NeighborTrack: Single Object Tracking by Bipartite Matching with Neighbor Tracklets and Its Applications to Sports

Yu-Hsi Chen[1], Chien-Yao Wang[1], Cheng-Yun Yang[2], Hung-Shuo Chang[1], Youn-Long Lin[3], Yung-Yu Chuang[4] and Hong-Yuan Mark Liao[1]
Institute of Information Science, Academia Sinica, Taiwan[1]
Department of Electrical Engineering, Purdue University[2]
Department of Computer Science, National Tsing Hua University[3]
Department of Computer Science and Information Engineering, National Taiwan University[4]
{franktpmvu,kinyiu,jonathanc,liao}@iis.sinica.edu.tw[1]
yang2316@purdue.edu[2], ylin@cs.nthu.edu.tw[3], lcyy@csie.ntu.edu.tw[4]

Abstract

We propose a post-processor, called NeighborTrack, that leverages neighbor information of the tracking target to validate and improve single-object tracking (SOT) results. It requires no additional data or retraining. Instead, it uses the confidence score predicted by the backbone SOT network to automatically derive neighbor information and then uses this information to improve the tracking results. When tracking an occluded target, its appearance features are untrustworthy. However, a general siamese network often cannot tell whether the tracked object is occluded by reading the confidence score alone, because it could be misled by neighbors with high confidence scores. Our proposed NeighborTrack takes advantage of unoccluded neighbors' information to reconfirm the tracking target and reduces false tracking when the target is occluded. For the VOT challenge dataset commonly used in short-term object tracking, we improve three famous SOT networks, Ocean, TransT, and OSTrack, by an average of 1.92% EAO and 2.11% robustness. For the mid- and long-term tracking experiments based on OSTrack, we achieve state-of-the-art 72.25% AUC on LaSOT and 75.7% AO on GOT-10K. Most of the tracking examples we have used are related to sports.

1. Introduction

Single Object Tracking (SOT) is a fundamental computer vision task that establishes the correspondence of an arbitrarily specified object along time [22]. There are numerous applications for it, including video surveillance [18, 25], video annotation [1], human-computer interaction [17], etc. An SOT network takes a user-specified target of interest in the first frame and then tracks its position in subsequent frames. In contrast to multi-object tracking (MOT), which knows the target classes in advance, SOT is unaware of the target classes. The current SOT approaches use various algorithms derived from the Siamese network [2]. With the help of a deep learning network, they extract the appearance feature of the target object and then use that feature to locate the positions of objects with similar features in subsequent frames. Using pairwise appearance feature alignment, the same network can be used to track multiple objects with different appearance features without retraining.

In sports-related tasks, the trajectories of objects and players are often used as features connecting time and space, such as in the cases of tactics analysis, event detection, or activity recognition, are all good examples of this sort. In order to obtain the trajectory of moving objects, a common way is to create a dedicated dataset for each object to be tracked, and train on MOT network in a supervised learning manner. However, MOT networks trained in this way are often only effective for a single type of motion. Looking at SOT from another perspective, it can help track objects of interest without knowing the object category and help build trajectories for unspecified sports.

This approach, however, might fail to track when the appearance characteristics of either side change.

To address the challenges caused by appearance changes, we propose a post-processor named NeighborTrack. The main idea is to verify the correctness of the tracking result using the spatial-temporal trajectories of the neighbors of the target. NeighborTrack simultaneously tracks the target and its neighbors. When the target is occluded, it reduces the number of matching errors by involving neighbors in the matching process.

Consider the example in Figure 1. In the current frame

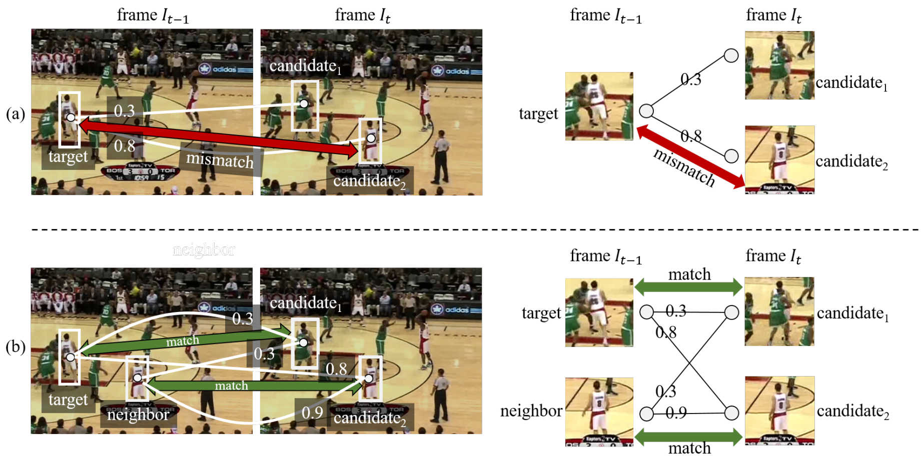


Figure 1. An illustrative example of how neighbor information can be used to correct tracking results.

I_t, there are two candidates for the tracking target. Numbers on the edges indicate the similarity between two objects. In frame I_t, the target object is occluded. When only considering appearance cues (Figure 1(a)), the target is incorrectly matched to the wrong candidate$_2$ because of occlusion. Aside from visual and spatiotemporal cues, NeighborTrack makes use of neighbors in the previous frame in order to resolve ambiguities caused by occlusions. Neighbors are objects that are similar to the target object. Often, they are the source of ambiguity. By considering neighbors (Figure 1(b)), we construct a bipartite graph with two sets of nodes: one for the hypothesis of matches (candidates) and one for the source of ambiguity (neighbors) and the target. After bipartite matching, since the neighbor is not occluded, it can be correctly matched with candidate$_2$. The target should therefore choose candidate$_1$ (the correct one), even if their similarity remains low due to occlusions. Our experiments indicate that NeighborTrack effectively mitigates the problems caused by occlusion and increases tracking accuracy.

We propose NeighborTrack, a post-processor for improving SOT, and have made the following contributions.

- We formulate the association problem as a bipartite matching problem between candidate and neighbor tracklets. Unlike attention-based methods that strengthen single-target appearance features, our method uses neighbor information to help correct wrong tracking when appearance features change.

- Based on cycle consistency, we calculate Intersection over Union (IoU) between two tracklets as a more robust measure of similarity. Both spatiotemporal and visual cues are considered in the measurement.

- Our approach is not in conflict with, and it can complement, methods for enhancing appearance features or enhancing object localization, such as DIMP [3] and the Alpha-refine [28]. The agnostic nature of our method enables it to be combined with most SOT methods.

- Our method does not require additional training data or retraining or fine-tuning, and neighbor information is readily available from the network-generated confidence scores without additional computation.

- We achieved state-of-the-art 72.25% AUC on the LaSOT [7] dataset. For the GOT-10K [9] dataset, we achieved 75.7% AO. On the bbox and mask datasets of VOT challenge [12], our method increases EAO and robustness by 2.11% and 1.92%, respectively.

2. Related Work

SOT methods based on Siamese networks [2] have been continuously refined and improved. The majority of research has focused on improving the header function by adding more effective tracking mechanisms. Some methods [14,15] borrow the region proposal network mechanism from Faster R-CNNs [21] to make a network more adaptive to target size changes. Wang *et al.* [22] propose a method that generates a mask for tracking the target while training the tracker in order to enable the network to track targets more accurately and interpretably when used in real scenes. Ocean [31] replaces anchor-based headers with anchor-free ones to alleviate issues associated with overlapping tracked targets and anchors in a crowded scene. Aside from the header issue, effective extraction of appearance features has always been a focus of SOT research. Recent methods [5,6,29] achieve breakthrough performance by employing a transformer network to simultaneously track the target and all the surrounding backgrounds, and then establishing an appearance feature model covering both self-attention and cross-attention.

Although the above-mentioned methods improve object tracking, they cannot overcome the tracking difficulties caused by occlusion [13]. Siamese network-based methods often fail when appearances change; therefore, relying solely on appearances is not unreliable. The confidence score predicted by a siamese network is a good indicator of appearance similarity, and objects with a similar appearance to the target will have high confidence scores. According to the tracking results on VOT challenge [12] reported by Zhang *et al.* [31], when non-tracking objects occlude the target object, the target appearance will change significantly, and this will greatly reduce the confidence score of the target position. It is therefore necessary to consider other information to overcome the challenge of occlusion.

The use of neighboring information is common in most multi-object tracking (MOT) systems. Even though the MOT task must consider multiple-to-multiple tracking, it is actually equivalent to taking into account the time-space matching relationship of all objects in the search range at the same time. There are, however, two differences between SOT and MOT. (1) Most MOT networks such as FairMOT [30] only track certain pre-determined classes. In contrast, a general SOT system should be able to track virtually any type of object. The feature space of an MOT model can be pre-trained on a class-specific Re-ID task to distinguish objects within a class, which is not applicable to objects outside the class. As for SOT, it does not have the above class information, and the requirements for appearance characteristics must be universal. Therefore, it is more susceptible to occlusion problems. (2) In the design of an MOT system, objects of the same class are tracked together. This is actually beneficial to tracking because the matching relationship between multiple objects and the background or other objects can be easily defined and then used to correct the tracking results. As for SOT, it does not have direct access to information about object adjacency. This paper presents a systematic approach to utilizing neighbor information for SOT.

Some methods [19, 32] take into account multiple objects in the SOT task. In comparison with these methods that utilize information across objects or frames, our method offers the following advantages: (1) Our method is a post-processing method. Thus, unlike these methods, our method does not require an additional network and retraining to obtain neighbor information, making it more flexible to use. (2) These methods only take into account information from two adjacent frames. Whenever a target is occluded, at least one frame's target appearance features will be unreliable. As a result of incorporating past information for a period of time, our method is able to better reduce tracking inaccuracies resulting from occlusions or changes in appearance over time.

3. NeighborTrack

This paper proposes NeighborTrack for improving the tracking results of an SOT network Φ as long as it meets two requirements. First, given a target template patch z, its bounding box b_0 specifying its position at the first frame I_0, and a sequence of frames $(I_1, \cdots, I_T)$, the network Φ can generate a tracking result $(b_1, \cdots, b_T)$ where b_t is the bounding box indicating the position of the target object at frame I_t. Second, for a frame I_t, the network can generate a set of candidate bounding boxes B^t and their confidence scores S^t. The two requirements are met by the majority of SOT networks.

To determine the bounding box of the target object at the current frame I_t, NeighborTrack produces a set of candidate bounding boxes $\mathcal{C}^t$. These are the hypotheses that the target matches. Among the candidates, we are seeking to identify the best match for the target. To have a more robust similarity measurement, NeighborTrack is inspired by the principle of cycle consistency [23]: an object tracked forward along a time line should be able to return to its original position when tracked backwards along time. In the current frame I_t, we have a forward-tracking tracklet $(b_{t-\tau}, \cdots, b_{t-1})$ for the target object. For the purpose of utilizing cycle consistency for similarity, we backtrack each candidate using Φ for τ frames in order to extract its backward-tracking tracklet. A candidate tracklet pool $\mathbf{P}^c$ is formed by these tracklets. Similarity between two tracklets is determined by their intersection over union (IoU) value. The measurement considers both visual and spatiotemporal cues since IoU takes the spatiotemporal cue into account, while visual cues are considered when tracklets are formed through forward/backward tracking. By measuring the similarity between the forward-tracking target tracklet and the backward-tracking candidate tracklet, we can verify how closely a candidate matches the target object using forward/backward tracking and cycle consistency. Figure 2 illustrates a real example of how a tracker can eliminate incorrect matches by ensuring cycle consistency.

NeighborTrack also maintains a neighbor tracklet pool $\mathbf{P}^n$ which contains neighbors that are similar to the target. Since neighbors are similar to the target (visually and spatiotemporally), they can cause false matches, particularly when the target's appearance changes or when it is obscured. As illustrated in Section 1, the neighbors are used to resolve ambiguity.

3.1. Candidate and neighbor tracklets

Candidate set. As one of the requirements, given the previous tracking results $(b_1, \cdots, b_{t-1})$ for $(I_1, \cdots, I_{t-1})$, the SOT network can generate a list of n_t candidate bounding boxes $\tilde{B}^t = \{\tilde{b}^t_1, \cdots, \tilde{b}^t_{n_t}\}$ and their corresponding con-

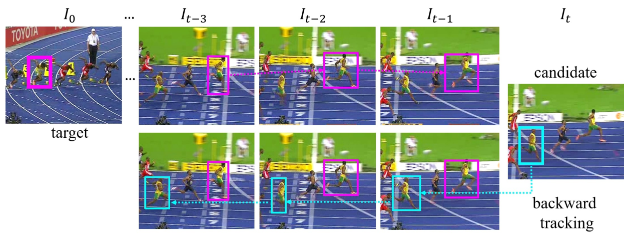


Figure 2. A real example of using cycle consistency to eliminate incorrect candidates. The example is from VOT [12]. Based on the target template at I_0, the tracker (OSTrack [29] in this example) tracks it until the frame I_{t-1} and produces a target tracklet (magenta boxes). For the current frame I_t, it selects the incorrect candidate due to severe view changes from the template. As a result of backtracking the candidate, we are able to obtain its backward-tracking candidate tracklet (cyan boxes). After calculating its similarity to the forward-tracking target tracklet, it is clear that the candidate does not match the target well. Using the proposed measure, we are able to verify matches better by utilizing the spatiotemporal relationship.

fidence scores $\tilde{S}^t = \{\tilde{s}_1^t, \cdots, \tilde{s}_{n_t}^t\}$ for the current frame I_t. Most SOT methods find the bounding box $\tilde{b}_{i_{max}}^t$ with the highest confidence score as the tracking result of I_t, where $i_{max} = \arg\max_i \tilde{s}_i^t$. Instead of picking up the most confident one, NeighborTrack maintains a candidate set of bounding boxes $\mathcal{C}^t$ and finds the match within $\mathcal{C}^t$. We first filter out bounding boxes with insufficient confidence scores:

$$(\hat{B}^t, \hat{S}^t) = \{(\tilde{b}_i^t, \tilde{s}_i^t) \mid \tilde{b}_i^t \in \tilde{B}^t \text{ and } \tilde{s}_i^t > \alpha \tilde{s}_{i_{max}}^t)\}, \quad (1)$$

where $\alpha \in [0, 1]$ is a hyperparameter for the threshold confidence ratio, $\hat{B}^t$ is the set of candidate bounding boxes with sufficient confidence and $\hat{S}^t$ are their confidence scores. Next, we perform non-maximum suppression on the remaining bounding boxes:

$$(B^t, S^t) = \text{SoftNMS}(\hat{B}^t, \hat{S}^t), \quad (2)$$

where SoftNMS is an improved version of non-maximum suppression [4], B^t and S^t respectively contain the bounding boxes and adjusted scores after SoftNMS.

If the target object is severely occluded, the correct match may not be included in the candidate set even if a loose threshold is applied. Thus, we also apply a Kalman filter to predict the candidate bounding box b^κ without relying on appearance features. The candidate set $\mathcal{C}^t$ is formed by adding b^κ into B^t, *i.e.*, $\mathcal{C}^t = B^t \cup \{b^\kappa\}$.

Candidate tracklet pool. For each candidate $b_i^t \in \mathcal{C}^t$, we generate its tracklet ξ_i^t by backtracking for τ frames. We set the patch z_i^t as the target template which is the patch cropped from using the bounding box b_i^t and use the SOT network Φ for backtracking, *i.e.*,

$$\xi_i^t = \Phi(z_i^t, b_i^t, (I_{t-1}, \cdots, I_{t-\tau})). \quad (3)$$

The tracklet ξ_i^t is a sequence of bounding boxes indicating the positions of the target template z_i^t from time $t-1$ to time $t{-}!\tau$. The set of all candidate tracklets is referred to as the candidate tracklet pool $\mathbf{P}^c$. Intuitively, the candidate tracklet pool contains the backtracking tracklets of objects that could potentially be the target object.

Neighbor tracklet pool. NeighborTrack also maintains another tracklet pool called the neighbor pool $\mathbf{P}^n$. Neighbor tracklets are essentially the candidate tracklets from the previous frame. By applying the method described in section Section 3.2, NeighborTrack selects a bounding box b_m^t in the candidate set $\mathcal{C}^t$ as the tracking result for I_t. Afterwards, NeighborTrack updates the neighbor pool $\mathbf{P}^n$ using the current candidate tracklet pool $\mathbf{P}^c$. In the first step, the selected tracklet ξ_m^t is removed from $\mathbf{P}^c$. For each tracklet ξ_i^t remaining in $\mathbf{P}^c$, we adjust its time span from $[t{-}1, t{-}\tau]$ to $[t, t-\tau+1]$ to be ready for the next frame I_{t+1}. This is accomplished by appending the associate candidate bounding box b_i^t at the head and removing the last bounding box:

$$\zeta_i^{t+1} = (b_i^t)^\frown \rho(\xi_i^t), \quad (4)$$

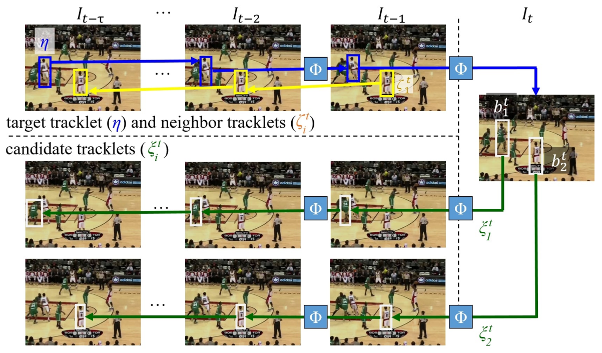
where $^\frown$ is the concatenation operator of two sequences and $\rho(s)$ removes the last element from the sequence s. The neighbor tracklet $\mathbf{P}^n$ is updated by $\{\zeta_i^{t+1} | i \neq m\}$ for the next frame I_{t+1}. As a result, the neighbor tracklets are the unselected candidate tracklets from the previous frame (after aligning the time span to the current frame).

We maintain the neighbor tracklets because they belong to neighbors of the target object in the tracking space. They are similar to the target object either visually or spatiotemporally and may cause ambiguity to the SOT network, particularly when the target object is obscured. By bipartite matching with those neighbors (Section 3.2), the ambiguity could be better resolved.

We only retain neighbor tracklets from the previous frame, not those from earlier frames. There are several reasons for this. To begin with, tracking accuracy degrades over time as outdated neighbor tracklets rarely provide useful information. Additionally, if a neighbor tracklet continues to survive, it should be possible to find the correspondence at the most recent time. Finally, retaining more tracklets will increase computation overhead. Thus, to avoid filling the pool with outdated neighbor tracklets that slow down computation speed, we only retain candidate tracklets from the previous time instance.

3.2. Tracking by bipartite matching

Through the process introduced in Section 3.1, we have the candidate pool $\mathbf{P}^c$ and the neighbor pool $\mathbf{P}^n$. The candidate pool contains hypothesis of tracking results while the neighbor pool contains the source of potential ambiguity. Given the target tracklet $\eta = (b_{t-1}, \cdots, b_{t-\tau})$ which is the



$I_{t-\tau}$
I_{t-2}
I_{t-1}
I_t
η
target tracklet (η) and neighbor tracklets (ζ_i^t)
candidate tracklets (ξ_i^t)
Φ
b_1^t
b_2^t
ξ_1^t
ξ_2^t

Figure 3. An illustration of the candidate and neighbor tracklets. The blue boxes represent the target tracklet (η), while the yellow boxes represent a neighbor tracklet (ζ_1^t). For the current frame I_t, the tracker Φ generates a set of candidates (the white boxes), b_1^t and b_2^t, in I_t. Using Φ for backtracking, we obtain their candidate tracklets (ξ_1^t and ξ_2^t). Note that since the target object is occluded in I_t, the confidence score of the correct candidate (b_1^t) is lower than that of the incorrect candidate (b_2^t). However, as b_2^t's candidate tracklet (ξ_2^t) matches very well with the neighbor tracklet (ζ_1^t), bipartite matching decides to match it with the neighbor rather than the target. Therefore, the target η has to be matched with the candidate ξ_1^t, thus correcting the false tracking. The example is from VOT [12].

tracking results for the previous τ frames, our goal is to find the association of η among hypothesis $\mathbf{P}^c$ while verifying with the source of ambiguity $\mathbf{P}^n$.

We cast the association problem into a bipartite matching problem. We have two sets of tracklets: $\mathbf{S}^c = \mathbf{P}^c$ and $\mathbf{S}^n = \mathbf{P}^n \cup \{\eta\}$. Each tracklet in $\mathbf{S}^c$ and $\mathbf{S}^n$ can be taken as a node. Thus, we have a bipartite graph whose two independent sets of nodes formed by $\mathbf{S}^c$ and $\mathbf{S}^n$. It is a complete bipartite graph since every node of the first set $\mathbf{S}^c$ is connected to every node of the second set $\mathbf{S}^n$. The weight w_{ij} associated with the edge between two nodes, $\xi_i^t \in \mathbf{S}^c$ and $\zeta_j^t \in \mathbf{S}^n$, is defined as the average IoU values between two tracklets. That is, if $\xi_i^t = (b_{t-1}^c, \cdots, b_{t-\tau}^c)$ and $\zeta_j^t = (b_{t-1}^n, \cdots, b_{t-\tau}^n)$, then we have

$$w_{ij} = \frac{1}{\tau} \sum_{k=t-\tau}^{t-1} \mathrm{IoU}(b_k^c, b_k^n), \tag{5}$$

where IoU calculates the IoU values between two bounding boxes. The weight reflects the trajectory similarity between two tracklets. We employ the Hungarian algorithm [20] to find the maximum matching for the resultant bipartite graph. Suppose that the candidate tracklet ξ_m^t is paired with the target tracklet η, its corresponding candidate bounding box b_m^t is selected as the tracking result for frame I_t. When the target tracklet η is not matched, we select the non-matched candidate tracklet with the highest IoU values with η. If the highest IoU is zero, we select the candidate b^κ predicted by the Kalman filter.

It is important to note that, although we only take the match of the target tracklet as the tracking result, all neighboring tracklets serve as a means of verifying the tracking result. If a candidate tracklet has a high IoU similarity to the target tracklet, it is likely to be the target. In contrast, if it has a high IoU similarity with any neighbor tracklet, it is likely to be associated with another object rather than the target. In this way, neighbor tracklets contribute to the matching process and assist in resolving ambiguities.

Tracking is significantly hampered by occlusions. Since the appearance feature is unreliable, it is easy to lose track of a target that has been occluded in the current frame. NeighborTrack addresses this issue with the help of neighbor tracklets. Assuming that the target object is occluded in the current frame, the correct candidate tracklet will have a lower similarity to the target tracklet. However, NeighborTrack can still find the correct tracking result as long as the objects in the candidate set are not occluded. Since the candidate objects are not occluded, their candidate tracklets can find excellent matches among the neighbor tracklets and will not select the target tracklet as their matches. Figure 3 gives an example on how bipartite matching works.

Our edge weight only considers the similarity of bounding boxes, and not the visual similarity of patches. There are two reasons for this. First of all, the appearance feature of an occluded target is less unreliable. In the event that the target is occluded, its appearance may resemble that of the occluder. Thus, visual similarity could mislead the results. Secondly, we construct candidate tracklets by backtracking patches of the candidate objects. In this way, the appearance features of candidates have already been implicitly considered in the tracklets.

4. Experiments

We begin by discussing the details of implementation in this section. Then, we apply NeighborTrack to improve several SOT networks and report the results on the VOT [12], LaSOT [7] and GOT-10K [9] datasets. We separate the sports-related videos in VOT and LaSOT datasets into two subsets: VOT-sports and LaSOT-sports, and use them to verify the effectiveness of our developed method in the domain of sports.

4.1. Implementation details

We used the following parameters in all experiments. A threshold ratio of $\alpha = 0.7$ is used to select candidate bounding boxes. For SoftNMS [4], the IoU threshold is set to 0.25, and σ is set to 0.01 for the Gaussian penalty function. The time period τ of backtracking tracklets is 9. In order to maintain neighbor information and enforce bipartite matching, our method slightly lowers the frame rate, as shown in Table 1. The hardware used in the experiment is 8 GTX 1080ti and 2 Intel E5 2620v4 CPU. When the time period τ

Table 1. The effect of changing τ on the calculation speed/AUC: Take for example the experiments on NeighborTrack on LaSOT benchmark [7] using OSTrack as the basis. From the table, we can see that the frame rate decays as τ increases. In addition, the tracking effect is slightly reduced if Kalman filter is not introduced, but at the same time, the frame rate is slightly improved.

model name	AUC↑	LaSOT FPS(Hz)↑
OSTrack384 [29]	0.711	3.63
OSTrack384 [29]+ours τ=3	0.714	2.40
OSTrack384 [29]+ours τ=9	0.722	1.58
OSTrack384 [29]+ours w/o Kalman filter τ=9	0.720	1.71
OSTrack384 [29]+ours τ=27	0.720	0.75
OSTrack384 [29]+ours τ=36	0.721	0.6

Table 2. Results of applying NeighborTrack to three different SOT methods on the VOT benchmarks [12]. Green numbers indicate that the original model has been improved.

model name	accuracy↑	VOT2020 robustness↑	EAO↑
Ocean [31]+AR [28]	0.757	0.810	0.515
Ocean [31]+AR [28] +ours	0.755	0.807	0.516
TransT [5]+AR [28]	0.769	0.775	0.490
TransT [5]+AR [28] +ours	0.768	0.816	0.523
OSTrack [29]+AR [28]	0.770	0.812	0.526
OSTrack [29]+AR [28] +ours	0.769	0.844	0.553
		VOT2021	
Ocean [31]+AR [28]	0.756	0.803	0.510
Ocean [31]+AR [28] +ours	0.754	0.803	0.515
TransT [5]+AR [28]	0.768	0.775	0.494
TransT [5]+AR [28] +ours	0.767	0.809	0.519
OSTrack [29]+AR [28]	0.770	0.810	0.528
OSTrack [29]+AR [28] +ours	0.769	0.843	0.556
		VOT2022bbox	
Ocean [31]	0.703	0.823	0.484
Ocean [31] +ours	0.703	0.822	0.486
TransT [5]	0.780	0.775	0.493
TransT [5] +ours	0.781	0.808	0.519
OSTrack [29]	0.779	0.824	0.538
OSTrack [29] +ours	0.779	0.845	0.564
		Average	
+ours	−0.07%	+2.11%	+1.92%

is equal to 9, the frame rate is 43% of the baseline. Also, not including Kalman filter will slightly reduce the performance of our method $(0.722 \rightarrow 0.720)$ when setting the same τ. As for the frame rate, it will increase slightly $(1.58 \rightarrow 1.71)$ due to the number decrease of the neighbors.

Because NeighborTrack requires extra computation, it is only activated when tracking results become unstable. We consider tracking results stable and will not activate NeighborTrack if two conditions are met. (1) If there is only one candidate in the set $\mathcal{C}^t$, there is no other option to match except $b^t_{i_{max}}$. (2) If the average IoU between the target tracklet η and the most confident candidate tracklet $\xi^t_{i_{max}}$ is higher than a threshold, then the tracking result is stable.

4.2. Datasets and competing methods

We conduct experiments on both short-term and long-term tracking. For short-term tracking, we use the VOT challenge [12] datasets including VOT2020, VOT2021, and VOT2022bbox [10–12]. These sequences feature multiple objects and many occlusions, which are the scenarios we wish to address. The NeighborTrack algorithm can be used with various S-networks, such as Ocean [31], TransT [5] and OSTrack [29], to effectively overcome appearance changes and occlusions. In addition, we use the mainstream medium and long-term datasets of the SOT field, including LaSOT [7] and GOT-10K [9], to verify NeighborTrack with the state-of-the-art method, OSTrack [29]. VOT2020 and VOT2021 are mask datasets, while LaSOT, VOT2022bbox, and GOT-10K are bbox datasets. We apply NeighborTrack to three previous models: a traditional attention-based network Ocean [31], a transformer-based network TransT [5], and a state-of-the-art network OSTrack [29].

4.3. VOT datasets

Table 2 shows the results of the proposed NeighborTrack. NeighborTrack is used to augment three representative models, Ocean, TransT, and OSTrack. Overall, NeighborTrack improves EAO by 1.92% and robustness by 2.11% on average. The accuracy decreases slightly at -0.07% because this metric only considers the average IoU of successful tracking, while the frames that fail to track do not participate in its calculation.

Figure 4 illustrates the tracking results for some examples. In the figure, the green box represents the ground truth; the magenta box represents the original results of the baseline, OSTrack [29];and the red box represents the NeighborTrack-corrected results. In the example of basketball, the players of the same team all wear the same color jerseys, and the angle of camera often causes occlusion problems. These reasons can easily lead to tracking the wrong target. In the Gymnastics3 example, the deformation of the gymnast's movement will cause a large change in the appearance characteristics, and the system will therefore mistakenly track the jumping platform and the referee. In the last example, Bolt1, multiple similar targets are on the screen at the same time, which makes the baseline mistakenly track other runners. After the correction of NeighborTrack, the correct targets of the above three examples can be tracked.

4.4. LaSOT and GOT-10K datasets

For the experiments of medium-term and long-term tracking, we used two datasets, LaSOT [7] and GOT-10K [9]. We use the metrics suggested by each dataset. Table 3 summarizes the results. The top three results are colored red, green, and blue, respectively. For LaSOT, OSTrack384 [29] leads all other methods. When

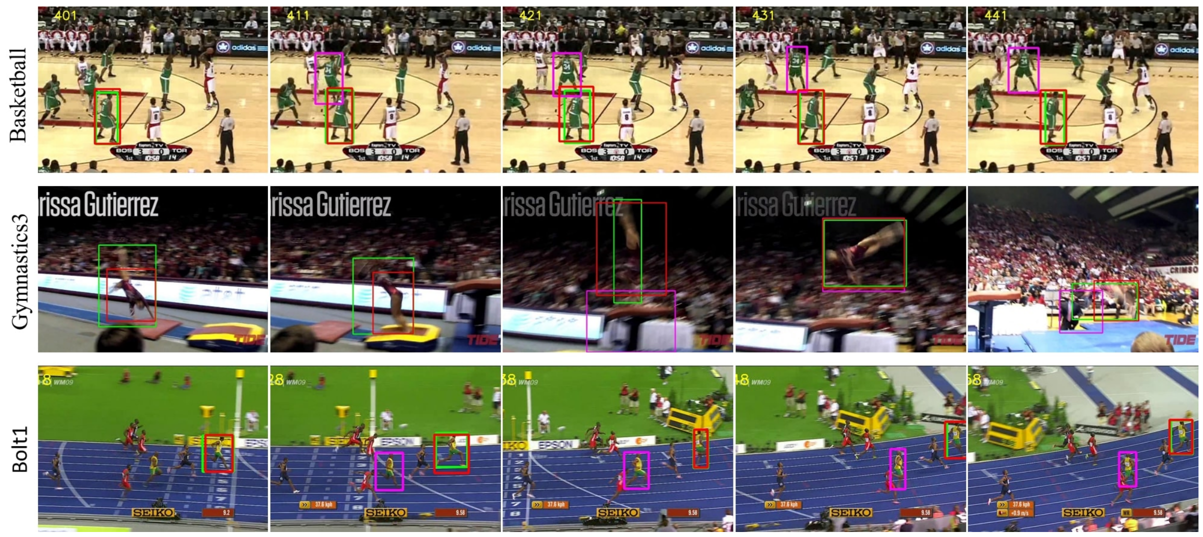


Figure 4. Examples of tracking results with and without NeighborTrack from VOT2022 [12]. We use OSTrack [29] as a baseline and correct its results using NeighborTrack. Ground truth is indicated by the green boxes. Magenta boxes represent baseline results, while red boxes represent results after applying NeighborTrack. Basekitball are examples that the baseline is unable to track because there are other object that resemble the target. In the case of Gymnastics3, deformation cause severe appearance changes, leading to the failure of the baseline. Whereas in Bolt 1, there are multiple similar targets in the frame, and when the target's angle changes, it can cause the baseline tracker to incorrectly track other similar-looking targets. In most cases, tracking errors are corrected after NeighborTrack has been applied.

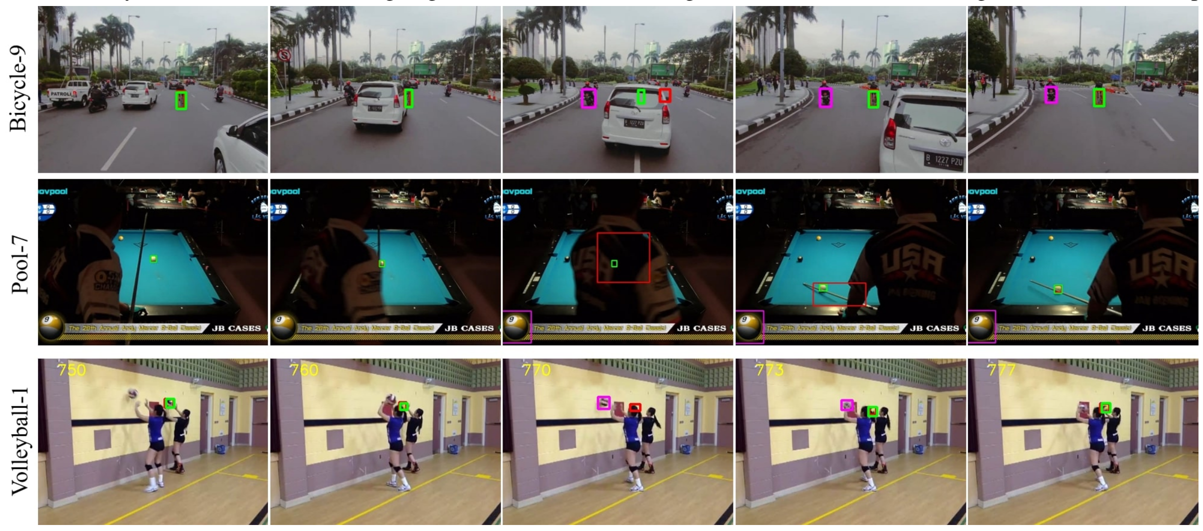


Figure 5. Examples of tracking results with and without NeighborTrack from LaSOT [7]. We apply NeighborTrack to correct the tracking results of OSTrack [29]. Ground truth is indicated by the green boxes. OSTrack's results are represented by magenta boxes, while NeighborTrack's results are represented by red boxes. OSTrack fails to track Bicycle-9, Pool-7 and Volleyball-1 due to the occlusion of the target, Kalman filter can prevent the tracker from being confused by neighbors similar to the target when the target is completely occluded, so as long as the target reappears, it can quickly return to the correct tracking position.

NeighborTrack is applied to OSTrack384, its performance is further improved, and OSTrack384+NeighborTrack outperforms all other methods in all metrics. The performance of OSTrack384 on GOT-10K is substantially inferior to MixFormer-L [6], the best method excluding ours. OSTrack384+NeighborTrack, however, outperforms MixFormer-L, because NeighborTrack boosts OSTrack384 significantly. Accordingly, NeighborTrack achieves state-of-the-art results for both the LaSOT and GOT-10K datasets.

Figure 5 provides examples of tracking results for LaSOT. In Bicycle-9, a bicycle is tracked as it travels in an array of vehicles. When a white car obscures the bicycle, the tracker tracks a similar-looking black motorcycle, since it loses its target. NeighborTrack recognizes that the black motorcycle is not the intended target, so it is set up to re-

Table 3. Comparisons of NeighborTrack (applied to OSTrack384) and other leading methods on the LaSOT [7] and GOT-10K [9] datasets.

model name	LaSOT AUC↑	LaSOT Norm-Precision↑	LaSOT Precision↑
OSTrack384 [29] +ours	0.722	0.818	0.780
OSTrack384 [29]	0.711	0.811	0.776
SwinV2-L 1K-MIM [24]	0.707	—	—
SwinTrack-B-384 [16]	0.702	0.784	0.753
MixFormer-L [6]	0.701	0.799	0.763
AiATrack [8]	0.690	0.794	0.738
Unicorn [26]	0.685	0.766	0.741
KeepTrack [19]	0.671	0.772	0.702
DMTrack [32]	0.574	—	0.580
model name	**GOT-10K AO↑**	**GOT-10K $SR_{0.5}$ ↑**	**GOT-10K $SR_{0.75}$ ↑**
OSTrack384 [29] +ours	0.757	0.8572	0.733
MixFormer-L [6]	0.756	0.8573	0.728
OSTrack384 [29]	0.737	0.832	0.708
SwinV2-L 1K-MIM [24]	0.729	—	—
SwinTrack-B-384 [16]	0.724	0.805	0.678
AiATrack [8]	0.696	0.800	0.632
STARK [27]	0.688	0.781	0.641

track the target at a later time. Our method has succeeded in this example by introducing the Kalman filter, which provides candidates independent of appearance features, thus providing a greater range of options to the tracking system. Although the Kalman filter is not extremely accurate, it can serve as a guide to prevent the tracker from becoming outrageous. Pool-7 is another example of specifying that the tracked object is fully occluded. In this example, the baseline selects the wrong target with a similar appearance when the pool ball is occluded by a player. And the Kalman filter keeps the tracking result near that position when the target disappears, and stops there until the target reappears. The volleyball-1 example is tracking the ball. When the ball is occluded by a player, the baseline method tracks to another volleyball at the next court. If we use our method, good tracking results can be obtained as long as one of the distractor and the target is not occluded.

4.5. LaSOT and VOT2022 sports dataset

We cut out two sports sub-datasets from short-term dataset VOT2022 and long-term dataset LaSOT. VOT2022-sports contains 21 videos, which are: $\{$ball2 to 3, basketball, bolt1, diver, gymnastics1 to 3, handball1 to 2, iceskater1 to 2, marathon, motocross1, polo, rowing, soccer1 to 2, surfing, tennis, wheel$\}$, and LaSOT-sports contains 32 videos, which are: $\{$basketball-1, basketball-6, basketball-7, basketball-11, bicycle-2, bicycle-7, bicycle-9, bicycle-18, surfboard-4, surfboard-5, surfboard-8, surfboard-12, pool-3, pool-7, pool-12, pool-15, skateboard-3, skateboard-8, skateboard-16, skateboard-19, yoyo-7, yoyo-15, yoyo-17, yoyo-19, motorcycle-1, motorcycle-3, motorcycle-9,

Table 4. LaSOT and VOT2022 sports dataset: We cut out the videos about sports in the original dataset as sub-dataset, then based on OSTrack384, added NeighborTrack and then made a comparison.

model name	LaSOT-sports AUC↑	LaSOT-sports Norm-Precision↑	LaSOT-sports Precision↑
OSTrack384 [29]	0.694	0.809	0.823
OSTrack384 [29]+ours	0.702	0.819	0.831
model name	**VOT2022bbox-sports accuracy↑**	**VOT2022bbox-sports robustness↑**	**VOT2022bbox-sports EAO↑**
OSTrack384 [29]	0.787	0.854	0.525
OSTrack384 [29]+ours	0.790	0.867	0.539

motorcycle-18, volleyball-1, volleyball-13, volleyball-18, volleyball-19$\}$. The results of our experiments are shown in Table 4. From Table 4, we found that the main indicators of baseline model, AUC and EAO, have decreased to a certain extent compared with the original dataset $(\text{AUC } 0.711 \rightarrow 0.694, \text{EAO } 0.538 \rightarrow 0.525)$, which shows that the difficulty of sports-subset is higher than that of the original dataset. Under these circumstances, our method improves all metrics on both subsets, which proves that our proposed method is effective for sports.

5. Conclusion

We propose NeighborTrack, a post-processing scheme that is agnostic and can be applied to state-of-the-art single object tracking methods provided that confidence scores are available. Through the use of neighbor information, NeighborTrack effectively mitigates the appearance change problem caused by occlusion or deformation. Our approach can be applied to both non-transformer-based methods [31] and transformer-based methods [5, 29]. Extensive experiments demonstrate that the proposed method is capable of improving the tracking performance of various SOT methods. There is no need to collect any additional training data for the proposed method. Furthermore, there is no need to retrain the SOT network. In summary, NeighborTrack complements most current state-of-the-art SOT algorithms and improves their accuracy.

6. Acknowledgement

The authors gratefully acknowledge the financial support provided by the Taiwan National Science and Technology Council (NSTC111-2634-F-002-023 and NSTC111-2634-F-002-022) and the ELAN Microelectronics (5T-1100309-1Q), which significantly contributed to the success of this research.

References

[1] Amanda Berg, Joakim Johnander, Flavie Durand de Gevigney, Jorgen Ahlberg, and Michael Felsberg. Semi-

automatic annotation of objects in visual-thermal video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019. 1

[2] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 850–865, Cham, 2016. 1, 2

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6181–6190, 2019. 2

[4] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – improving object detection with one line of code, 2017. 4, 5

[5] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, June 2021. 2, 6, 8

[6] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13608–13618, June 2022. 2, 7, 8

[7] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 5, 6, 7, 8

[8] Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. AiATrack: Attention in attention for transformer visual tracking. In *European Conference on Computer Vision*, pages 146–164. Springer, 2022. 8

[9] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021. 2, 5, 6, 8

[10] Matej Kristan, Alan Lukeziˇ c, Martin Danelljan, Luka Cehovin Zajc, and Jiri Matas. The new VOT2020 short-term tracking performance evaluation protocol and measures, 2020. 6

[11] Matej Kristan, JirıMatas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Hyung Jin Chang, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, Jani Kapyla, Gustav Hager, Song Yan, Jinyu Yang, Zhongqun Zhang, Gustavo Fernandez, and et. al. The ninth visual object tracking VOT2021 challenge results, 2021. 6

[12] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016. 2, 3, 4, 5, 6, 7

[13] B Y Lee, L H Liew, W S Cheah, and Y C Wang. Occlusion handling in videos object tracking: A survey. *IOP Conference Series: Earth and Environmental Science*, 18:012020, feb 2014. 3

[14] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[15] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018. 2

[16] Liting Lin, Heng Fan, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. *arXiv preprint arXiv:2112.00995*, 2021. 8

[17] Liwei Liu, Junliang Xing, Haizhou Ai, and Xiang Ruan. Hand posture recognition using finger geometric feature. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 565–568, 2012. 1

[18] Akshay Mangawati, Mohana, Mohammed Leesan, and H. V. Ravish Aradhya. Object tracking algorithms for video surveillance applications. In *2018 International Conference on Communication and Signal Processing (ICCSP)*, pages 0667–0671, 2018. 1

[19] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13444–13454, 2021. 3, 8

[20] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957. 5

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. 2

[22] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019. 1, 2

[23] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 3

[24] Zhenda Xie, Zigang Geng, Jingcheng Hu, Zheng Zhang, Han Hu, and Yue Cao. Revealing the dark secrets of masked image modeling, 2022. 8

[25] Junliang Xing, Haizhou Ai, and Shihong Lao. Multiple human tracking based on multi-view upper-body detection and discriminative learning. In *2010 20th International Conference on Pattern Recognition*, pages 1698–1701, 2010. 1

[26] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. In *European Conference on Computer Vision(ECCV)*, 2022. 8

[27] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10448–10457, 2021. 8

[28] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5289–5298. Computer Vision Foundation / IEEE, 2021. 2, 6

[29] Botao Ye, Hong Chang, Bingpeng Ma, and Shiguang Shan. Joint feature learning and relation modeling for tracking: A one-stream framework. In *European Conference on Computer Vision(ECCV)*, 2022. 2, 4, 6, 7, 8

[30] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision (to appear)*, August 2021. 3

[31] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXI*, volume 12366 of *Lecture Notes in Computer Science*, pages 771–787. Springer, 2020. 2, 3, 6, 8

[32] Zikai Zhang, Bineng Zhong, Shengping Zhang, Zhenjun Tang, Xin Liu, and Zhaoxiang Zhang. Distractor-aware fast tracking via dynamic convolutions and mot philosophy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1024–1033, June 2021. 3, 8