TAPTRv3: Spatial and Temporal Context Foster Robust Tracking of Any Point in Long Video

Jinyuan Qu[1,3] * **Hongyang Li**[2,3] * **Shilong Liu**[1,3]
Tianhe Ren[3] **Zhaoyang Zeng**[3] **Lei Zhang**[3] †

[1]Tsinghua University.
[2]South China University of Technology.
[3]International Digital Economy Academy (IDEA).
taptr.github.io

Abstract

In this paper, we present TAPTRv3, which is built upon TAPTRv2 to improve its point tracking robustness in long videos. TAPTRv2 is a simple DETR-like framework that can accurately track any point in real-world videos without requiring cost-volume. TAPTRv3 improves TAPTRv2 by addressing its shortage in querying high quality features from long videos, where the target tracking points normally undergo increasing variation over time. In TAPTRv3, we propose to utilize both spatial and temporal context to bring better feature querying along the spatial and temporal dimensions for more robust tracking in long videos. For better spatial feature querying, we present Context-aware Cross-Attention (CCA), which leverages surrounding spatial context to enhance the quality of attention scores when querying image features. For better temporal feature querying, we introduce Visibility-aware Long-Temporal Attention (VLTA) to conduct temporal attention to all past frames while considering their corresponding visibilities, which effectively addresses the feature drifting problem in TAPTRv2 brought by its RNN-like long-temporal modeling. TAPTRv3 surpasses TAPTRv2 by a large margin on most of the challenging datasets and obtains state-of-the-art performance. Even when compared with methods trained with large-scale extra internal data, TAPTRv3 is still competitive.

1. Introduction

Localizing points across different frames in a video is a long-standing problem [33]. Recently, with the growing demand for the trajectory and visibility information of arbitrary points in videos for various down-stream tasks, such as video editing [14], SLAM [40], and manipulation [43], the Tracking Any Point (TAP) task has gradually regained attention [7, 12, 20, 25, 26, 30, 41, 45, 46].

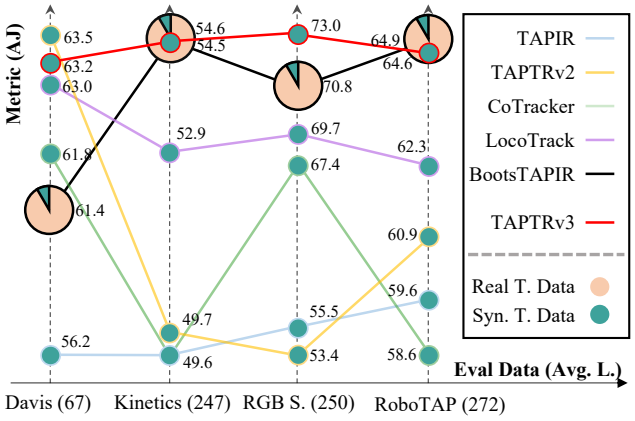


Figure 1. **Comparison of TAPTRv3 with prior methods.** Evaluation on four challenging test sets, where 'Avg. L.' denotes average video length. Circle colors represent training data composition, 'Real T. Data' and 'Syn. T. Data' denote the real and synthetic data, larger circle indicates more data. Results show TAPTRv3 achieves the best performance on most datasets. Even with only 11K synthetic training data, TAPTRv3 remains competitive against BootsTAPIR that is trained on 15M additional real data.

To solve this problem, some methods try to construct a 4D field [45] and track points in the constructed 3D space while also achieving 3D scene perception. Though promising, such methods are normally not general and have inferior performance. By contrast, more methods attempt to solve the TAP task directly from a 2D perspective [7, 12, 20, 25, 26, 41]. Some of these methods follow the traditional optical flow estimation pipeline, RAFT [39] more specifically, and highly rely on cost-volume [48] to perform point tracking like sparse optical flow. Although such methods have achieved im-

*Equal contribution, random listing order. Work done by Jinyuan Qu and Hongyang Li during an internship at IDEA Research.
†Corresponding author.

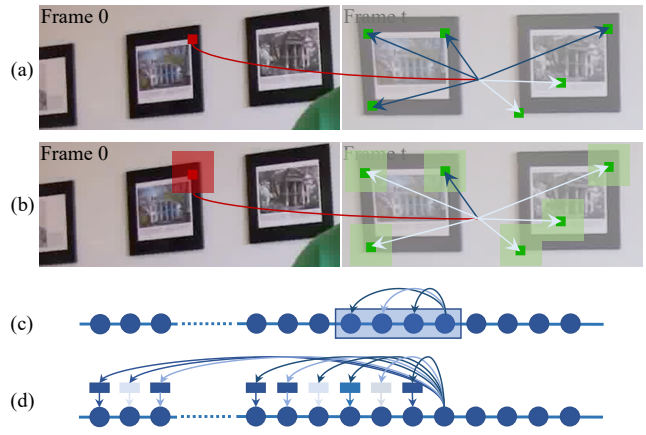


Figure 2. **Illustration of our key contributions.** The darker arrow indicates larger attention weight. (a) Point-level feature in TAPTRv2 causes ambiguous attention in cross-attention because of the large variation and distractions. The red point is the tracking point, the green points are sampling points in cross-attention. (b) Patch-level spatial context features help eliminate distractions in TAPTRv3. (c) TAPTRv2 limits temporal attention in a small window. (d) TAPTRv3 extends temporal attention to arbitrary length while utilizing visibility to further adjust the temporal attention weights. Rectangles indicate visibility predictions, the darker the larger.

pressive performance, the computation of cost-volume is resource-consuming, especially when the number of points, the length of videos, or the video resolution increases.

Inspired by recent visual prompt-based detection [17, 24], TAPTR [26] proposes a DEtection-TRansformer (DETR)-like framework, which regards each tracking point as a point query and addresses the TAP task from the perspective of 2D detection. TAPTRv2 [25] further improves TAPTR by eliminating the cost-volume as it will contaminate the feature of point queries. To compensate for the role of cost-volume for point localization via self-correlation, TAPTRv2 proposes an attention-based position update (APU) operation that utilizes key-aware deformable attention [23] to compare a query point with a set of surrounding sampling points and find a better position. With this improvement, TAPTRv2 leads to both a simpler framework and a better performance.

However, we find that TAPTRv2's performance in long videos is still not satisfying, due to its shortage of feature querying in both spatial and temporal dimensions in long videos, in which the target tracking points normally undergo increasing variation over time. In the spatial dimension, the key-aware deformation attention used in TAPTRv2 is essentially to compare a query point with a set of surrounding sampling points using point-level features. Such point features are generally too local and often lead to ambiguous matching results due to the existence of repetitive patterns and textureless regions in images (See Fig. 2 (a)). In the temporal dimension, the RNN-like long-temporal modeling in TAPTRv2 often suffers from the drifting problem, as the feature of a tracking point may be gradually affected by ambiguous surrounding features and unknown occlusion over time. Moreover, the current training and testing sets have a large discrepancy in terms of video length[1]. Excessive feature updates in long videos during testing further exacerbate the feature drifting problem. In our study, we also observe the existence of scene cuts in many videos. While such videos are the result of artificial editing, they are quite prevalent in public datasets [2]. The lack of global matching in TAPTRv2 makes it hard to reestablish tracking effectively when a scene cut occurs with sudden large motions.

With these insights, we propose to enhance the feature querying ability of TAPTRv3 in both spatial and temporal dimensions. For *spatial feature querying*, we develop a Context-aware Cross-Attention (CCA) operation, which replaces the point-level feature comparison employed in key-aware deformable attention with patch-level feature comparison. As illustrated in Fig. 2 (b), such patch-level similarities are more capable of eliminating distractions, making them a more reliable choice for attention weights and leading to a better perception of 2D image space. For *temporal feature querying*, to address the drifting issue, we discard the RNN-like long-temporal modeling in TAPTRv2. Instead, for any tracking point, we resort to the initial feature sampled from its starting frame where it is specified, as this feature is the most reliable input for point tracking in subsequent frames. To compensate for feature change over time, we introduce a Visibility-aware Long-Temporal Attention (VLTA) operation, which treats the initial feature as a query and performs dense attention over all the past frames to aggregate past feature changes (See Fig. 2 (d)). This not only enables the perception of longer temporal context but also makes TAPTRv3 an online tracker. Meanwhile, recognizing that the target tracking point may be occluded in some frames, we reweight the long-temporal attention weights using the estimated visibility scores in the past frames, directing more attention to frames where the point is visible. This further enhances the feature querying ability along the temporal dimension.

For the scene cut issue, as inspired by previous works [5–7], we introduce a global matching module to reinitialize the point query's positional part for subsequent frames. Note that we only trigger the global matching when detecting a scene cut. This is based on our observation that in regular videos, using the predicted positions from the previous frame as the initial position for the current frame yields better results.

In summary, our contributions are threefold: (1) The primary contribution is the development of a more robust solution for point tracking in long videos, which improves upon TAPTRv2 by leveraging both spatial and temporal context.

[1]The training set consists of short videos, each having fixed 24 frames, whereas the test set contains videos varying from 50 to 1300 frames.

[2]For instance, in TAP-Vid-Kinetics [6], which is one of the challenging test sets, approximately 27% of the videos contain scene cuts.

The two corresponding operations, namely Context-aware Cross-Attention and Visibility-aware Long-Temporal Attention, effectively improve local feature matching and long-term feature updating, enhancing feature querying. (2) To address the scene cut issue, we introduce a global matching module, which is only triggered when a scene cut is detected. This ensures stable tracking on regular videos while being able to quickly reestablish tracking when encountering scene cuts. (3) Extensive experimental results show that TAPTRv3 significantly outperforms TAPTRv2 and achieves state-of-the-art performance on most datasets. Even when compared to models trained on large-scale extra internal real data, TAPTRv3 remains competitive.

2. Related Work

Optical Flow Estimation. Establishing the correspondences of every pixel between two consecutive frames is a long-standing problem. Over the past few decades, extensive research has been dedicated to addressing this issue. Traditional methods [1, 2, 13] use carefully designed descriptors to find correspondences and apply manually designed rules to filter out distractions. DCFlow [48] first demonstrated the feasibility of using the features learned from deep neural network to obtain optical flow estimation through cost-volume that is constructed by calculating 4D correlation, and dominant this field [9, 15, 16, 18, 38, 39, 44, 47–49, 53]. Although the features extracted by deep neural networks are much stronger, the cost volume still suffers ambiguity. To address this, these methods typically feed the cost-volume into convolutions to normalize it based on contextual information. Although the recent optical flow estimation methods [34–36] have shown remarkable results, they still can not handle video data well, especially when the interesting points are occluded.

Tracking Any Point. Influenced by optical flow estimation methods, especially the RAFT [39], most approaches [5–8, 12, 19, 20, 54] follow a similar framework, calculating a cost-volume between the target tracking point and every frame, and then feeding the cost-volume into a transformer [42] to regress the position of the point in each frame. However, the lack of contextual information about the target tracking point, makes the cost-volume a 2D correlation, which is unreliable. Inspired by optical flow methods [39], LocoTrack [5] first points out this limitation, and introduces the local 4D correlation to enhance their performance. In another line, TAPTR [26] and TAPTRv2 [25] address the TAP task from the perspective of detection with their DETR-like [3, 22, 27, 50] framework. With the help of the insight that the cost-volume and attention weights are essentially the same, TAPTRv2 becomes the first method to achieve state-of-the-art performance without requiring the source-consuming cost-volume. However, TAPTRv2 still lacks designs for more challenging long-term tracking, resulting in poor performance in long sequences.

3. Method

Since TAPTRv3 shares a similar architecture with TAPTRv2, to make this section self-contained, we will first provide a brief overview of TAPTRv3's framework and then describe the main improvements of TAPTRv3 in detail.

3.1. Overview

Before describing TAPTRv3, for clarity and without loss of generality, we assume that only a single point is being tracked, starting from the first frame. Given the first frame of the video $\mathbf{I}_0$ and the user-specified point to be tracked at $\mathbf{l}_0 \in \mathbb{R}^2$, TAPTRv3 is expected to detect this point in every subsequent frame $\{\mathbf{I}_t\}_{t=1}^{T-1}$, determining its location $\{\mathbf{l}_t\}_{t=1}^{T-1}$ and visibility $\{\alpha_t\}_{t=1}^{T-1}$, where $\mathbf{l}_t \in \mathbb{R}^2$, $\alpha_t \in [0, 1]$, and T is an integer that indicates the length of the video. As shown in Fig. 3 (a), TAPTRv3 can be roughly divided into Point Query Preparation stage and Sequential Point Tracking stage.

Point Query Preparation. As shown in Fig. 3 (a), after the user has specified the point to be tracked on the 0-th frame $\mathbf{I}_0$ at $\mathbf{l}_0$, the point query preparation stage will sample a point-level feature $\mathbf{f} \in \mathbb{R}^D$ to describe the target tracking point, where D is the number of channels. Following TAPTR and TAPTRv2, the sample is achieved by conducting bilinear interpolation on the $\mathbf{I}_0$'s corresponding image feature map[3] $\mathbf{X}_0 \in \mathbb{R}^{H \times W \times D}$ at $\mathbf{l}_0$, where H and W indicate the height and width of the feature map. To have a more comprehensive description of the point, TAPTRv3 additionally sample N^2 context features $\mathbf{C} \in \mathbb{R}^{N^2 \times D}$ around $\mathbf{l}_0$ in a grid form to describe the point's initial spatial context,

$$\mathbf{C} = \texttt{Bili}\left(\mathbf{X}_0, \mathbf{l}_0 + \mathbf{G}\right), \tag{1}$$

where N is a hyperparameter and is set as 3 by default, $\mathbf{G} \in \mathbb{R}^{N^2 \times 2}$ is the sampling grid, and `Bili` is the bilinear interpolation operation. After that, $\mathbf{f}$ and $\mathbf{l}_0$ will be regarded as the initial content part and the positional part of the target tracking point's corresponding point query. The point query will be sent to the transformer decoder as a query, to detect the target tracking point in subsequent frames in the next Sequential Point Tracking stage.

Sequential Point Tracking. As shown in Fig. 3 (b), when TAPTRv3 receives a new frame, $\mathbf{I}_1$ for example, its corresponding feature map $\mathbf{X}_1$, which is regarded as a set of keys and values, as well as the point query, will be sent to the multi-layer transformer decoder. In each transformer decoder layer, both the content part and the positional part of the point query will be refined by our Visibility-aware Long-Temporal Attention, Context-aware Cross-Attention with APU, Self-Attention, and Feed Forward Network. After the multi-layer refinement. The output refined positional part of the point query $\mathbf{l}_1$ will be regarded as the detection result of the target

[3]For clarity, without loss of generality, we assume that each frame's feature map, obtained from the backbone and transformer encoder, has only one scale.

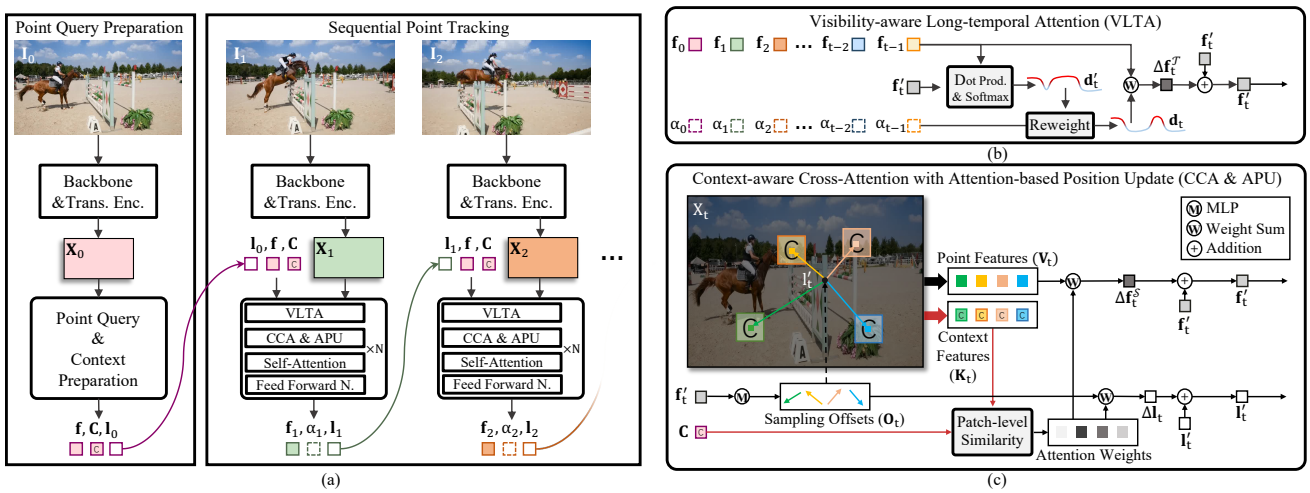


Figure 3. **Overview (a) and core components (b) (c) of TAPTRv3.** After the user has specified the point to track, the point query preparation stage prepares the content feature and spatial context features for this point in the initial frame. When TAPTRv3 receives a new frame, the sequential point tracking stage will regard the content feature and the specified location as the point query, and regard the new frame's corresponding image feature map as a set of keys and values. The points query, keys, and values will be sent to the multi-layer transformer decoder to detect the tracking point in the new frame. The location prediction will further be used to update the point query's positional part, providing a better initial position for tracking in the next frame. For clarity, the global matching module is not plotted.

tracking point in $\mathbf{I}_1$, and the output refined content feature of the point query $\mathbf{f}_1$ will be sent to an MLP-based binary classifier to predict the confidence α_1 that the point is visible in $\mathbf{I}_1$. After that, the position prediction $\mathbf{l}_1$ will be used to update the point query's positional part, providing a better initial position for detecting the target tracking point in the next frame, while the content part remains as the initial content feature $\mathbf{f}$. This process proceeds repeatedly until the end of the video.

3.2. Visibility-aware Long-Temporal Attention

The RNN-like long-temporal modeling in TAPTR and TAPTRv2 can lead to feature drift, primarily due to the excessive feature updates during testing, rooted in the disparity in video lengths between training (24 frames) and testing (ranging from 50 to 1300 frames). Thus, as shown in Fig. 3 (b), we resort to the attention mechanism, for its ability to handle varying token lengths, as demonstrated in modern LLMs [10, 31, 51, 52]. Following modern LLMs, we also utilize the rotary positional embedding [37] to help our long-temporal attention pay more attention to recent frames. More specifically, the long-temporal attention weights of the target tracking point between the t-th frame and all past frames can be first computed as,

$$\begin{aligned} \mathbf{F}_t = [\mathbf{f}_0, \mathbf{f}_1, \cdots, \mathbf{f}_{t-1}]^\top, \mathbf{R}_t = [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{t-1}]^\top, \\ \mathbf{d}'_t = \texttt{SoftMax}\left((\mathbf{F}_t + \mathbf{R}_t) \otimes (\mathbf{f}'_t + \mathbf{r}_t)\right), \end{aligned} \tag{2}$$

where $\mathbf{F}_t \in \mathbb{R}^{t\times D}$ and $\mathbf{R}_t \in \mathbb{R}^{t\times D}$ are the point query's refined content features in the past frames of the t-th frame, and their corresponding rotary positional embeddings, $\mathbf{d}'_t \in \mathbb{R}^t$ is the long-temporal attention distribution (weights), $\otimes$ indicates the matrix multiplication, and $\mathbf{f}'_t \in \mathbb{R}^D$ is the content feature of the point query in the t-th frame that has not been fully refined by decoder. In the first decoder layer, $\mathbf{f}'_t \equiv \mathbf{f}$.

Different from the textual scenarios, since the target tracking point will be occluded sometimes, the refined content features from the frames that the target tracking point is occluded may contribute noise to the long-temporal attention. To prevent being affected by the noise, we utilize the visibility predictions in the past frames $\mathbf{a}_t \in \mathbb{R}^t = [\alpha_0, \alpha_1, \ldots, \alpha_{t-1}]^T$ to reweight the attention distribution, making it visibility-aware. The final visibility-aware attention weights are used to weight-sum their corresponding content features to obtain temporal querying result $\Delta\mathbf{f}_t^{\mathcal{T}} \in \mathbb{R}^D$. The querying result will be further utilized as a residual to update the point query's content feature to complete the VLTA. The process can be formulated as,

$$\begin{aligned} \mathbf{d}_t = \frac{\mathbf{d}'_t \odot \mathbf{a}_t}{\texttt{Sum}(\mathbf{a}_t)}, \Delta\mathbf{f}_t^{\mathcal{T}} = \mathbf{F}_t^\top \otimes \mathbf{d}_t, \\ \mathbf{f}'_t \Leftarrow \texttt{LN}\left(\mathbf{f}'_t + \Delta\mathbf{f}_t^{\mathcal{T}}\right), \end{aligned} \tag{3}$$

where $\mathbf{d}_t \in \mathbb{R}^t$ is the attention distribution that is reweighted by visibility predictions, $\mathbf{f}'_t$ is the output and will be sent to the following modules for further refinement, LN is the Layernorm [21], and $\odot$ is the element-wise multiplication.

3.3. Context-aware Cross-Attention with APU

Unlike object-level DETR-like methods, the content feature of a point query in TAPTRv3 is a point-level feature. This granularity can limit the model's receptive field during cross-attention with the image feature map, often causing ambiguity in attention weights. This issue becomes more serious when the target tracking point undergoes significant varia-

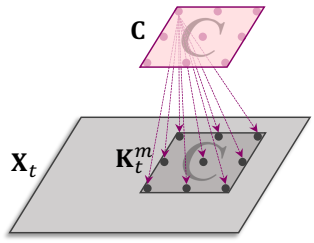


Figure 4. **Illustration of patch-level similarity calculation.**

tions or when the image contains uniform regions or repetitive patterns. Such conditions can lead to noisy querying of spatial features as well as noisy position updates in the cross-attention's belonging APU block. Inspired by previous optical flow estimation methods [39] and the recent LocoTrack [5], we propose to leverage more informative spatial context features to provide the point query with a more comprehensive understanding of its surrounding environment, thus helping eliminate ambiguity.

As illustrated in Fig. 3 (c), different from the vanilla key-aware deformable attention [23], the query's point-level content feature will only be used to predict M sampling offsets $\mathbf{O}_t \in \mathbb{R}^{M\times 2} = \left[\mathbf{o}_t^0, \mathbf{o}_t^1, \ldots, \mathbf{o}_t^{M-1}\right]^\top$ by an MLP. For the corresponding attention weights, we obtain them by leveraging the patch-level context features. More specifically, as shown in Fig. 4, for the m-th sampling point, its corresponding spatial context features in the t-th frame $\mathbf{K}_t^m \in \mathbb{R}^{N^2\times D}$ can be constructed by

$$\mathbf{K}_t^m = \texttt{Bili}\left(\mathbf{X}_t, \mathbf{l}_t' + \mathbf{o}_t^m + \mathbf{G}\right), \tag{4}$$

where $\mathbf{l}_t' \in \mathbb{R}^2$ is the positional part of the point query in the t-th frame that has not been fully refined by decoder layers. In the first decoder layer's CCA module $\mathbf{l}_t' \equiv \mathbf{l}_{t-1}$. After that, the m-th sampling point's corresponding patch-level similarity $w_t^m \in \mathbb{R}$ is calculated as,

$$\begin{aligned} \mathbf{S}_t^m \in \mathbb{R}^{N^2\times N^2} &= \mathbf{C} \otimes \mathbf{K}_t^{m\top}, \\ w_t^m &= \texttt{MLP}(\texttt{Flatten}(\mathbf{S}_t^m)), \end{aligned} \tag{5}$$

where $\mathbf{S}_t^m$ is the intermediate representation of the patch-level similarity, and `Flatten` indicates the flatten operation. By sending the intermediate representation $\mathbf{S}_t^m$ to an MLP, the more fine-grained similarities between every two points in the two patches are comprehensively considered, leading to a high-quality patch-level similarity. The patch-level similarities between the point query and all sampling points $\mathbf{w}_t \in \mathbb{R}^M = [w_t^0, w_t^1, \ldots, w_t^{M-1}]^\top$ will work as attention weights to aggregate their corresponding values $\mathbf{V}_t \in \mathbb{R}^{M\times D}$, obtaining the spatial querying result $\Delta\mathbf{f}_t^{\mathcal{S}} \in \mathbb{R}^D$,

$$\begin{aligned} \mathbf{V}_t = [\mathbf{v}_t^0, \mathbf{v}_t^1, \cdots, \mathbf{v}_t^{M-1}]^\top, \mathbf{v}_t^m = \texttt{Bili}(\mathbf{X}_t, \mathbf{l}_t' + \mathbf{o}_t^m), \\ \Delta\mathbf{f}_t^{\mathcal{S}} = \mathbf{V}_t^\top \otimes \texttt{SoftMax}\left(\mathbf{w}_t/\sqrt{D}\right), \end{aligned} \tag{6}$$

where $\mathbf{V_t}$ are the sampled values of M sampling points on $\mathbf{X}_t$. After that, the spatial querying results will be utilized as a residual to complete the CCA,

$$\mathbf{f}_t' \Leftarrow \texttt{LN}(\mathbf{f}_t' + \Delta\mathbf{f}_t^{\mathcal{S}}), \tag{7}$$

Following TAPTRv2, we also add the attention-based position update (APU) in CCA, to benefit from the attention weights' resilience to the domain gap without contaminating the point query. This process can be formulated as,

$$\begin{aligned} \Delta\mathbf{l}_t = \mathbf{O}_t^\top \otimes \texttt{SoftMax}\left(\texttt{MLP}\left(\mathbf{w}_t\right)/\sqrt{D}\right), \\ \mathbf{l}_t' \Leftarrow \mathbf{l}_t' + \Delta\mathbf{l}_t, \end{aligned} \tag{8}$$

where $\Delta\mathbf{l}_t \in \mathbb{R}^2$ is the additional position update. We follow TAPTRv2, using an MLP to decouple the attention weights used for content and position updates.

3.4. Tracking Reestablish with Global Matching

In general, point motions are smooth in video. However, in long videos, especially in offline scenarios, scene cut often occurs, leading to sudden large motions. Since the initial position of a point query in the current frame is inherited from the previous frame's prediction as we have described in Sec. 3.1, the sudden large motion will cost many frames for TAPTRv3 to catch up with the target tracking point.

Thus, when a scene cut occurs[4] at the t-th frame, we trigger the global matching module to help TAPTRv3 reestablish tracking. Similar to previous methods [5, 7], the global matching module will construct a similarity map $\mathbf{H}_t \in \mathbb{R}^{H\times W}$ between the target tracking point and the current frame's feature map. However, instead of relying on a point-level feature as in previous methods, similar to CCA, we leverage the spatial context features to help improve the accuracy of the similarity map. After that, SoftArgMax is conducted on the similarity map to obtain a final position prediction $\mathbf{l}_t \in \mathbb{R}^2$ that is differentiable. The prediction will be used to replace the unreliable position prediction of the current frame from the transformer decoder. The replacement helps TAPTRv3 to prevent wrong predictions in the subsequent frames. The process can be formulated as,

$$\begin{aligned} \mathbf{H}_t' = \mathbf{X}_t \otimes \mathbf{C}^\top, \mathbf{H}_t = \texttt{Softmax}(\texttt{MLP}(\mathbf{H}_t')), \\ \mathbf{l}_t = \texttt{SoftArgMax}(\mathbf{H}_t), \end{aligned} \tag{9}$$

where $\mathbf{H}_t' \in \mathbb{R}^{H\times W\times N^2}$ is the similarity maps between the initial context features and the current frame's feature map. These maps will be fused through an MLP to obtain the $\mathbf{H}_t$.

4. Experiments

We conduct extensive experiments on multiple challenging evaluation datasets collected from both the real and synthetic world to verify the superiority of TAPTRv3.

4.1. Datasets and Evaluation

Training Data. For fair comparison, following the previous methods [5, 7, 20, 25, 26], we trained our model on the TAP-Vid-Kubric [6] dataset, which consists of 11,000 synthetic

[4] We use the off-the-shelf PySceneDetect to detect the scene cuts.

Method	Training Data	Kinetics AJ	Kinetics $< \delta^x_{avg}$	Kinetics OA	RGB-Stacking AJ	RGB-Stacking $< \delta^x_{avg}$	RGB-Stacking OA	RoboTAP AJ	RoboTAP $< \delta^x_{avg}$	RoboTAP OA	DAVIS AJ	DAVIS $< \delta^x_{avg}$	DAVIS OA
PIPs [12]	FT†	31.7	53.7	72.9	–	59.1	–	–	–	–	42.2	64.8	77.7
PIPs++ [54]	PO†	–	63.5	–	–	58.5	–	–	63.0	–	–	69.1	–
TAP-Net [6]	Kub24	38.5	54.4	80.6	53.5	68.1	86.3	45.1	62.1	82.9	33.0	48.6	78.8
TAPIR [7]	Kub24	49.6	64.2	85.0	55.5	69.7	88.0	59.6	73.4	87.0	56.2	70.0	86.5
CoTracker [20]	Kub24	49.6	64.3	83.3	67.4	78.9	85.2	58.6	70.6	87.0	61.8	76.1	88.3
TAPTR [26]	Kub24	49.0	64.4	85.2	60.8	76.2	87.0	60.1	75.3	86.9	63.0	76.1	91.1
TAPTRv2 [25]	Kub24	49.7	64.2	85.7	53.4	70.5	81.2	60.9	74.6	87.7	**63.5**	75.9	**91.4**
LocoTrack [5]	Kub24	52.9	66.8	85.3	69.7	83.2	89.5	62.3	76.2	87.1	63.0	75.3	87.2
BootsTAPIR† [8]	Kub24+15M	54.6	68.4	86.5	70.8	83.0	89.9	64.9	80.1	86.3	61.4	73.6	88.7
CoTracker3 (online)† [19]	Kub64	54.1	66.6	87.1	71.1	81.9	90.3	60.8	73.7	87.1	64.5	76.7	89.7
CoTracker3 (online)† [19]	Kub64+15K	55.8	68.5	88.3	71.7	83.6	91.1	66.4	78.8	90.8	63.8	76.3	90.2
TAPTRv3 (Ours)	Kub24	**54.5**	**67.5**	**88.2**	**73.0**	**86.2**	**90.0**	**64.6**	**77.2**	**90.1**	63.2	**76.7**	91.0

Table 1. **Comparison of TAPTRv3 with prior methods.** Note that, CoTracker3† is a concurrent work. CoTracker3† and BootsTAPIR† both introduced extra data for training. Specifically, CoTracker3† re-rendered synthetic Kubric videos with length of 64 frames and additionally incorporated 15K real videos. In contrast, BootsTAPIR† trained on an additional 15M real videos. TAPTRv3 obtains state-of-the-art performance on most datasets and remains competitive with methods trained on extra internal data. Training data: (Kub24) Kubric [11] with 24 frames per video, (Kub64) Kubric with 64 frames per video, (PO) PointOdyssey [54], (FT) FlyingThings++ [29].

videos generated by Kubric Engine [11], each containing 24 frames showing 3D rigid objects falling to the ground and bouncing. Each video has 2,048 points sampled on moving objects and backgrounds to be tracked, and their corresponding trajectories are also generated for training. The points to be tracked in the video are occasionally occluded to allow the model to cope with this situation. During training, the resolution of the videos is resized to 512×512 and we randomly select 800 trajectories in each video for efficient training.

Evaluation Data. We follow previous methods to evaluate TAPTRv3 on the challenging TAP-Vid [6] benchmark, which consists of 3 subsets, TAP-Vid-Kinetics, TAP-Vid-DAVIS, and RGB-Stacking. TAP-Vid-DAVIS comprises 30 real-world videos from the DAVIS 2017 validation set [32]. These videos are relatively short, averaging less than 70 frames. TAP-Vid-Kinetics contains 1,144 YouTube videos from the Kinetics-700-2020 validations set [4], with camera shakes, complex environments, and even scene cuts. These videos are relatively long, averaging about 250 frames per video. RGB-Stacking is a synthetic dataset that captures the process of a robotic arm grasping solid-colored blocks, with an average duration of about 250 frames. Although it has a relatively smaller domain gap compared to the training set, which is also a synthetic dataset, the objects in this dataset usually lack texture, making it also difficult to track. In addition, we also evaluated TAPTRv3 on RoboTAP [43], which comprises 265 real-world videos from robotic manipulation tasks. The video length in this dataset varies significantly, with the longest videos reaching up to 1300 frames and the average length of about 270 frames per video.

Evaluation Metrics and Settings. We use the standard metrics proposed in TAP-Vid [6] for evaluation, including three important metrics. Occlusion Accuracy (OA), describes the accuracy of classifying whether a target tracking point is visible or occluded. $< \delta^x_{avg}$, reflecting the average precision of visible points' position at thresholds of 1, 2, 4, 8, and 16 pixels. Average Jaccard (AJ), a comprehensive metric, considers both the precision of position and visibility prediction. Since TAPTRv3 is an online tracker, we use the "First query" mode [6] to evaluate the model, which tracks the target tracking point from the first frame when they are visible until the end of the video. This is much more difficult than the "Strided query" mode for offline trackers. Besides, since the resolution of the input video has a large impact on the final performance, we limit the resolution of the input video to 256×256 for a fair comparison with other methods during evaluation.

4.2. Implementation Detail

Following previous methods [25, 26], we use ResNet-50 as the backbone to extract image features of every frame, 2 encoder layers with deformable attention [55] to further enhance the image features. Benefiting from our improvement, only 4 decoder layers are required to achieve optimal performance. For the supervision of location and visibility prediction, we utilize the L1 loss and binary cross-entropy loss respectively as in previous works. We use the AdamW [28] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and set the weight decay to 1×10^{-4}. We train TAPTRv3 on a cluster of 8 NVIDIA A100 GPUs with a batch size of 8 in total. To make the training process more stable, we accumulate gradients 4 times to approximate a batch size of 32. After the training of TAPTRv3, we freeze it and add the additional global matching for the second stage of training. Since there are only a few parameters to be trained in this stage, it only requires about 5,300 iterations to converge.

For efficiency, in ablation studies in Sec. 4.4, we have a

Row	LTA	Vis. Aware	No Window	No Inv. Sup.	CCA	AJ	$< \delta^x_{avg}$	OA
1	✗	✗	✗	✗	✗	46.0	62.8	83.8
2	✓	✗	✗	✗	✗	48.9	63.0	85.7
3	✓	✓	✗	✗	✗	50.1	63.1	85.9
4	✓	✓	✓	✗	✗	51.1	63.3	85.8
5	✓	✓	✓	✓	✗	51.9	64.8	86.9
6	✓	✓	✓	✓	✓	**52.9**	**65.8**	**87.2**

Table 2. **Ablations on key component.** "LTA" refers to Long-Temporal Attention, "Vis." is short for visibility, "No Window" indicates the removal of the sliding window, and "No Inv. Sup." represents no supervision of the invisible points' positions.

Patch-level Similarity	AJ	$< \delta^x_{avg}$	OA
Element-wise	52.5	65.1	85.7
Every two point	**52.9**	**65.8**	**87.2**

Table 3. **Ablation on patch-level similarity.**

Update Methods	AJ	$< \delta^x_{avg}$	OA
VLTA	52.6	65.3	86.8
MLP	52.5	65.7	**87.3**
No Updates	**52.9**	**65.8**	87.2

Table 4. **Ablation on context features updating.**

few modifications in our experimental settings. We reduce the number of encoder and decoder layers to 1 and 3 respectively, resize the resolution of the input video to 384×384, and also reduce the number of tracking points on each video to 200. The ablation studies are conducted on 4 GeForce RTX3090 GPUs for about 28,000 iterations with randomly sampled half-size training and evaluation sets.

4.3. Comparison with the State of the Arts

We evaluate TAPTRv3 on the TAP-Vid [6] and RoboTAP [43] benchmarks and compare TAPTRv3 with previous methods to demonstrate the superiority of our approach. As shown in Table 1, TAPTRv3 achieves state-of-the-art on most metrics on these four datasets. In particular, on the Kinetics, RGB-Stacking, and RoboTAP subsets, which are composed of long videos, TAPTRv3 shows a large improvement (9.3 AJ on average) compared to TAPTRv2 [25] and surpasses the previous SoTA method significantly by about 2.4 AJ on average. On the DAVIS dataset with shorter video, although it is slightly behind TAPTRv2 on AJ, there is still an improvement in $< \delta^x_{avg}$, indicating better localization capability.

Although Cotracker3 [19] and BootsTAPIR [8] achieve remarkable performance, they both introduced extra internal real-world data for training. Specifically, CoTracker3 re-renders the Kubric training set with a length of 64 frames, which narrows the gap in video length between training and evaluation. Then an additional 15K real-world videos are introduced for fine-tuning. BootsTAPIR trains their model on the original Kubric training set but introduces an extra 15M real-world video clips, which is approximately 1,360 times more than the synthetic data (11K) we use for training. The results show that despite these methods utilizing much more additional data for training, TAPTRv3 still achieves competitive performance.

Note that, for fair comparison, we do not utilize global matching to help reestablish tracking here. When the global matching is also considered, the results are further improved. For more detail please refer to Sec. 4.4.

4.4. Ablation Studies and Analysis

We start our ablation from TAPTRv2 and conduct ablation studies on every key component in TAPTRv3 to validate their effectiveness. We further conduct some more detailed ablations to investigate the best implementation choice. To focus on the ability of TAPTRv3 in handling long videos, we conduct ablations on Kinetics.

Visibility-aware Long-Temporal Attention. We first replace the RNN-like long-temporal modeling with the long-temporal attention. As shown in Table 2, the comparison between Row 2 and Row 1 shows that the replacement provides a significant improvement (2.9 AJ), indicating the superiority of the attention mechanism over RNN in handling varying data length, which aligns with the findings in modern LLMs. Furthermore, the comparison between Row 3 and Row 2 shows that enabling the long-temporal attention to utilize visibility prediction to reduce noises caused by occlusion will further improve the performance by 1.2 AJ. The significant improvements brought by the VLTA module indicate the effectiveness of incorporating richer temporal information.

Discarding Sliding Window. With VLTA, the model captures temporal information from all previous frames, making it redundant to recompute temporal attention within a small window. Therefore, we eliminate the sliding window by reducing the window size from 8 to 1. In this case, the original temporal attention in TAPTRv2 will degenerate to an MLP with residual. We retain this module in the following ablations for fair comparison. As shown in Table 2, the comparison between Row 4 and Row 3 shows a 1.0 AJ improvement. We attribute this to better position initialization. More specifically, initializing the position of the target tracking point in the current frame with the estimation from a more nearby frame simplifies the estimation process, as the distance that the model needs to refine is significantly reduced.

Context-aware Cross-Attention. As shown in Table 2, the comparison between Row 6 and Row 5 shows that the introduction of CCA further brings a significant improvement of 1.0 AJ. The improvement verifies the effectiveness of CCA in improving the robustness of the spatial feature querying.

Ignore the Position Supervision of Invisible Points. It is worth noting that, predicting the position of the target tracking point when it is occluded is an ill-posed problem. Forcing the model to localize the occluded point can destabilize the learning process and may lead the model to learn a bias toward a fixed motion pattern. As shown in Rows 5 and 4 of Table 2, simply ignoring the supervision of invisible points' location predictions results in an improvement of 0.8 AJ.

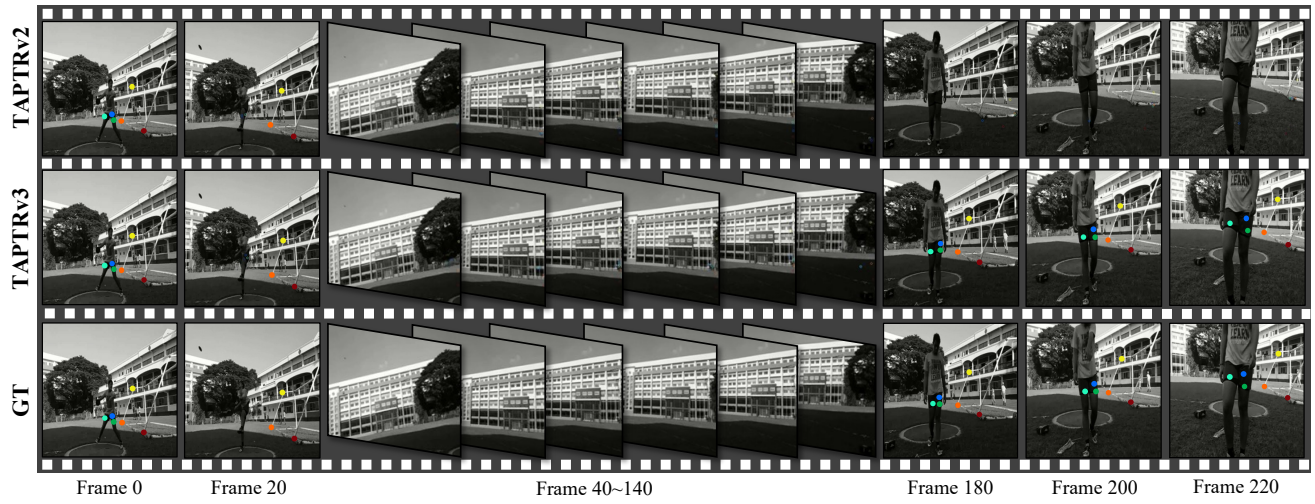


Figure 5. **Visualization of ground truth, TAPTRv3, and TAPTRv2.** In the early stages of tracking, both TAPTRv2 and TAPTRv3 can produce accurate tracking results. As the camera shifts and the tracking points disappear from the video for an extended period (over 100 frames), TAPTRv2 completely loses the tracking, whereas TAPTRv3 maintains stable tracking. Best view in electronic version.

Patch-level Similarity Calculation in CCA. As shown in Table 3, we conduct comparative experiments on different methods to obtain patch-level similarity. The results show that the "Element-wise", which only considers the similarities between two points that are located at the same position in the two patches, is less effective than the "Every two point" strategy that is adopted in our current CCA. We believe this may be because the "Every two point" strategy has an advantage in handling more complex spatial variations, such as rotation, and is more tolerant of the sampling point's location. For more details, please refer to our supplementary.

Spatial Context Updating. As shown in Tabel 4, neither utilizing our VLTA nor an MLP to update the query's context features yields better results. This indicates that maintaining the target tracking points' spatial context throughout the tracking process not only reduces computation costs but also helps spatial feature querying. For more details about this ablation please refer to supplementary.

Ablation on Tracking Reestablish with Global Matching As discussed in Sec. 3.4, TAPTRv3 triggers global matching only when a scene cut is detected. This is because we find that, as shown in Table 5, solely relying on global matching to provide the initial location of the point queries in every frame obtains inferior performance. To quantitatively demonstrate the tracking improvements brought by global matching, we conducted comparative experiments on the Kinetics dataset and its subset, which is composed of videos with scene cuts, accounting for about 27% of the full set. As shown in Table 6, Although the gain from global matching is not substantial on the full set, it brings an average improvement of 0.7 AJ on videos with scene cuts, which is significant. This demonstrates the effectiveness of the global matching module and the mechanism we use to trigger it.

Input Positions of Decoder	AJ	$< \delta^x_{avg}$	OA
Global Matching Calculation	51.0	63.3	84.1
Previous Frame's Prediction	**52.9**	**65.8**	**87.2**

Table 5. **Ablation on decoder input positions.**

Input Positions of Decoder	Dataset	AJ	$< \delta^x_{avg}$	OA
Previous Frame's Prediction	Kin.	54.5	67.5	88.2
Global Matching if S.C.	Kin.	**54.6**	**67.8**	**88.4**
Previous Frame's Prediction	Kin. w/ S.C.	54.4	66.9	86.6
Global Matching if S.C.	Kin. w/ S.C.	**55.1**	**68.1**	**87.4**

Table 6. **Ablation on global matching trigger.** Whether to employ the global matching positions when detecting scene cuts. "Kin." is short for TAP-Vid-Kinetics, and "S.C." indicates scene cuts.

5. Visualization

We visualize the tracking results of TAPTRv2 and TAPTRv3 in a long video (about 250 frames) in Fig. 5. Note that, for a fair comparison, we do not add global matching. As the camera shifts away from the player and pans around for a long period (approximately 100 frames), TAPTRv2 loses track completely. Whereas, TAPTRv3 maintains stable tracking even after the target has disappeared for a long duration without the help of global matching, demonstrating the robustness of TAPTRv3 in tracking any point in long videos. For more visualizations, please refer to our supplementary.

6. Conclusion

In this paper, we have presented TAPTRv3, a strong method for the TAP task. TAPTRv3 improves TAPTRv2 primarily by developing the Context-aware Cross-Attention (CCA) and Visibility-aware Long-Temporal Attention (VLTA) to address the shortage of feature querying. CCA utilizes the spatial

context to help obtain more robust attention weights in cross-attention, leading to a better perception of 2D image space. VLTA replace the RNN-like long-temporal modeling with an attention mechanism to enable the perception of longer temporal context while avoiding the feature drifting issue. VLTA further utilizes the visibilities to improve the quality of long-temporal attention, leading to a better feature querying ability along the temporal dimension. Additionally, TAPTRv3 further improves its performance in long videos by introducing the global matching module with a carefully designed trigger strategy. With the help of these novel designs, TAPTRv3 surpasses TAPTRv2 by a large margin and obtains state-of-the-art performance on multiple challenging datasets. Even when compared with the methods trained on extra internal data, TAPTRv3 remains competitive.

References

[1] M.J. Black and P. Anandan. A Framework for the Robust Estimation of Optical Flow. In *1993 (4th) International Conference on Computer Vision*, 2002. 3

[2] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: combining local and global optic flow methods. *International Journal of Computer Vision,International Journal of Computer Vision*, 2005. 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 3

[4] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 6

[5] Seokju Cho, Jiahui Huang, Jisu Nam, Honggyu An, Seungryong Kim, and Joon-Young Lee. Local All-Pair Correspondence for Point Tracking. In *European Conference on Computer Vision*. Springer, 2024. 2, 3, 5, 6, 12

[6] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A Benchmark for Tracking Any Point in a Video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022. 2, 5, 6, 7

[7] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking Any Point with Per-frame Initialization and Temporal Refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023. 1, 2, 5, 6, 12

[8] Carl Doersch, Yi Yang, Dilara Gokay, Pauline Luc, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped Training for Tracking-Any-Point. *arXiv preprint arXiv:2402.00847*, 2024. 3, 6, 7

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 3

[10] Google. Bard. https://bard.google.com/, 2023. 4

[11] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022. 6

[12] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. In *European Conference on Computer Vision*, pages 59–75. Springer, 2022. 1, 3, 6

[13] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence*, page 185–203, 1981. 3

[14] Jiahui Huang, Leonid Sigal, Kwang Moo Yi, Oliver Wang, and Joon-Young Lee. INVE: Interactive Neural Video Editing. *arXiv e-prints*, pages arXiv–2307, 2023. 1

[15] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A Transformer Architecture For Optical Flow. In *European conference on computer vision*, pages 668–685. Springer, 2022. 3

[16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[17] Qing Jiang, Feng Li, Zhaoyang Zeng, Tianhe Ren, Shilong Liu, and Lei Zhang. T-rex2: Towards Generic Object Detection via Text-visual Prompt Synergy. In *European Conference on Computer Vision*, pages 38–57. Springer, 2025. 2

[18] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning Optical Flow from a Few Matches. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[19] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos. *arXiv preprint arXiv:2410.11831*, 2024. 3, 6, 7

[20] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is Better to Track Together. In *European Conference on Computer Vision*. Springer, 2024. 1, 3, 5, 6

[21] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *ArXiv e-prints*, pages arXiv–1607, 2016. 4

[22] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. DN-DETR: Accelerate DETR Training by Introducing Query DeNoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022. 3

[23] Feng Li, Ailing Zeng, Shilong Liu, Hao Zhang, Hongyang Li, Lei Zhang, and Lionel M Ni. Lite DETR: An Interleaved

Multi-Scale Encoder for Efficient DETR. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18558–18567, 2023. 2, 5

[24] Feng Li, Qing Jiang, Hao Zhang, Tianhe Ren, Shilong Liu, Xueyan Zou, Huaizhe Xu, Hongyang Li, Jianwei Yang, Chunyuan Li, et al. Visual In-context Prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12871, 2024. 2

[25] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Feng Li, Tianhe Ren, Bohan Li, and Lei Zhang. TAPTRv2: Attention-based Position Update Improves Tracking Any Point. *Advances in Neural Information Processing Systems*, 2024. 1, 2, 3, 5, 6, 7, 12

[26] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, and Lei Zhang. TAPTR: Tracking Any Point with Transformers as Detection. In *European Conference on Computer Vision*, pages 57–75. Springer, 2024. 1, 2, 3, 5, 6, 12

[27] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR. In *International Conference on Learning Representations*, 2022. 3

[28] I Loshchilov. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[29] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 6

[30] Michal Neoral, Jonáš Šerỳch, and Jiří Matas. MFT: Long-Term Tracking of Every Pixel. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6837–6847, 2024. 1

[31] OpenAI. ChatGPT. https://openai.com/blog/chatgpt/, 2023. 4

[32] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 724–732, 2016. 6

[33] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International journal of computer vision*, 80:72–91, 2008. 1

[34] Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J Fleet. The Surprising Effectiveness of Diffusion Models for Optical Flow and Monocular Depth Estimation. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[35] Xiaoyu Shi, Zhaoyang Huang, Weikang Bian, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. VideoFlow: Exploiting Temporal Cues for Multi-frame Optical Flow Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12469–12480, 2023.

[36] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked Cost Volume Autoencoding for Pretraining Optical Flow Estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1599–1610, 2023. 3

[37] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. 4

[38] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 3

[39] Zachary Teed and Jia Deng. 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*, pages 402–419. Springer, 2020. 1, 3, 5

[40] Timo Teufel, Hongchao Shu, Roger D Soberanis-Mukul, Jan Emily Mangulabnan, Manish Sahu, S Swaroop Vedula, Masaru Ishii, Gregory Hager, Russell H Taylor, and Mathias Unberath. OneSLAM to Map Them All: A Generalized Approach to SLAM for Monocular Endoscopic Imaging Based on Tracking Any Point. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–8, 2024. 1

[41] Narek Tumanyan, Assaf Singer, Shai Bagon, and Tali Dekel. DINO-Tracker: Taming DINO for Self-Supervised Point Tracking in a Single Video. In *European Conference on Computer Vision*, pages 367–385. Springer, 2024. 1

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in neural information processing systems*, 30, 2017. 3

[43] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking Arbitrary Points for Few-shot Visual Imitation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5397–5403. IEEE, 2024. 1, 6, 7

[44] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-Invariant Matching Cost Learning for Accurate Optical Flow Estimation. *Cornell University - arXiv,Cornell University - arXiv*, 2020. 3

[45] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking Everything Everywhere All at Once. *ICCV*, 2023. 1

[46] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: Tracking Any 2D Pixels in 3D Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20406–20417, 2024. 1

[47] Haofei Xu, Jiaolong Yang, Jianfei Cai, Jie Zhang, and Xin Tong. High-Resolution Optical Flow from 1D Attention and Correlation. *Cornell University - arXiv,Cornell University - arXiv*, 2021. 3

[48] Jia Xu, Rene Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3

[49] Feihu Zhang, Oliver J. Woodford, Victor Prisacariu, and Philip H. S. Torr. Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[50] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[51] Hao Zhang, Hongyang Li, Feng Li, Tianhe Ren, Xueyan Zou, Shilong Liu, Shijia Huang, Jianfeng Gao, Chunyuan Li, Jainwei Yang, et al. Llava-Grounding: Grounded Visual Chat with Large Multimodal Models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2025. 4

[52] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. *arXiv preprint arXiv:2303.16199*, 2023. 4

[53] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global Matching with Overlapping Attention for Optical Flow Estimation. 3

[54] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19855–19865, 2023. 3, 6

[55] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 2021. 6

TAPTRv3: Spatial and Temporal Context Foster Robust Tracking of Any Point in Long Video

Supplementary Material

A. Appendix

In this appendix, we provide more ablation studies on the design of our key components and some hyper-parameters (See Sec. A.1). We then provide more details about the implementation and experimental settings (see Sec.A.2). Finally, we present some more visualization results to show the superiority of TAPTRv3 and the effectiveness of our designs (See Sec. A.3 and the supplementary videos).

A.1. More Ablation Studies

Number of Context Features. As discussed in Sec. 3.1 and Sec. 3.3, we sample N^2 features in the form of a grid with 1-pixel grid interval when preparing the point query's context features $\mathbf{C}$ and each sampling point's context features $\mathbf{K}_t^m$ in cross-attention. To find the optimal value of N, we conduct experiments with $N = 1$[5], $N = 3$, and $N = 5$. The results are shown in Table 7, where $N = 3$ obtains the best performance. Compared with sampling 25 context features ($N = 5$), sampling 9 context features ($N = 3$) around the point not only requires fewer computing resources but also yields better performance, with an AJ improvement of 0.8. These results suggest that for the task of point tracking, context features are necessary, but they must not be so excessive that they fail to accurately describe the point being tracked.

Number of Context Features N^2	AJ	$< \delta^x_{avg}$	OA
$N^2 = 1$	**51.9**	**64.8**	86.9
$N^2 = 9$	**52.9**	**65.8**	87.2
$N^2 = 25$	52.1	65.3	**87.4**

Table 7. **Ablation on number of context features N^2.**

Memory Size of VLTA. TAPTRv3 eliminates the sliding window [25, 26] and introduces the Visibility-aware Long-Temporal Attention (VLTA) module to extend the temporal attention to an arbitrary length, while considering the visibility. Excluding the use of visibility, the recent SAM2 [56][6] adopts a similar temporal modeling approach, where SAM2 maintains a memory to record features from past frames in a FIFO manner and limits the memory size to 8 to focus on recent frames. To demonstrate the advantage of extending temporal attention to arbitrary lengths and the generalization ability of our VLTA to temporal lengths, we conduct the ablation studies on the memory size. As illustrated in Table 8, the significant positive correlation between memory size and model performance indicates that, although the perception range of our VLTA is limited to 1 to 23 frames during training [7], our VLTA is still able to generalize beyond 24 frames. This generalization ability enables us to expand the range of temporal perception, allowing the collection of long-term temporal information from all past frames to help improve the robustness of long-term point tracking.

Memory Size of VLTA	AJ	$< \delta^x_{avg}$	OA
12	48.0	65.7	83.6
24	51.4	66.7	86.5
48	53.7	67.3	87.9
All Past	**54.5**	**67.5**	**88.2**

Table 8. **Ablation on memory size of VLTA.**

The Method of Obtaining Similarity Map in Global Matching. In Sec. 3.4, we propose to conduct the global matching to reestablish point tracking when a scene cut is encountered. Instead of simply relying on a point-level feature to construct a similarity map in global matching as in previous works [5, 7], we propose incorporating spatial context features, similar to the CCA module, to enhance accuracy. (For more implementation details, please refer to Sec. A.2). To investigate the effectiveness of this approach, we conduct an ablation study on it. As shown in Table 9, the incorporation of spatial context features brings a relatively small performance improvement. However, considering that we only trigger global matching when a scene cut is detected and its computational cost is negligible, we adopt the use of spatial context features in TAPTRv3.

Feature of Tracking Point	AJ	$< \delta^x_{avg}$	OA
Point-level Feature	55.07	67.96	87.16
Spatial Context Features	**55.10**	**68.06**	**87.37**

Table 9. **Ablation on calculation of similarity map**

To avoid misunderstanding, we emphasize that our main contribution in global matching lies in the novel triggering mechanism, rather than the global matching itself.

[5]When $N = 1$, the CCA will actually degenerate to the vanilla key-aware deformable attention, we can also find this performance in Row 5 of Table 2.

[6][56]Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman R¨adle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment Any-thing in Images and Videos. arXiv preprint arXiv:2408.00714, 2024. 12.

[7]Because our training data are videos with fixed lengths of 24 frames

A.2. More Implementation Details

Calculation of Similarity Map in Global Matching. In the global matching module of TAPTRv3, we construct a similarity map $\mathbf{H}_t \in \mathbb{R}^{H\times W}$ between the target tracking point and the feature map $\mathbf{X}_t \in \mathbb{R}^{H\times W\times D}$ of the current frame. However, instead of solely relying on a point-level feature as in previous methods, similar to CCA, we leverage the spatial context features $\mathbf{C} \in \mathbb{R}^{N^2\times D}$ to help improve the accuracy of the similarity map $\mathbf{H}_t$. As illustrated in Fig. 6, in our global matching, we first utilize each feature of the target tracking point's context feature to compute a group of similarity maps $H'_t \in \mathbb{R}^{H\times W\times N^2}$. However, since each feature in the context features is still point-level, computing the similarity map using any single one of them independently still introduces noise. Therefore, we further employ an MLP to comprehensively integrate these noisy similarity maps, resulting in a more accurate one $\mathbf{H}_t$.

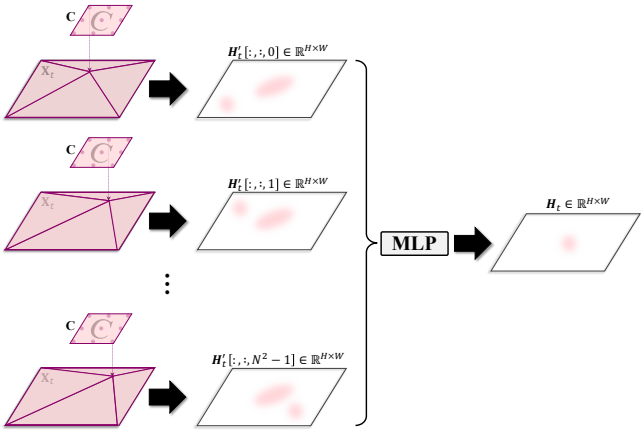


Figure 6. **Detailed visual illustration of global matching.**

Patch-level Similarity Calculation in CCA. Sec. 3.3 briefly introduces how to calculate patch-level similarity in Context-aware Cross-Attention (CCA), and Sec. 4.4 presents an ablation study on it. In this section, we provide a more detailed description. For the "Every two points" method that is adopted in TAPTRv3 by default, as shown in Fig 7 (a), each feature of the point query's N^2 context features is paired with every feature in the sampling point's N^2 context features K_t^m to obtain $\mathbf{S}_t^m \in \mathbb{R}^{N^2\times N^2}$. For the "Element-wise" method (See Sec. 4.4 and Table 9), as shown in Fig 7 (b), each feature of the point query's N^2 context features is only paired with its corresponding one in K_t^m to obtain $\mathbf{S}_t^m \in \mathbb{R}^{N^2}$

Spatial Context Updating. In TAPTRv3, we consistently employ the initial spatial context features $\mathbf{C}$ throughout all decoder layers. To validate its effectiveness, we add a spatial context feature update module in the transformer decoder for

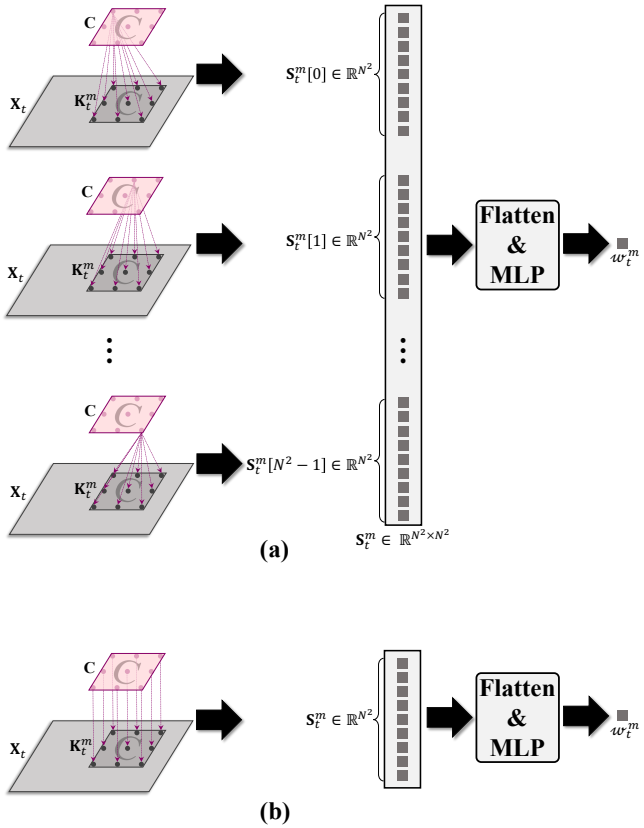


Figure 7. **Detailed visual illustration of different methods for computing patch-level similarity.**

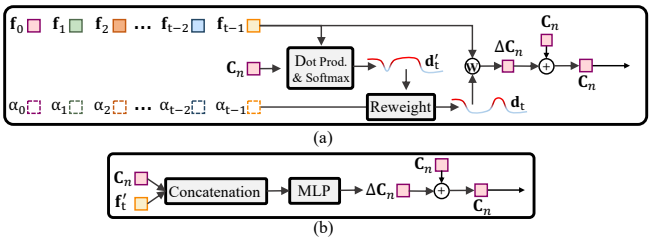


Figure 8. **Detailed visual illustration of different methods for the updating of spatial context features.**

comparison in our ablation study section (See Sec. 4.4 and Table 4). Here, we provide a detailed description of the two methods for updating spatial context features proposed in this ablation. For the one that uses our VLTA, as shown in Fig. 8 (a), for the n-th spatial context feature $\mathbf{C}_n$, we treat it as a query and attend VLTA to the refined content features of the past frames to update it. For the one that uses an MLP, as shown in Fig. 8 (b), we concatenate the spatial context feature with the point query's content feature in the current frame $\mathbf{f}'_t$ and update the spatial context features with the MLP.

A.3. More Visualizations

To further demonstrate the superiority of TAPTRv3 and the effectiveness of our designs, we provide more visualizations of TAPTRv3's predictions on some challenging real-world long videos. In these visualizations, we label the frame ID at the upper left corner of each frame to indicate the time stamp. To make the tracking results more visually distinct, we adjust the color of the tracking points based on the overall color of the video.

More Visual Comparison with TAPTRv2. We provide more visualizations of the comparisons between TAPTRv2 and TAPTRv3, demonstrating the superiority of TAPTRv3. For more details, please refer to the image captions.

Visual Comparison w. and wo. Global Matching. We provide more visualizations of the comparisons between TAPTRv3 with and without the auto-triggered global matching, demonstrating its effectiveness. For more details, please refer to the image captions.

More Robust Prediction Visualizations. We additionally provide more visualizations to demonstrate TAPTRv3's robustness in in-the-wild scenarios, showing its potential for various downstream applications.

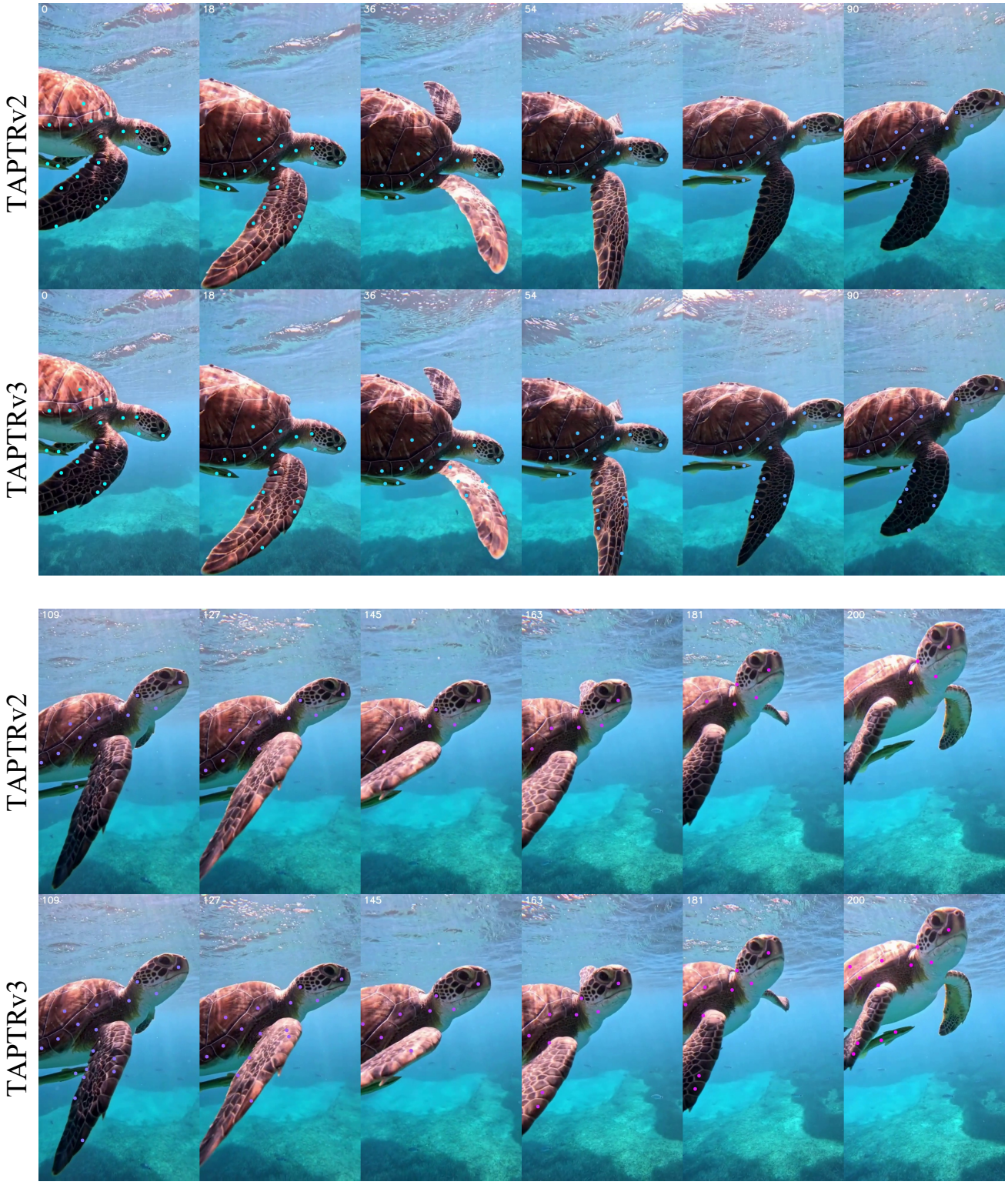


Figure 9. **Visual comparison between TAPTRv2 and TAPTRv3.** Best view in electronic version. From the third image in the first row (36th frame), TAPTRv2 loses tracking of the turtle's flippers, and in the last few frames loses tracking of the turtle shell and the point on the fish below the turtle. TAPTRv3, on the other hand, maintains stable and accurate tracking throughout the video.

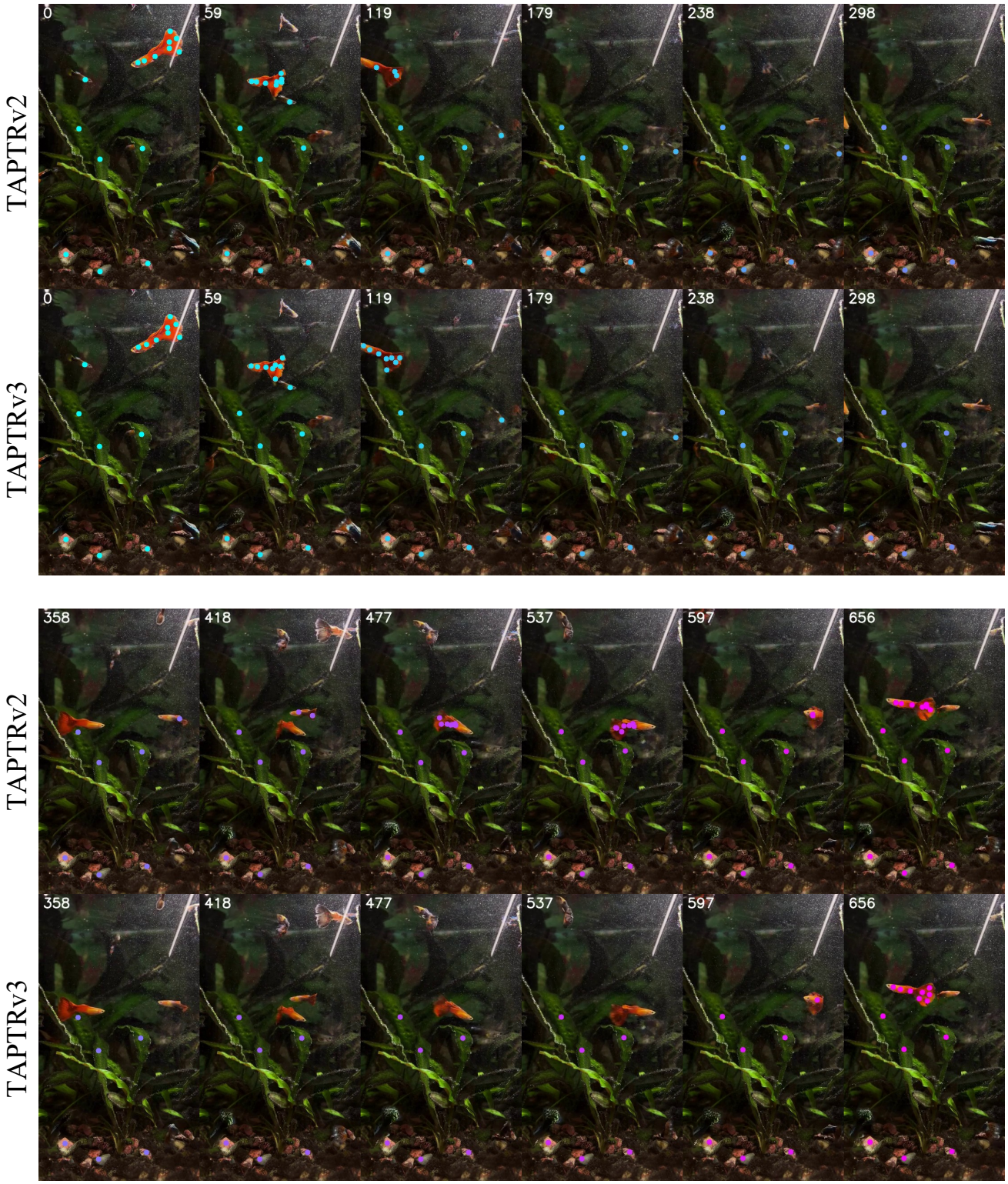


Figure 10. **Visual comparison between TAPTRv2 and TAPTRv3.** Best view in electronic version. When the goldfish is about to swim out of the frame from right to left (119th frame), TAPTRv2 loses many target tracking points. Afterward, the goldfish swims back from left to right, and starting from the 358th frame, the video shows the other side of the goldfish, where the original target tracking points are occluded. However, TAPTRv2 incorrectly estimates them as visible or on another fish. TAPTRv3, on the other hand, maintains the correct estimation. Until the last dozens of frames, when the goldfish turns around again, TAPTRv3 successfully detects the initial target tracking points, estimates them as visible, and provides accurate positions.

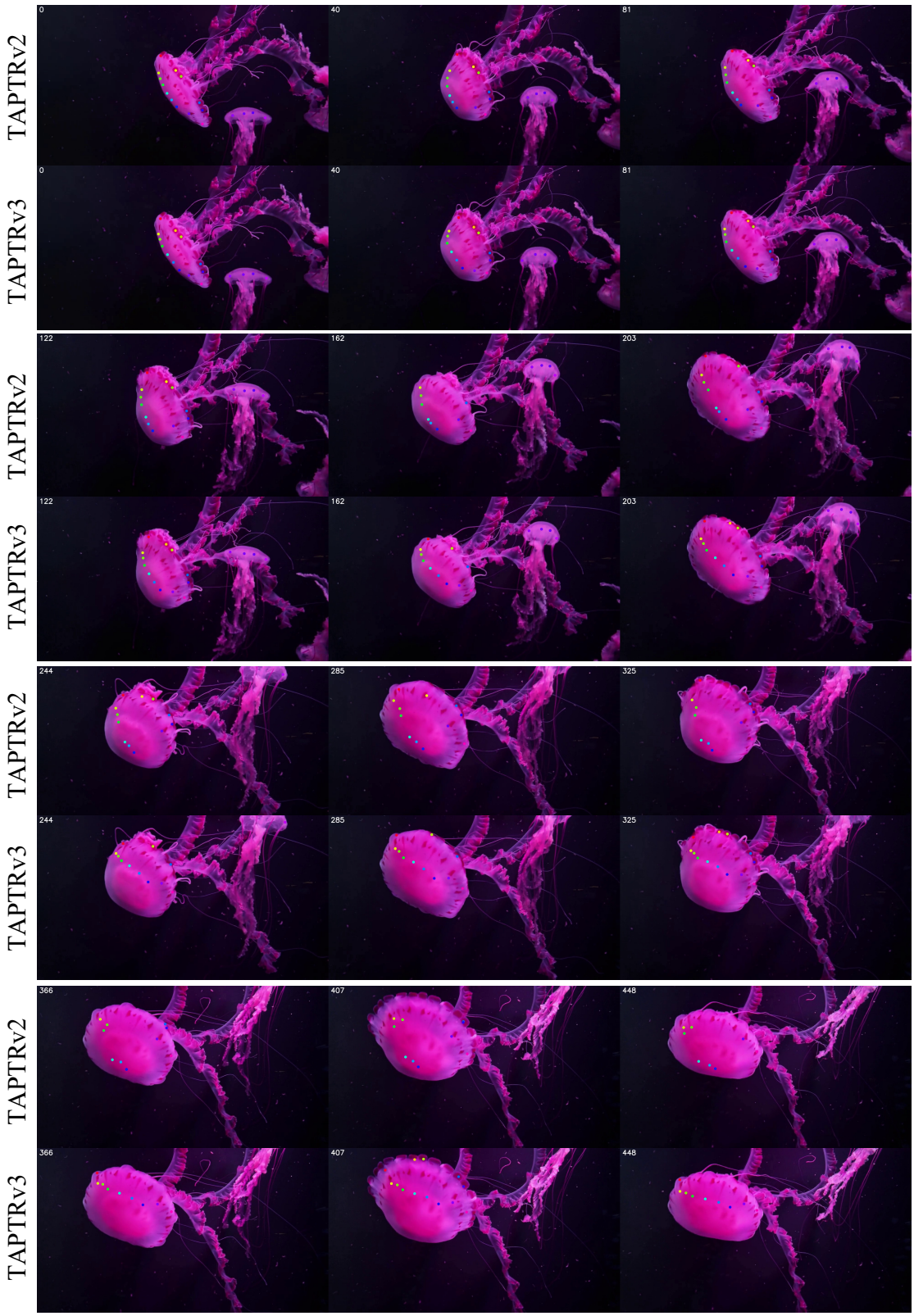


Figure 11. **Visual comparison between TAPTRv2 and TAPTRv3.** Best view in electronic version. Over time, TAPTRv2 incorrectly estimates the location and visibility of points on jellyfish, and the error accumulates, while TAPTRv3's results are more accurate.

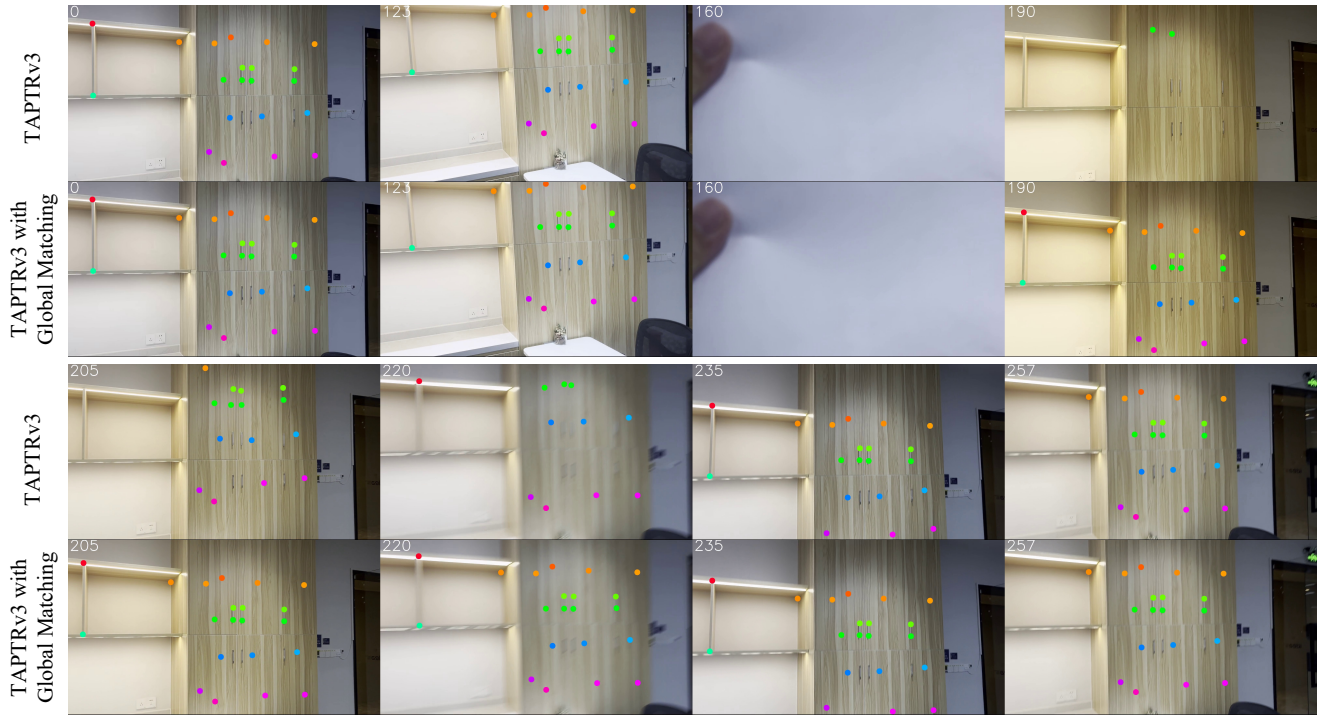


Figure 12. **Visual comparison between TAPTRv3 with and without the auto-triggered global matching.** After the occluder appears and then disappears, TAPTRv3 without auto-triggered global matching takes about 70 frames to successfully re-track the target tracking points. However, with the help of global matching, this process takes only two frames.

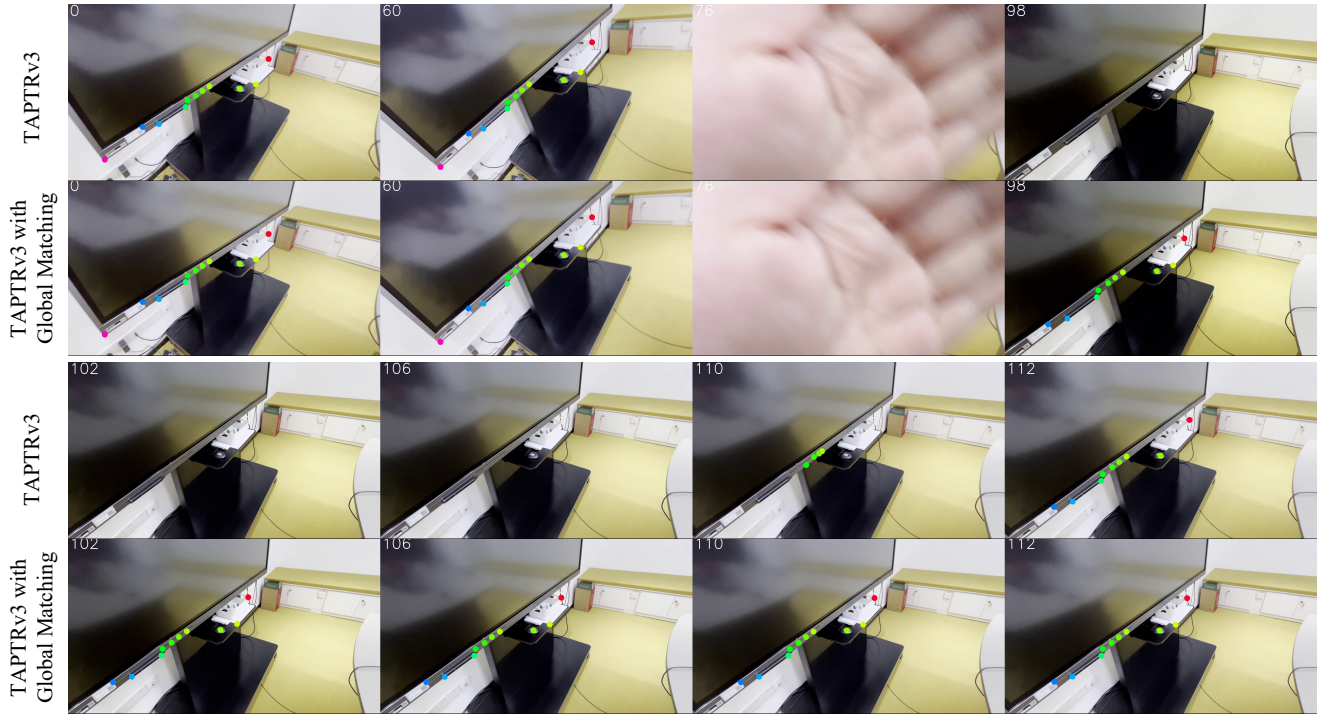
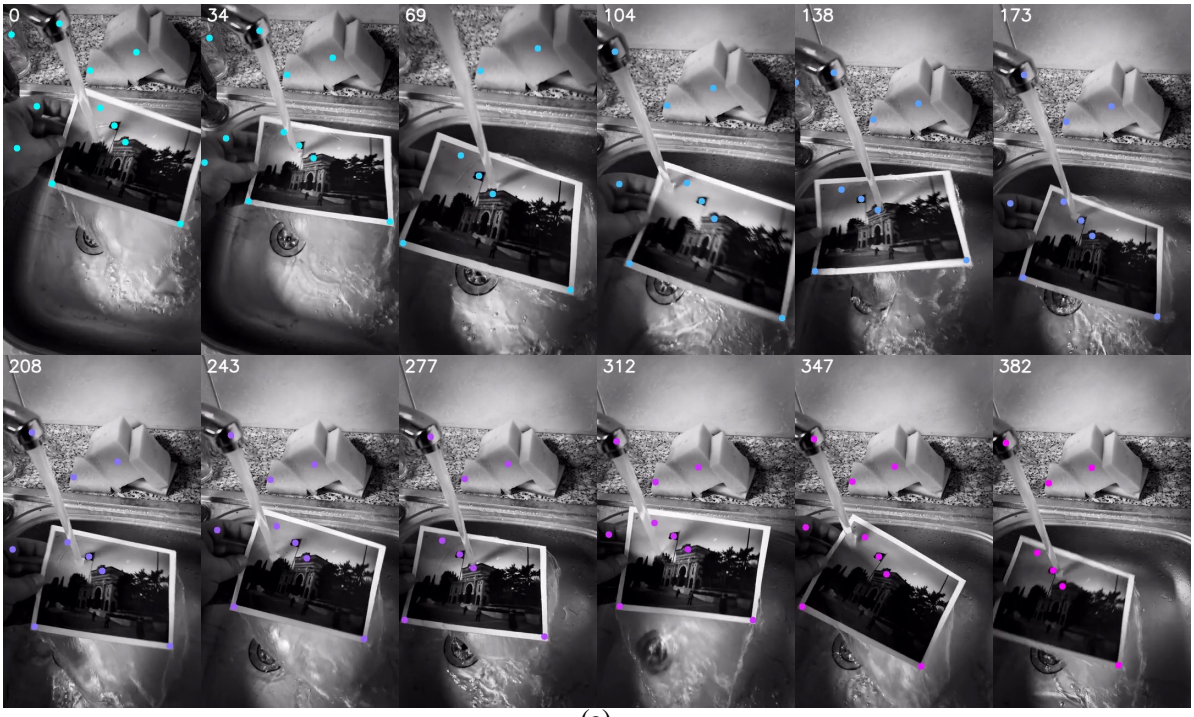
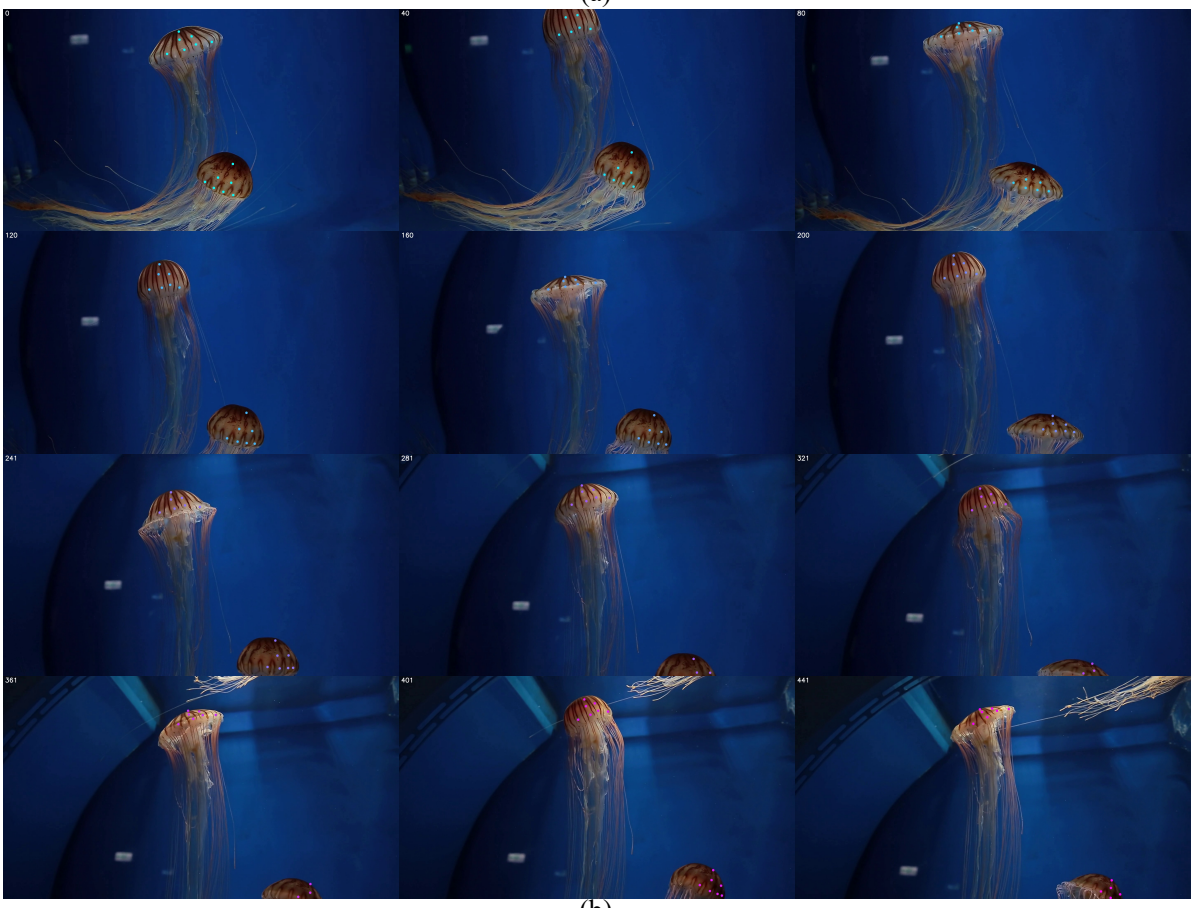


Figure 13. **Visual comparison between TAPTRv3 with and without the auto-triggered global matching.** After the occluder appears and then disappears, TAPTRv3 without auto-triggered global matching takes about 14 frames to successfully re-track the target tracking points. However, with the help of global matching, this process takes only two frames.



(a)



(b)

Figure 14. **Additional visualizations of TAPTRv3's robust predictions.**