJ (Row 3 vs. Row 4). These two experiments reflect the importance of establishing connections among points and the validity of leveraging positional encoding to allocate more attention to the nearby points.

Transformer Encoder. After removing the Transformer Encoder, the comprehensive metric of AJ drops by about 1.0 (Row 2 vs. Row 3). A closer inspection reveals that the main performance loss stems from the accuracy of visibility estimation, indicating that increasing the quality and the receptive field of image feature maps gives a better understanding of the relationships between objects in the scene.

Temporal Information. After removing the temporal attention from our decoder and the feature updating between every two windows, a significant drop of 6.8 AJ occurs (Row 4 vs. Row 5). If we further drop the updating of content feature between windows, there will be an additional drop of about 3.3 AJ (Row 8 vs. Row 9). The drops indicate the importance of temporal information. At the same time, removing temporal attention within the sliding window leads to a larger performance drop, indicating that short-term temporal information should be more important.

Cost Volume Aggregation. Since the cost volume provides a basic visual similarity between the tracking point and the original image, removing the cost volume from our decoder results in a significant drop of 4.8 AJ (Row 5 vs. Row 6), which aligns with the importance of cost volume in previous works.

Row	Update-Train	Pad-Train	Update-Inference	Pad-Inference	DAVIS		
					AJ	$< \delta^x_{avg}$	OA
1	✗	✗	✗	✗	62.0	75.2	89.7
2	✓	✓	✓	✓	54.7	73.0	77.3
3	Random Drop	✓	✗	✗	55.9	71.7	85.3
4	Random Drop	✓	Gap	✓	62.5	75.5	90.9
5	Random Drop	✓	✓	Gap	**63.0**	**76.1**	**91.1**

Table 3: Ablation study of the content feature updating between windows. "Gap" here indicates dynamically gap the feature updating or padding.

Cross Attention. The dropping of cross-attention will result in a drop of 1.6 AJ (Row 5 vs. Row 7), indicating the importance of supplementing detailed visual information to cost volume from cross-attention. Note that, since cross-attention is considered as a basic perception of images in the transformer decoder, we keep cross-attention in the following ablations.

Residual Updating. The initial content feature of a tracking point provides a strong prior, which conveys the specific information of the point to be detected in every frame. Since the updating within the decoder is not stable enough, updating the content feature between every two decoder layers as in the original DETR may bring in noise. Replacing our residual updating with the original one results in a drop of 1.7 AJ (Row 6 vs. Row 8).

4.5 Ablation of Feature Updating Strategy Between Windows

As shown in Table 3, if we directly update our content feature without limitation, there will be a critical drifting problem (Row 1 vs. Row 2). As described in Sec. 3.5, during training we randomly drop off the updating of content features for the tracking points with a probability of 0.6. During inference, due to the length of every training video being 24, instead of random updating, we update the content feature every $T/24$ windows and gap the updating of the intermediate windows to ensure stability. As shown in Row 4 of Table 3, although this strategy loses some temporal information, it still works. To reserve more temporal information, we keep the feature updating open but drop the feature padding with the same gaps. Compared with the direct drop off of feature updating, this strategy at least keeps the temporal information between every two windows, and thus obtains the best performance as shown in Row 5.

4.6 Ablation of Cost Volume Updating

To verify the advantage of our cost volume updating approach, we conduct ablation studies on the updating frequency. As shown in Table 6, if we update the cost volume after every iteration of the decoder as in previous methods [18], there will be a decline of about 1.1 AJ on the DAVIS dataset and 2.8 AJ on the RGB-Stacking dataset. On the other hand, if we keep the cost volume never updated, although achieving comparable performance on the DAVIS dataset, the performance on the RGB-Stacking dataset still decreases by about 2.5 AJ. This is reasonable because, in this case, the cost volume can not benefit from the long temporal information, leading to worse performance on longer videos.

Supervision	AJ	$< \delta^x_{avg}$	OA
Last Layer Only	55.4	70.0	87.3
All Layers	**63.0**	**76.1**	**91.1**

Table 4: Ablation study of the multi-layer losses.

# Decoder	AJ	$< \delta^x_{avg}$	OA
2	58.2	72.3	89.0
4	61.6	75.0	89.8
6	**63.0**	**76.1**	**91.1**

Table 5: Ablation of decoder layer.

Updating Freq.	DAVIS AJ	DAVIS $< \delta^x_{avg}$	DAVIS OA	RGB-Stacking AJ	RGB-Stacking $< \delta^x_{avg}$	RGB-Stacking OA
Per Iter.	61.9	75.2	90.7	58.0	75.6	84.0
No Update	63.0	76.1	90.9	58.3	76.0	84.1
Per Wind.	**63.0**	**76.1**	**91.1**	**60.8**	**76.2**	**87.0**

Table 6: Ablation study of cost volume updating frequency.

4.7 Ablation of Decoder Layer Number and Multi-layer Supervision

Consistent with the conclusions drawn from previous DETR-based object detection methods [3, 23, 26, 61] and multi-layer refinement-based TAP methods [11, 18, 64], as shown in Table 5, as we increase the number of decoder layers, the number of refinement increases and the performance improves accordingly. However, if we only increase the number of decoder layers without supervising the output of each layer, the performance is still poor and even worse than the one with only 2 decoder layers, as shown in Table 4. This indicates that, if not fully supervised, the multi-layer refinement brings a negative impact instead.

5 Visualization

As shown in Fig. 4, when the dog turns around, CoTracker shows a significant drifting, where the tracking result shifts from the right side to the top of the dog. In contrast, TAPTR tracks stably even when the tracking target is occluded. For more comparisons, fancy visualizations, and corresponding videos, please refer to Sec. **??** and Sec. **??** in the appendix.

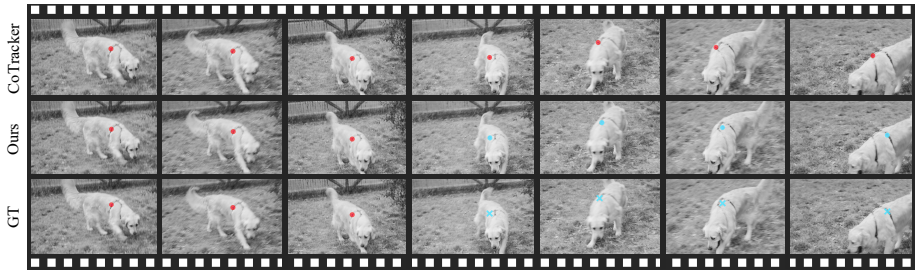

Fig. 4: Red and blue indicate visible and occluded respectively. We manually supplement the ground truth location of invisible points with blue crosses for better comparison. Best view in electronic version.

6 Conclusion

In this paper, we have proposed a conceptually simple yet effective method for tracking any point task. With the help of our designs in mitigating feature drifting we obtain SoTA performance while demonstrating an advantage in inference speed. We also conduct extensive ablations to provide references for future work.

Limitations and Future work. Due to the difficulty in annotating in the TAP task, most training data currently used is synthetic. How to further leverage detection and segmentation annotations to assist TAP tasks is an interesting problem that remains to be explored in our future work. At the same time, as there have been many studies to reduce the cost of self-attention to near linear, how to boost TAPTR with these works could also be explored in future work.

References

1. Black, M., Anandan, P.: A Framework for the Robust Estimation of Optical Flow. In: 1993 (4th) International Conference on Computer Vision (Dec 2002)
2. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: combining local and global optic flow methods. International Journal of Computer Vision,International Journal of Computer Vision (Feb 2005)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-End Object Detection with Transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
4. Chen, L.H., Zhang, J., Li, Y., Pang, Y., Xia, X., Liu, T.: HumanMAC: Masked Motion Completion for Human Motion Prediction. arXiv preprint arXiv:2302.03665 (2023)
5. Doersch, C., Gupta, A., Markeeva, L., Recasens, A., Smaira, L., Aytar, Y., Carreira, J., Zisserman, A., Yang, Y.: TAP-Vid: A Benchmark for Tracking Any Point in a Video. Advances in Neural Information Processing Systems **35**, 13610–13626 (2022)
6. Doersch, C., Yang, Y., Gokay, D., Luc, P., Koppula, S., Gupta, A., Heyward, J., Goroshin, R., Carreira, J., Zisserman, A.: BootsTAP: Bootstrapped Training for Tracking-Any-Point. arXiv preprint arXiv:2402.00847 (2024)
7. Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., Zisserman, A.: TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10061–10072 (2023)
8. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P.v.d., Cremers, D., Brox, T.: FlowNet: Learning Optical Flow with Convolutional Networks. In: 2015 IEEE International Conference on Computer Vision (ICCV) (Dec 2015)
9. Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D.J., Gnanapragasam, D., Golemo, F., Herrmann, C., et al.: Kubric: A scalable dataset generator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3749–3761 (2022)
10. Guler, R.A., Neverova, N., Kokkinos, I.: DensePose: Dense Human Pose Estimation in the Wild. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (Jun 2018)
11. Harley, A.W., Fang, Z., Fragkiadaki, K.: Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. In: European Conference on Computer Vision. pp. 59–75. Springer (2022)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Horn, B.K., Schunck, B.G.: Determining Optical Flow. Artificial Intelligence p. 185–203 (Aug 1981)
14. Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K.C., Qin, H., Dai, J., Li, H.: FlowFormer: A Transformer Architecture for Optical Flow. In: European conference on computer vision. pp. 668–685. Springer (2022)
15. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jul 2017)

16. Jiang, S., Lu, Y., Li, H., Hartley, R.: Learning Optical Flow from a Few Matches. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2021)
17. Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., Yi, K.M.: COTR: Correspondence Transformer for Matching Across Images. In: ICCV (2021)
18. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: CoTracker: It is Better to Track Together. arXiv preprint arXiv:2307.07635 (2023)
19. Klinker, F.: Exponential moving average versus moving exponential average. Mathematische Semesterberichte **58**, 97–107 (2011)
20. Li, B., Sun, Y., Jin, X., Zeng, W., Zhu, Z., Wang, X., Zhang, Y., Okae, J., Xiao, H., Du, D.: StereoScene: BEV-Assisted Stereo Matching Empowers 3D Semantic Scene Completion. arXiv preprint arXiv:2303.13959 (2023)
21. Li, F., Jiang, Q., Zhang, H., Ren, T., Liu, S., Zou, X., Xu, H., Li, H., Li, C., Yang, J., Zhang, L., Gao, J.: Visual In-Context Prompting (2023)
22. Li, F., Zeng, A., Liu, S., Zhang, H., Li, H., Zhang, L., Ni, L.M.: Lite DETR : An Interleaved Multi-Scale Encoder for Efficient DETR (2023)
23. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: DN-DETR: Accelerate DETR Training by Introducing Query DeNoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13619–13627 (2022)
24. Li, H., Zhang, H., Zeng, Z., Liu, S., Li, F., Ren, T., Zhang, L.: DFA3D: 3D Deformable Attention For 2D-to-3D Feature Lifting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6684–6693 (2023)
25. Liang, Z., Guo, Y., Feng, Y., Chen, W., Qiao, L., Zhou, L., Zhang, J., Liu, H.: Stereo Matching Using Multi-Level Cost Volume and Multi-Scale Feature Constancy. IEEE transactions on pattern analysis and machine intelligence **43**(1), 300–315 (2019)
26. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR. arXiv preprint arXiv:2201.12329 (2022)
27. Liu, S., Ren, T., Chen, J., Zeng, Z., Zhang, H., Li, F., Li, H., Huang, J., Su, H., Zhu, J., Zhang, L.: Detection Transformer with Stable Matching (2023)
28. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., Zhang, L.: Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection (2023)
29. Liu, Z., Li, Y., Okutomi, M.: Global Occlusion-Aware Transformer for Robust Stereo Matching. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3535–3544 (2024)
30. Lu, S., Chen, L.H., Zeng, A., Lin, J., Zhang, R., Zhang, L., Shum, H.Y.: Humantomato: Text-aligned whole-body motion generation. arXiv preprint arXiv:2310.12978 (2023)
31. Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C.: TrackFormer: Multi-Object Tracking with Transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8844–8854 (2022)
32. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional DETR for Fast Training Convergence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3651–3660 (2021)
33. Neoral, M., Šerỳch, J., Matas, J.: MFT: Long-Term Tracking of Every Pixel. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6837–6847 (2024)

34. Ning, G., Pei, J., Huang, H.: LightTrack: A Generic Framework for Online Top-Down Human Pose Tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 1034–1035 (2020)
35. Pan, L., Wang, J., Huang, B., Zhang, J., Wang, H., Tang, X., Wang, Y.: Synthesizing Physically Plausible Human Motions in 3D Scenes (2023)
36. Ren, T., Liu, S., Li, F., Zhang, H., Zeng, A., Yang, J., Liao, X., Jia, D., Li, H., Cao, H., Wang, J., Zeng, Z., Qi, X., Yuan, Y., Yang, J., Zhang, L.: detrex: Benchmarking Detection Transformers (2023)
37. Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., Zeng, Z., Zhang, H., Li, F., Yang, J., Li, H., Jiang, Q., Zhang, L.: Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks (2024)
38. Ren, T., Yang, J., Liu, S., Zeng, A., Li, F., Zhang, H., Li, H., Zeng, Z., Zhang, L.: A Strong and Reproducible Object Detector with Only Public Datasets (2023)
39. Shen, Z., Dai, Y., Rao, Z.: CFNet: Cascade and Fused Cost Volume for Robust Stereo Matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13906–13915 (2021)
40. Shi, X., Huang, Z., Li, D., Zhang, M., Cheung, K.C., See, S., Qin, H., Dai, J., Li, H.: FlowFormer++: Masked Cost Volume Autoencoding for Pretraining Optical Flow Estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1599–1610 (2023)
41. Shi, Y., Wang, J., Jiang, X., Dai, B.: Controllable Motion Diffusion Model. arXiv preprint arXiv:2306.00416 (2023)
42. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (Jun 2018)
43. Sun, P., Cao, J., Jiang, Y., Yuan, Z., Bai, S., Kitani, K., Luo, P.: Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20993–21002 (2022)
44. Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., Luo, P.: TransTrack: Multiple Object Tracking with Transformer. arXiv preprint arXiv:2012.15460 (2020)
45. Teed, Z., Deng, J.: RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 402–419. Springer (2020)
46. Tumanyan, N., Singer, A., Bagon, S., Dekel, T.: DINO-Tracker: Taming DINO for Self-Supervised Point Tracking in a Single Video. arXiv preprint arXiv:2403.14548 (2024)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention Is All You Need. Advances in neural information processing systems **30** (2017)
48. Vendrow, E., Le, D.T., Cai, J., Rezatofighi, H.: JRDB-Pose: A Large-scale Dataset for Multi-Person Pose Estimation and Tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4811–4820 (2023)
49. Wang, J., Zhong, Y., Dai, Y., Zhang, K., Ji, P., Li, H.: Displacement-Invariant Matching Cost Learning for Accurate Optical Flow Estimation. Cornell University - arXiv,Cornell University - arXiv (Oct 2020)
50. Wang, M., Tighe, J., Modolo, D.: Combining detection and tracking for human pose estimation in videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11088–11096 (2020)

51. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19795–19806 (2023)
52. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor DETR: Query Design for Transformer-Based Object Detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 2567–2575 (2022)
53. Xu, G., Wang, X., Ding, X., Yang, X.: Iterative Geometry Encoding Volume for Stereo Matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21919–21928 (2023)
54. Xu, H., Yang, J., Cai, J., Zhang, J., Tong, X.: High-Resolution Optical Flow from 1D Attention and Correlation. Cornell University - arXiv,Cornell University - arXiv (Apr 2021)
55. Xu, J., Ranftl, R., Koltun, V.: Accurate Optical Flow via Direct Cost Volume Processing. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jul 2017)
56. Xu, Y., Ban, Y., Delorme, G., Gan, C., Rus, D., Alameda-Pineda, X.: TransCenter: Transformers with Dense Representations for Multiple-Object Tracking. IEEE transactions on pattern analysis and machine intelligence **45**(6), 7820–7835 (2022)
57. Yang, J., Zeng, A., Zhang, R., Zhang, L.: UniPose: Detecting Any Keypoints. arXiv preprint arXiv:2310.08530 (2023)
58. Yuan, Y., Song, J., Iqbal, U., Vahdat, A., Kautz, J.: PhysDiff: Physics-Guided Human Motion Diffusion Model. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16010–16021 (2023)
59. Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., Wei, Y.: MOTR: End-to-End Multiple-Object Tracking with Transformer. In: European Conference on Computer Vision. pp. 659–675. Springer (2022)
60. Zhang, F., Woodford, O.J., Prisacariu, V., Torr, P.H.S.: Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (Oct 2021)
61. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L.M., Shum, H.Y.: DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. arXiv preprint arXiv:2203.03605 (2022)
62. Zhao, H., Zhou, H., Zhang, Y., Chen, J., Yang, Y., Zhao, Y.: High-Frequency Stereo Matching Network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1327–1336 (2023)
63. Zhao, S., Zhao, L., Zhang, Z., Zhou, E., Metaxas, D.: Global Matching with Overlapping Attention for Optical Flow Estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17592–17601 (2022)
64. Zheng, Y., Harley, A.W., Shen, B., Wetzstein, G., Guibas, L.J.: PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19855–19865 (2023)
65. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable Transformers for End-to-End Object Detection. arXiv preprint arXiv:2010.04159 (2020)
66. Zhuang, Z., Liu, M., Cutkosky, A., Orabona, F.: Understanding AdamW through Proximal Methods and Scale-Freeness. arXiv preprint arXiv:2202.00089 (2022)