Learning Target Candidate Association to Keep Track of What Not to Track

Christoph Mayer Martin Danelljan Danda Pani Paudel Luc Van Gool
Computer Vision Lab, D-ITET, ETH Zürich, Switzerland

Abstract

The presence of objects that are confusingly similar to the tracked target, poses a fundamental challenge in appearance-based visual tracking. Such distractor objects *are easily misclassified as the target itself, leading to eventual tracking failure. While most methods strive to suppress distractors through more powerful appearance models, we take an alternative approach.*

We propose to keep track *of distractor objects in order to continue tracking the target. To this end, we introduce a learned association network, allowing us to propagate the identities of all target candidates from frame-to-frame. To tackle the problem of lacking ground-truth correspondences between distractor objects in visual tracking, we propose a training strategy that combines partial annotations with self-supervision. We conduct comprehensive experimental validation and analysis of our approach on several challenging datasets. Our tracker sets a new state-of-the-art on six benchmarks, achieving an AUC score of* 67.1% *on LaSOT [21] and a* +5.8% *absolute gain on the OxUvA long-term dataset [41]. The code and trained models are available at* `https://github.com/visionml/pytracking`

1. Introduction

Generic visual object tracking is one of the fundamental problems in computer vision. The task involves estimating the state of the target object in every frame of a video sequence, given only the initial target location. Most prior research has been devoted to the development of robust appearance models, used for locating the target object in each frame. The two currently dominating paradigms are Siamese networks [2, 35, 34] and discriminative appearance modules [3, 13]. While the former employs a template matching in a learned feature space, the latter constructs an appearance model through a discriminative learning formulation. Although these approaches have demonstrated promising performance in recent years, they are effectively limited by the quality and discriminative power of the appearance model.

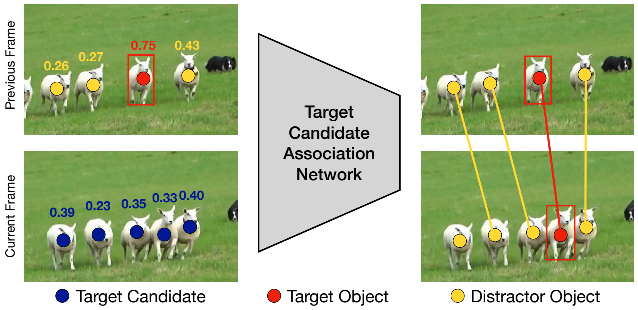


Figure 1. Visualization of the proposed target candidate association network used for tracking. For each target candidate (●) we extract a set of features such as score, position and appearance in order to associate candidates across frames. The proposed target association network then allows to associate these candidates (●) with the detected distractors (●) and the target object (●) of the previous frame. Lines connecting circles represent associations.

As one of the most challenging factors, co-occurrence of distractor objects similar in appearance to the target is a common problem in real-world tracking applications [4, 56, 46]. Appearance-based models struggle to identify the sought target in such cases, often leading to tracking failure. Moreover, the target object may undergo a drastic appearance change over time, further complicating the discrimination between target and distractor objects. In certain scenarios, *e.g.*, as visualized in Fig. 1, it is even virtually impossible to unambiguously identify the target solely based on appearance information. Such circumstances can only be addressed by leveraging other cues during tracking, for instance the spatial relations between objects. We therefore set out to address problematic distractors by exploring such alternative cues.

We propose to actively keep track of distractor objects in order to ensure more robust target identification. To this end, we introduce a target candidate association network, that matches distractor objects as well as the target across frames. Our approach consists of a base appearance tracker, from which we extract target candidates in each frame. Each candidate is encoded with a set of distinctive features, consisting of the target classifier score, location, and appearance. The encodings of all candidates are jointly processed by a graph-based candidate embedding network. From the resulting embeddings, we compute the association scores between all candidates in subsequent frames, allowing us to keep track of the target and distractor objects over time. In addition, we estimate a target detection confidence, used to increase the robustness of the target classifier.

While associating target candidates over time provides a powerful cue, learning such a matching network requires tackling a few key challenges. In particular, generic visual object tracking datasets only provide annotations of one object in each frame, *i.e*., the target. As a result, there is a lack of ground-truth annotations for associating distractors across frames. Moreover, the definition of a distractor is not universal and depends on the properties of the employed appearance model. We address these challenges by introducing an approach that allows our candidate matching network to learn from real tracker output. Our approach exploits the single target annotations in existing tracking datasets in combination with a self-supervised strategy. Furthermore, we actively mine the training dataset in order to retrieve rare and challenging cases, where the use of distractor association is important, in order to learn a more effective model.

Contributions: In summary, our contributions are as follows: **(i)** We propose a method for target candidate association based on a learnable candidate matching network. **(ii)** We develop an online object association method in order to propagate distractors and the target over time and introduce a sample confidence score to update the target classifier more effectively during inference. **(iii)** We tackle the challenges with incomplete annotation by employing partial supervision, self-supervised learning, and sample-mining to train our association network. **(iv)** We perform comprehensive experiments and ablative analyses by integrating our approach into the tracker SuperDiMP [11, 16, 3]. The resulting tracker *KeepTrack* sets a new state-of-the-art on six tracking datasets, obtaining an AUC of 67.1% on LaSOT [21] and 69.7% on UAV123 [37].

2. Related Work

Discriminative appearance model based trackers [12, 3, 27, 25, 51, 15] aim to suppress distractors based on their appearance by integrating background information when learning the target classifier online. While often increasing robustness, the capacity of an online appearance model is still limited. A few works have therefore developed more dedicated strategies of handling distractors. Bhat *et al*. [4] combine an appearance based tracker and an RNN to propagate information about the scene across frames. It internally aims to track all regions in the scene by maintaining a learnable state representation. Other methods exploit the existence of distractors explicitly in the method formulation. DaSiamRPN [56] handles distractor objects by subtracting corresponding image features from the target template during online tracking. Xiao *et al*. [46] use the locations of distractors in the scene and employ hand crafted rules to classify image regions into background and target candidates on each frame. SiamRCNN [42] associates subsequent detections across frames using a hand-crafted association score to form short tracklets. In contrast, we introduce a learnable network that explicitly associates target candidates from frame-to-frame. Zhang *et al*. [55] propose a tracker inspired by the multi object tracking (MOT) philosophy of tracking by detection. They use the top-k predicted bounding boxes for each frame and link them between frames by using different features. In contrast, we omit any hand crafted features but fully learn to predict the associations using self-supervision.

Many online trackers [12, 3, 13] employ a memory to store previous predictions to fine-tune the tracker. Typically the oldest sample is replaced in the memory and an age-based weight controls the contribution of each sample when updating the tracker online. Danelljan *et al*. [14] propose to learn the tracking model and the training sample weights jointly. LTMU [10] combines an appearance based tracker with a learned meta-updater. The goal of the meta-updater is to predict whether the employed online tracker is ready to be updated or not. In contrast, we use a learned target candidate association network to compute a confidence score and combine it with sample age to manage the tracker updates.

The object association problem naturally arises in MOT. The dominant paradigm in MOT is *tracking-by-detection* [5, 1, 49, 40, 52], where tracking is posed as the problem of associating object detections over time. The latter is typically formulated as a graph partitioning problem. Typically, these methods are non-causal and thus require the detections from all frames in the sequence. Furthermore, MOT typically focuses on a limited set of object classes [17], such as pedestrians, where strong object detectors are available. In comparison we aim at tracking different objects in different sequences solely defined by the initial frame. Furthermore, we lack ground truth correspondences of all distractor objects from frame to frame whereas the ground-truth correspondences of different objects in MOT datasets are typically provided [17]. Most importantly, we aim at associating target candidates that are defined by the tracker itself, while MOT methods associate all detections that correspond to one of the sought objects.

3. Method

In this section, we describe our tracking approach, which actively associates distractor objects and the sought target across multiple frames.

3.1. Overview

An overview of our tracking pipeline is shown in Fig. 2. We use a base tracker with a discriminative appearance

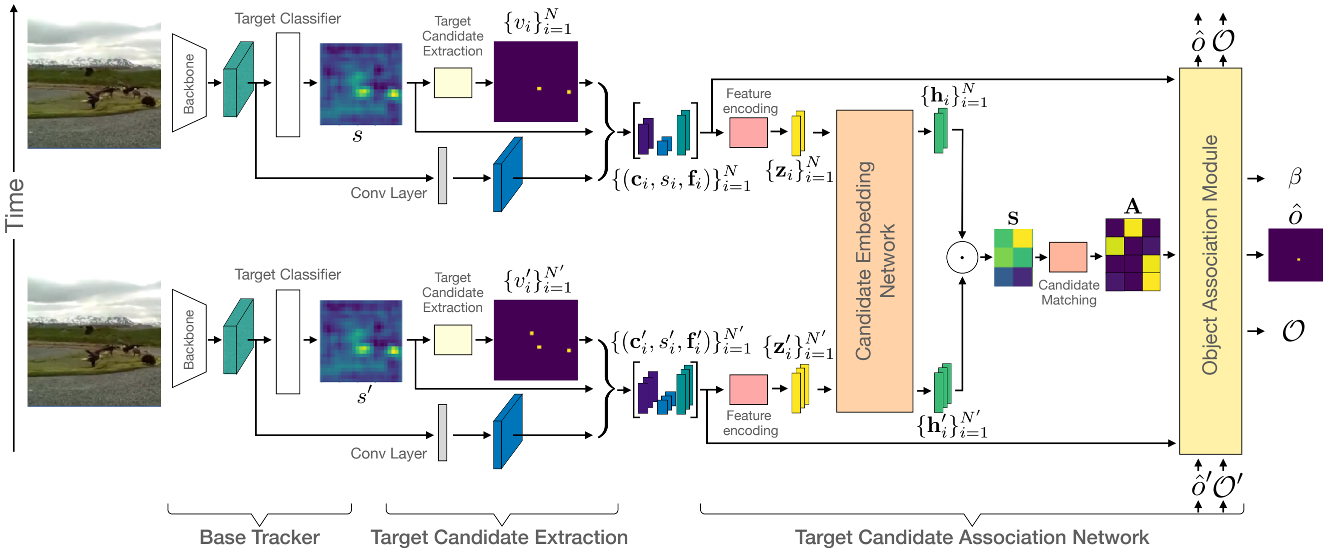

Time
Backbone
Target Classifier
Target Candidate Extraction
$\{v_i\}_{i=1}^{N}$
s
Conv Layer
$\{(\mathbf{c}_i, s_i, \mathbf{f}_i)\}_{i=1}^{N}$
Feature encoding
$\{\mathbf{z}_i\}_{i=1}^{N}$
Candidate Embedding Network
$\{\mathbf{h}_i\}_{i=1}^{N}$
S
Candidate Matching
A
Object Association Module
$\hat{o}$ $\mathcal{O}$
β
$\hat{o}$
$\mathcal{O}$
Backbone
Target Classifier
Target Candidate Extraction
$\{v'_i\}_{i=1}^{N'}$
s'
Conv Layer
$\{(\mathbf{c}'_i, s'_i, \mathbf{f}'_i)\}_{i=1}^{N'}$
$\{\mathbf{z}'_i\}_{i=1}^{N'}$
Feature encoding
$\{\mathbf{h}'_i\}_{i=1}^{N'}$
$\hat{o}'$ $\mathcal{O}'$
Base Tracker
Target Candidate Extraction
Target Candidate Association Network

Figure 2. Overview of the entire online tracking pipeline, processing the previous and current frames jointly to predict the target object.

model and internal memory. In particular, we adopt the SuperDiMP [11, 26] tracker, which employs the target classifier in DiMP [3] and the probabilistic bounding-box regression from [16], together with improved training settings.

We use the base tracker to predict the target score map s for the current frame and extract the target candidates v_i by finding locations in s with high target score. Then, we extract a set of features for each candidate. Namely: target classifier score s_i, location $\mathbf{c}_i$ in the image, and an appearance cue $\mathbf{f}_i$ based on the backbone features of the base tracker. Then, we encode this set of features into a single feature vector $\mathbf{z}_i$ for each candidate. We feed these representations and the equivalent ones of the previous frame – already extracted beforehand – into the candidate embedding network and process them together to obtain the enriched embeddings $\mathbf{h}_i$ for each candidate. These feature embeddings are used to compute the similarity matrix $\boldsymbol{S}$ and to estimate the candidate assignment matrix $\boldsymbol{A}$ between the two consecutive frames using an optimal matching strategy.

Once having the candidate-to-candidate assignment probabilities estimated, we build the set of currently visible objects in the scene $\mathcal{O}$ and associate them to the previously identified objects $\mathcal{O}'$, *i.e.*, we determine which objects disappeared, newly appeared, or stayed visible and can be associated unambiguously. We then use this propagation strategy to reason about the target object $\hat{o}$ in the current frame. Additionally, we compute the target detection confidence β to manage the memory and control the sample weight, while updating the target classifier online.

3.2. Problem Formulation

Let the set of *target candidates*, which includes distractors and the sought target, be $\mathcal{V} = \{v_i\}_{i=1}^{N}$, where N denotes the number of candidates present in each frame. We define the target candidate sets $\mathcal{V}'$ and $\mathcal{V}$ corresponding to the previous and current frames, respectively. We formulate the problem of target candidate association across two subsequent frames as, finding the assignment matrix $\boldsymbol{A}$ between the two sets $\mathcal{V}'$ and $\mathcal{V}$. If the target candidate v'_i corresponds to v_j then $\boldsymbol{A}_{i,j} = 1$ and $\boldsymbol{A}_{i,j} = 0$ otherwise.

In practice, a match may not exist for every candidate. Therefore, we introduce the concept of dustbins, which is commonly used for graph matching [38, 18] to actively handle the non-matching vertices. The idea is to match the candidates without match to the dustbin on the missing side. Therefore, we augment the assignment matrix $\boldsymbol{A}$ by an additional row and column representing dustbins. It follows that a newly appearing candidate v_j – which is only present in the set $\mathcal{V}$ – leads to the entry $\boldsymbol{A}_{N'+1,j} = 1$. Similarly, a candidate v'_i that is no longer available in the set $\mathcal{V}$ results in $\boldsymbol{A}_{i,N+1} = 1$. To solve the assignment problem, we design a learnable approach that predicts the matrix $\boldsymbol{A}$. Our approach first extracts a representation of the target candidates, which is discussed below.

3.3. Target Candidate Extraction

Here, we describe how to detect and represent target candidates and propose a set of features and their encoding. We define the set of target candidates $\mathcal{V}$ as all unique coordinates $\mathbf{c}_i$ that correspond to a local maximum with minimal score in the target score map s. Thus, each target candidate v_i and its coordinate $\mathbf{c}_i$ need to fulfill the following two constraints,

$$\phi_{\max}(s, \mathbf{c}_i) = 1 \quad \text{and} \quad s(\mathbf{c}_i) \geq \tau, \tag{1}$$

where $\phi_{\max}$ returns 1 if the score at $\mathbf{c}_i$ is a local maximum of s or 0 otherwise, and τ denotes a threshold. This definition allows us to build the sets $\mathcal{V}'$ and $\mathcal{V}$, by retrieving the local maxima of s' and s with sufficient score value. We use the max-pooling operation in a 5×5 local neighbourhood to retrieve the local maxima of s and set $\tau = 0.05$.

For each candidate we build a set of features inspired by two observations. First, we notice that the motion of the same objects from frame to frame is typically small and thus similar locations and similar distances between different objects. Therefore, the position $\mathbf{c}_i$ of a target candidate forms a strong cue. In addition, we observe only small changes in appearance for each object. Therefore, we use the target classifier score $s_i = s(\mathbf{c}_i)$ as another cue. In order to add a more discriminative appearance based feature $\mathbf{f}_i = \mathbf{f}(\mathbf{c}_i)$, we process the backbone features (used in the baseline tracker) with a single learnable convolution layer. Finally, we build a feature tuple for each target candidate as $(s_i, \mathbf{f}_i, \mathbf{c}_i)$. These features are combined in the following way,

$$\mathbf{z}_i = \mathbf{f}_i + \psi(s_i, \mathbf{c}_i), \quad \forall v_i \in \mathcal{V},$$

where ψ denotes a Multi-Layer Perceptron (MLP), that maps s and $\mathbf{c}$ to the same dimensional space as $\mathbf{f}_i$. This encoding permits jointly reasoning about appearance, target similarity, and position.

3.4. Candidate Embedding Network

In order to further enrich the encoded features and in particular to facilitate extracting features while being aware of neighbouring candidates, we employ a candidate embedding network. On an abstract level, our association problem bares similarities with the task of sparse feature matching. In order to incorporate information of neighbouring candidates, we thus take inspiration from recent advances in this area. In particular, we adopt the SuperGlue [38] architecture that establishes the current state-of-the-art in sparse feature matching. Its design allows to exchange information between different nodes, to handle occlusions, and to estimate the assignment of nodes across two images. In particular, the features of both frames translate to nodes of a single complete graph with two types of directed edges: 1) self edges within the same frame and 2) cross edges connecting only nodes between the frames. The idea is then to exchange information either along self or cross edges.

The adopted architecture [38] uses a Graph Neural Network (GNN) with message passing that sends the messages in an alternating fashion across self or cross edges to produce a new feature representation for each node after every layer. Moreover, an attention mechanism computes the messages using self attention for self edges and cross attention for cross edges. After the last message passing layer a linear projection layer extracts the final feature representation $\mathbf{h}_i$ for each candidate v_i.

3.5. Candidate Matching

To represent the similarities between candidates $v'_i \in \mathcal{V}'$ and $v_j \in \mathcal{V}$, we construct the similarity matrix $\boldsymbol{S}$. The sought similarity is measured using the scalar product: $\boldsymbol{S}_{i,j} = \langle \mathbf{h}'_i, \mathbf{h}_j \rangle$, for feature vectors $\mathbf{h}'_i$ and $\mathbf{h}_j$ corresponding to the candidates v'_i and v_j.

As previously introduced, we make use of the dustbin-concept [18, 38] to actively match candidates that miss their counterparts to the so-called dustbin. However, a dustbin is a virtual candidate without any feature representation $\mathbf{h}_i$. Thus, the similarity score is not directly predictable between candidates and the dustbin. A candidate corresponds to the dustbin, only if its similarity scores to all other candidates are sufficiently low. In this process, the similarity matrix $\boldsymbol{S}$ represents only an initial association prediction between candidates disregarding the dustbins. Note that a candidate corresponds either to an other candidate or to the dustbin in the other frame. When the candidates v'_i and v_j are matched, both constraints $\sum_{i=1}^{N'} \boldsymbol{A}_{i,j} = 1$ and $\sum_{j=1}^{N} \boldsymbol{A}_{i,j} = 1$ must be satisfied for one-to-one matching. These constraints however, do not apply for missing matches since multiple candidates may correspond to the same dustbin. Therefore, we make use of two new constraints for dustbins. These constraints for dustbins read as follows: all candidates not matched to another candidate must be matched to the dustbins. Mathematically, this can be expressed as, $\sum_j \boldsymbol{A}_{N'+1,j} = N - M$ and $\sum_i \boldsymbol{A}_{i,N+1} = N' - M$, where $M = \sum_{(i \leq N', j \leq N)} \boldsymbol{A}_{i,j}$ represents the number of candidate-to-candidate matching. In order to solve the association problem, using the discussed constraints, we follow Sarlin *et al.* [38] and use the Sinkhorn [39, 8] based algorithm therein.

3.6. Learning Candidate Association

Training the embedding network that parameterizes the similarity matrix used for optimal matching requires ground truth assignments. Hence, in the domain of sparse keypoint matching, recent learning based approaches leverage large scale datasets [19, 38] such as MegaDepth [36] or ScanNet [9], that provide ground truth matches. However, in tracking such ground truth correspondences (between distractor objects) are not available. Only the target object and its location provide a single ground truth correspondence. Manually annotating correspondences for distracting candidates, identified by a tracker on video datasets, is expensive and may not be very useful. Instead, we propose a novel training approach that exploits, (i) partial supervision from the annotated target objects, and (ii) self-supervision by artificially mimicking the association problem. Our approach requires only the annotations that already exist in standard tracking datasets. The candidates for matching are obtained by running the base tracker on the given training dataset.

Partially Supervised Loss: For each pair of consecutive frames, we retrieve the two candidates corresponding to the annotated target, if available. This correspondence forms a partial supervision for a single correspondence while all other associations remain unknown. For the retrieved candidates v'_i and v_j, we define the association as a tuple $(l', l) = (i, j)$. Here, we also mimic the association for redetections and occlusions by occasionally excluding one of the corresponding candidates from $\mathcal{V}'$ or $\mathcal{V}$. We replace the excluded candidate by the corresponding dustbin to form the correct association for supervision. More precisely, the simulated associations for redetection and occlusion are expressed as, $(l', l) = (N' + 1, j)$ and $(l', l) = (i, N + 1)$, respectively. The supervised loss, for each frame-pairs, is then given by the negative log-likelihood of the assignment probability,

$$L_{\text{sup}} = -\log \boldsymbol{A}_{l',l}. \tag{2}$$

Self-Supervised Loss: To facilitate the association of distractor candidates, we employ a self-supervision strategy. The proposed approach first extracts a set of candidates $\mathcal{V}'$ from any given frame. The corresponding candidates for matching, say $\mathcal{V}$, are identical to $\mathcal{V}'$ but we augment its features. Since the feature augmentation does not affect the associations, the initial ground-truth association set is given by $\mathcal{C} = \{(i, i)\}_{i=1}^{N}$. In order to create a more challenging learning problem, we simulate occlusions and redetections as described above for the partially supervised loss. Note that the simulated occlusions and redetections change the entries of $\mathcal{V}$, $\mathcal{V}'$, and $\mathcal{C}$. We make use of the same notations with slight abuse for simplicity. Our feature augmentation involves, randomly translating the location $\mathbf{c}_i$, increasing or decreasing the score s_i, and transforming the given image before extracting the visual features $\mathbf{f}_i$. Now, using the simulated ground-truth associations $\mathcal{C}$, our self-supervised loss is given by,

$$L_{\text{self}} = \sum_{(l',l) \in \mathcal{C}} -\log \boldsymbol{A}_{l',l}. \tag{3}$$

Finally, we combine both losses as $L_{\text{tot}} = L_{\text{sup}} + L_{\text{self}}$. It is important to note that the real training data is used only for the former loss function, whereas synthetic data is used only for the latter one.

Data Mining: Most frames contain a candidate corresponding to the target object and are thus applicable for supervised training. However, a majority of these frames are not very informative for training because they contain only a single candidate with high target classifier score and correspond to the target object. Conversely, the dataset contains adverse situations where associating the candidate corresponding to the target object is very challenging. Such situations include sub-sequences with different number of candidates, with changes in appearance or large motion between frames. Thus, sub-sequences where the appearance model either fails and starts to track a distractor or when the tracker is no longer able to detect the target with sufficient confidence are valuable for training. However, such failure cases are rare even in large scale datasets. Similarly, we prefer frames with many target candidates when creating synthetic sub-sequences to simultaneously include candidate associations, redetections and occlusions. Thus, we mine the training dataset using the dumped predictions of the base tracker to use more informative training samples.

Training Details: We first retrain the base tracker SuperDiMP without the learned discriminative loss parameters but keep everything else unchanged. We split the LaSOT training set into a *train-train* and a *train-val* set. We run the base tracker on all sequences and save the search region and score map for each frame on disk. We use the dumped data to mine the dataset and to extract the target candidates and its features. We freeze the weights of the base tracker during training of the proposed network and train for 15 epochs by sampling 6400 sub-sequences per epoch from the *train-train* split. We sample real or synthetic data equally likely. We use ADAM [30] with learning rate decay of 0.2 every 6*th* epoch with a learning rate of 0.0001. We use two GNN Layers and run 10 Sinkhorn iterations. Please refer to the supplementary for additional details about training.

3.7. Object Association

This part focuses on using the estimated assignments (see Sec. 3.5) in order to determine the object correspondences during online tracking. An object corresponds either to the target or a distractor. The general idea is to keep track of every object present in each scene over time. We implement this idea with a database $\mathcal{O}$, where each entry corresponds to an object o that is visible in the current frame. Fig. 3 shows these objects as circles. An object disappears from the scene if none of the current candidates is associated with it, *e.g.*, in Fig. 3 the purple and pink objects (●, ●) no longer correspond to a candidate in the last frame. Then, we delete this object from the database. Similarly, we add a new object to the database if a new target candidate appears (●, ●, ●). When initializing a new object, we assign it a new object-id (not used previously) and the score s_i. In Fig. 3 object-ids are represented using colors. For objects that remain visible, we add the score s_i of the corresponding candidate to the history of scores of this object. Furthermore, we delete the old and create a new object if the candidate correspondence is ambiguous, *i.e.*, the assignment probability is smaller than $\omega = 0.75$.

If associating the target object $\hat{o}$ across frames is unambiguous, it implies that one object has the same object-id as the initially provided object $\hat{o}_{\text{init}}$. Thus, we return this object as the selected target. However, in real world scenarios the target object gets occluded, leaves the scene or associating the target object is ambiguous. Then, none of

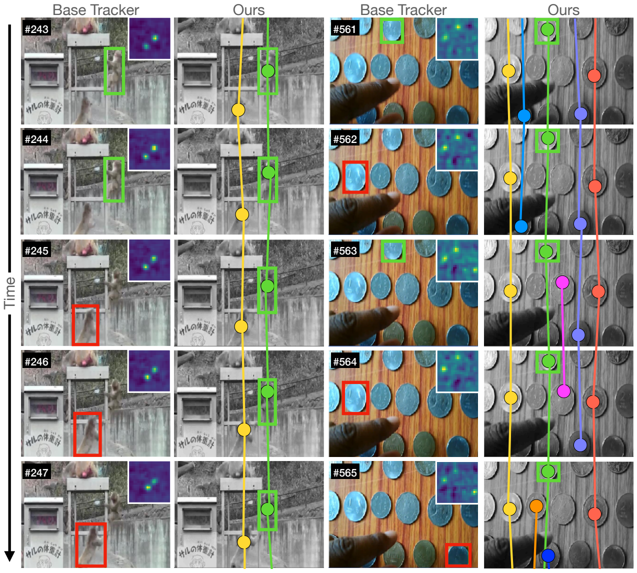


Figure 3. Visual comparison of the base tracker and our tracker. The bounding boxes represent the tracker result, green [🟩] indicates correct detections and red [🟥] refers to tracker failure. Each circle represents an object. Circles with the same color are connected to indicate that the object-ids are identical. If a target candidate cannot be matched with an existing object we add a new object (●, ●, ●). Similarly, we delete the object if no candidate corresponds to it anymore in the next frame (●, ●, ●).

the candidates corresponds to the sought target and we need to redetect. We redetect the target if the candidate with the highest target classifier score achieves a score that exceeds the threshold $\eta = 0.25$. We select the corresponding object as the target as long as no other candidate achieves a higher score in the current frame. Then, we switch to this candidate and declare it as target if its score is higher than any score in the history (of the currently selected object). Otherwise, we treat this object as a distractor for now, but if its score increases high enough, we will select it as the target object in the future. Please refer to the supplementary material for the detailed algorithm.

3.8. Memory Sample Confidence

While updating the tracker online is often beneficial, it is disadvantageous if the training samples have a poor quality. Thus, we describe a memory sample confidence score, that we use to decide which sample to keep in the memory and which should be replaced when employing a fixed size memory. In addition, we use the score to control the contribution of each training sample when updating the tracker online. In contrast, the base tracker replaces frames using a first-in-first out policy if the target was detected and weights samples during inference solely based on age.

First, we define the training samples in frame k as (x_k, y_k). We assume a memory size m that stores samples from frame $k \in \{1, \ldots, t\}$, where t denotes the current frame number. The online loss then given by,

$$J(\theta) = \lambda R(\theta) + \sum_{k=1}^{t} \alpha_k \beta_k Q(\theta; x_k, y_k), \tag{4}$$

where Q denotes the data term, R the regularisation term, λ is a scalar and θ represents appearance model weights. The weights $\alpha_k \geq 0$ control the impact of the sample from frame k, *i.e.*, a higher value increases the influence of the corresponding sample during training. We follow other appearance based trackers [3, 12] and use a learning parameter $\gamma \in [0, 1]$ in order to control the weights $\alpha_k = (1-\gamma)\alpha_{k+1}$, such that older samples achieve a smaller value and their impact during training decreases. In addition, we propose a second set of weights β_k that represent the confidence of the tracker that the predicted label y_k is correct. Instead of removing the oldest samples to keep the memory fixed [3], we propose to drop the sample that achieves the smallest score $\alpha_k \beta_k$ which combines age and confidence. Thus, if $t > n$ we remove the sample at position $k = \text{argmin}_{1 \leq k \leq n} \alpha_k \beta_k$ by setting $\alpha_k = 0$. This means, that if all samples achieve similar confidence the oldest is replaced, or that if all samples are of similar age the least confident sample is replaced.

We describe the extraction of the confidence weights as,

$$\beta_t = \begin{cases} \sqrt{\sigma}, & \text{if } \hat{o} = \hat{o}_{\text{init}} \\ \sigma, & \text{otherwise,} \end{cases} \tag{5}$$

where $\sigma = \max_i s_i^t$ denotes the maximum value of the target classifier score map of frame t. For simplicity, we assume that $\sigma \in [0, 1]$. The condition $\hat{o} = \hat{o}_{\text{init}}$ is fulfilled if the currently selected object is identical to the initially provided target object, *i.e.*, both objects share the same object id. Then, it is very likely, that the selected object corresponds to the target object such that we increase the confidence using the square root function that increases values in the range $[0, 1)$. Hence, the described confidence score combines the confidence of the target classifier with the confidence of the object association module, but fully relies on the target classifier once the target is lost.

Inference details: We propose *KeepTrack* and the speed optimized *KeepTrackFast*. We use the SuperDiMP parameters for both trackers but increase the search area scale from 6 to 8 (from 352 to 480 in image space) for *KeepTrack*. For the fast version we keep the original scale but reduce the number of bounding box refinement steps from 10 to 3. In addition, we skip running the association module if only one target caidndate with a high score is present in the previous and current frame. Overall, both trackers follow the target longer until it is lost such that small search areas occur frequently. Thus, we reset the search area to its previous size if it was drastically decreased before the target was lost, to facilitate redetections. Please refer to the supplementary for more details.

4. Experiments

We evaluate our proposed tracking architecture on seven benchmarks. Our approach is implemented in Python using PyTorch. On a single Nvidia GTX 2080Ti GPU, *KeepTrack* and *KeepTrackFast* achieve 18.3 and 29.6 FPS, respectively.

4.1. Ablation Study

We perform an extensive analysis of the proposed tracker, memory sample confidence, and training losses.

Online tracking components: We study the importance of memory sample confidence, the search area protocol, and target candidate association of our final method *KeepTrack*. In Tab. 1 we analyze the impact of successively adding each component, and report the average of five runs on the NFS [24], UAV123 [37] and LaSOT [21] datasets. The first row reports the results of the employed base tracker. First, we add the memory sample confidence approach (Sec. 3.8), observe similar performance on NFS and UAV but a significant improvement of 1.5% on LaSOT, demonstrating its potential for long-term tracking. With the added robustness, we next employ a larger search area and increase it if it was drastically shrank before the target was lost. This leads to a fair improvement on all datasets. Finally, we add the target candidate association network, which provides substantial performance improvements on all three datasets, with a $+1.3\%$ AUC on LaSOT. These results clearly demonstrate the power of the target candidate association network.

Training: In order to study the effect of the proposed training losses, we retrain the target candidate association network either with only the partially supervised loss or only the self-supervised loss. We report the performance on LaSOT [21] in Tab. 2. The results show that each loss individually allows to train the network and to outperform the baseline without the target candidate association network (no TCA), whereas, combining both losses leads to the best tracking results. In addition, training the network with the combined loss but without data-mining decreases the tracking performance.

Memory Sample Confidence	Search area Adaptation	Target Candidate Association Network	NFS	UAV123	LaSOT
–	–	–	64.4	68.2	63.5
✓	–	–	64.7	68.0	65.0
✓	✓	–	65.2	69.1	65.8
✓	✓	✓	**66.4**	**69.7**	**67.1**

Table 1. Impact of each component in terms of AUC (%) on three datasets. The first row corresponds to our SuperDiMP baseline.

Loss	no TCA	L_{sup}	L_{self}	$L_{\text{sup}} + L_{\text{self}}$	$L_{\text{sup}} + L_{\text{self}}$
Data-mining	n.a.	✓	✓	-	✓
LaSOT, AUC (%)	65.8	66.0	66.9	66.8	**67.1**

Table 2. Analysis on LaSOT of association learning using different loss functions with and without data-mining.

Sample Replacement with conf. score	Online updating with conf. score	Conf. score threshold	LaSOT AUC (%)
–	–	–	63.5
✓	–	–	64.1
✓	✓	0.0	64.6
✓	✓	0.5	65.0

Table 3. Analysis of our memory weighting component on LaSOT.

Memory management: We not only use the sample confidence to manage the memory but also to control the impact of samples when learning the target classifier online. In Tab. 3, we study the importance of each component by adding one after the other and report the results on LaSOT [21]. First, we use the sample confidence scores only to decide which sample to remove next from the memory. This, already improves the tracking performance. Reusing these weights when learning the target classifier as described in Eq. (4) increases the performance again. To suppress the impact of poor-quality samples during online learning, we ignore samples with a confidence score bellow 0.5. This leads to an improvement on LaSOT. The last row corresponds to the used setting in the final proposed tracker.

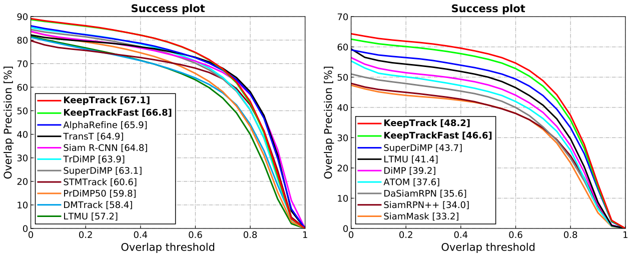
4.2. State-of-the-art Comparison

We compare our approach on seven tracking benchmarks. The same settings and parameters are used for all datasets. In order to ensure the significance of the results, we report the average over five runs on all datasets unless the evaluation protocol requires otherwise. We recompute the results of all trackers using the raw predictions if available or otherwise report the results given in the paper.

LaSOT [21]: First, we compare on the large-scale LaSOT dataset (280 test sequences with 2500 frames in average) to demonstrate the robustness and accuracy of the proposed tracker. The success plot in Fig. 4a shows the overlap precision OP_T as a function of the threshold T. Trackers are ranked w.r.t. their *area-under-the-curve* (AUC) score, denoted in the legend. Tab. 4 shows more results including precision and normalized precision. *KeepTrack* and *KeepTrackFast* outperform the recent trackers AlphaRefine [47], TransT [6] and TrDiMP [43] by a large margin and the base tracker SuperDiMP by 4.0% or 3.7% in AUC. The improvement in OP_T is most prominent for thresholds $T < 0.7$, demonstrating the superior robustness of our tracker. In Tab. 5, we further perform an apple-to-apple comparison with KYS [4], LTMU [10], AlphaRefine [47] and TrDiMP [43], where all methods use SuperDiMP as base tracker. We outperform the best method on each metric, achieving an AUC improvement of 1.8%.

	Keep Track	**Keep Track Fast**	Alpha Refine [47]	TransT [6]	Siam R-CNN [42]	TrDiMP [43]	Super Dimp [11]	STM Track [23]	Pr DiMP [16]	DM Track [55]	LTMU [10]	DiMP [3]	Ocean [54]
Precision	**70.2**	**70.0**	68.0	69.0	68.4	66.3	65.3	63.3	60.8	59.7	57.2	56.7	56.6
Norm. Prec	**77.2**	**77.0**	73.2	73.8	72.2	73.0	72.2	69.3	68.8	66.9	66.2	65.0	65.1
Success (AUC)	**67.1**	**66.8**	65.3	64.9	64.8	63.9	63.1	60.6	59.8	58.4	57.2	56.9	56.0

Table 4. State-of-the-art comparison on the LaSOT [21] test set in terms of AUC score.



(a) LaSOT [21]
(b) LaSOTExtSub [20]

Figure 4. Success plots, showing OP_T, on LaSOT [21] and LaSOTExtSub [20]. Our approach outperforms all other methods by a large margin in AUC, reported in the legend.

	KeepTrack	**KeepTrack**	AlphaRefine	LTMU	TrDiMP	KYS	SuperDiMP
		Fast	[47]	[10]	[43]	[4]	[16]
Precision	**70.2**	**70.0**	68.0	66.5	61.4	64.0	65.3
Norm. Prec.	**77.2**	**77.0**	73.2	73.7	–	70.7	72.2
Success (AUC)	**67.1**	**66.8**	65.3	64.7	63.9	61.9	63.1

Table 5. Results on the LaSOT [21] test set. All trackers use the same base tracker SuperDiMP [11].

LaSOTExtSub [20]: We evaluate our tracker on the recently published extension subset of LaSOT. LaSOTExtSub is meant for testing only and consists of 15 new classes with 10 sequences each. The sequences are long (2500 frames on average), rendering substantial challenges. Fig. 4b shows the success plot, that is averaged over 5 runs. All results, except ours and SuperDiMP, are obtained from [20], *e.g.*, DaSiamRPN [56], SiamRPN++ [34] and SiamMask [44]. Our method achieves superior results, outperforming LTMU by 6.8% and SuperDiMP by 3.5%.

OxUvALT [41]: The OxUvA long-term dataset contains 166 test videos with average length 3300 frames. Trackers are required to predict whether the target is present or absent in addition to the bounding box for each frame. Trackers are ranked by the maximum geometric mean (MaxGM) of the true positive (TPR) and true negative rate (TNR). We use the proposed confidence score and set the threshold for target presence using the separate dev. set. Tab. 6 shows the results on the test set, which are obtained through the evaluation server. *KeepTrack* sets the new state-of-the-art in terms of MaxGM by achieving an improvement of 5.8% compared to the previous best method and exceed the result of the base tracker SuperDiMP by 6.1%.

VOT2019LT [32]/VOT2020LT [31]: The dataset for both VOT [33] long-term tracking challenges contains 215,294 frames divided in 50 sequences. Trackers need to predict a confidence score that the target is present and the bounding box for each frame. Trackers are ranked by the F-score, evaluated for a range of confidence thresholds. We compare with the top methods in the challenge [32, 31], as well as more recent methods. As shown in Tab. 7, our tracker achieves the best result in terms of F-score and outperforms the base tracker SuperDiMP by 4.0% in F-score.

	Keep	**Keep**		Super	Siam		DM		Global		Siam	
	Track	**Track**	LTMU	DiMP	R-CNN	TACT	Track	SPLT	Track	MBMD	FC+R	TLD
		Fast	[10]	[11]	[42]	[7]	[55]	[48]	[28]	[53]	[41]	[29]
TPR	80.6	**82.7**	74.9	79.7	70.1	**80.9**	68.6	49.8	57.4	60.9	42.7	20.8
TNR	**81.2**	77.2	75.4	70.2	74.5	62.2	69.4	77.6	63.3	48.5	48.1	**89.5**
MaxGM	**80.9**	**79.9**	75.1	74.8	72.3	70.9	68.8	62.2	60.3	54.4	45.4	43.1

Table 6. Results on the OxUvALT [41] test set in terms of TPR, TNR, and the max geometric mean (MaxGM) of TPR and TNR.

	Keep	**Keep**			Mega		RLT	Super	Siam	
	Track	**Track**	LT_DSE	LTMU_B	track	CLGS	DiMP	DiMP	DW_LT	ltMBNet
		Fast	[32, 31]	[10, 31]	[31]	[32, 31]	[31]	[11]	[32, 31]	[31]
Precision	**72.3**	70.6	71.5	70.1	70.3	**73.9**	65.7	67.6	69.7	64.9
Recall	**69.7**	68.0	67.7	68.1	67.1	61.9	68.4	66.3	63.6	51.4
F-Score	**70.9**	69.3	**69.5**	69.1	68.7	67.4	67.0	66.9	66.5	57.4

Table 7. Results on the VOT2019LT [32]/VOT2020LT [31] dataset in terms of F-Score, Precision and Recall.

	Keep	**Keep**				Super	Pr	STM	Siam	Siam		
	Track	**Track**	CRACT	TrDiMP	TransT	DiMP	DiMP	Track	AttN	R-CNN	KYS	DiMP
		Fast	[22]	[43]	[6]	[11]	[16]	[23]	[50]	[42]	[4]	[3]
UAV123	**69.7**	**69.5**	66.4	67.5	69.1	68.1	68.0	64.7	65.0	64.9	–	65.3
OTB-100	70.9	71.2	**72.6**	71.1	69.4	70.1	69.6	**71.9**	71.2	70.1	69.5	68.4
NFS	**66.4**	65.3	62.5	**66.2**	65.7	64.7	63.5	–	–	63.9	63.5	62.0

Table 8. Comparison with state-of-the-art on the OTB-100 [45], NFS [24] and UAV123 [37] datasets in terms of AUC score.

UAV123 [37]: This dataset contains 123 videos and is designed to assess trackers for UAV applications. It contains small objects, fast motions, and distractor objects. Tab. 8 shows the results, where the entries correspond to AUC for OP_T over IoU thresholds T. Our method sets a new state-of-the-art with an AUC of 69.7%, exceeding the performance of the recent trackers TransT [6] and TrDiMP [43] by 0.6% and 2.2% in AUC.

OTB-100 [45]: For reference, we also evaluate our tracker on the OTB-100 dataset consisting of 100 sequences. Several trackers achieve tracking results over 70% in AUC, as shown in Tab. 8. So do *KeepTrack* and *KeepTrackFast* that perform similarly to the top methods, with a 0.8% and 1.1% AUC gain over the SuperDiMP baseline.

NFS [24]: Lastly, we report results on the 30 FPS version of the Need for Speed (NFS) dataset. It contains fast motions and challenging distractors. Tab. 8 shows that our approach sets a new state-of-the-art on NFS.

5. Conclusion

We propose a novel tracking pipeline employing a learned target candidate association network in order to track both the target and distractor objects. This approach allows us to propagate the identities of all target candidates throughout the sequence. In addition, we propose a training strategy that combines partial annotations with self-supervision. We do so to compensate for lacking ground-truth correspondences between distractor objects in visual tracking. We conduct comprehensive experimental validation and analysis of our approach on seven generic object tracking benchmarks and set a new state-of-the-art on six.

Acknowledgments: This work was partly supported by the ETH Zürich Fund (OK), Siemens Smart Infrastructure, the ETH Future Computing Laboratory (EFCL) financed by a gift from Huawei Technologies, an Amazon AWS grant, and an Nvidia hardware grant.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[2] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, October 2016. 1

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 6, 7, 8

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020. 1, 2, 7, 8

[5] Guillem Braso and Laura Leal-Taixe. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[6] 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)*, June 2021. 7, 8

[7] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Visual tracking by tridentalign and context embedding. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. 8

[8] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, December 2013. 4

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4

[10] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 7, 8

[11] Martin Danelljan and Goutam Bhat. PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking, 2019. Accessed: 1/08/2020. 2, 3, 7, 8

[12] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 6

[13] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2017. 1, 2

[14] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[15] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2016. 2

[16] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3, 7, 8

[17] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision (IJCV)*, 129(4):1–37, 2020. 2

[18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2018. 3, 4

[19] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4

[20] Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Mingzhen Huang, Juehuan Liu, Yong Xu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision (IJCV)*, 129(2):439–461, 2021. 8

[21] 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. 1, 2, 7, 8

[22] Heng Fan and Haibin Ling. Cract: Cascaded regression-align-classification for robust visual tracking. *arXiv preprint arXiv:2011.12483*, 2020. 8

[23] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. Stmtrack: Template-free visual tracking with space-time memory networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 7, 8

[24] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017. 7, 8

[25] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual

tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2017. 2

[26] Fredrik K Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B Schön. How to train your energy-based model for regression. In *Proceedings of the British Machine Vision Conference (BMVC)*, September 2020. 3

[27] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(3):583–596, 2015. 2

[28] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Globaltrack: A simple and strong baseline for long-term tracking. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, February 2020. 8

[29] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(7):1409–1422, 2012. 8

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 5

[31] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernández, and et al. The eighth visual object tracking vot2020 challenge results. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, August 2020. 8

[32] Matej Kristan, Jirí Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Luka Cehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, Abdelrahman Eldesokey, Jani Käpylä, Gustavo Fernández, and et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, October 2019. 8

[33] 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 (TPAMI)*, 38(11):2137–2155, 2016. 8

[34] 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. 1, 8

[35] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[36] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4

[37] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2016. 2, 7, 8

[38] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3, 4

[39] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. 4

[40] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[41] Jack Valmadre, Luca Bertinetto, João F. Henriques, Ran Tao, Andrea Vedaldi, Arnold W.M. Smeulders, Philip H.S. Torr, and Efstratios Gavves. Long-term tracking in the wild: a benchmark. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1, 8

[42] Paul Voigtlaender, Jonathon Luiten, Philip H.S. Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by re-detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 7, 8

[43] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 7, 8

[44] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H.S. Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 8

[45] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1834–1848, 2015. 8

[46] Jingjing Xiao, Linbo Qiao, Rustam Stolkin, and Ale*vs*. Leonardis. Distractor-supported single target tracking in extremely cluttered scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2016. 1, 2

[47] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 7, 8

[48] Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. 'skimming-perusal' tracking: A framework for real-time and robust long-term tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 8

[49] Qian Yu, Gérard Medioni, and Isaac Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007. 2

[50] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R. Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 8

[51] Jianming Zhang, Shugao Ma, and Stan Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2014. 2

[52] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008. 2

[53] Yunhua Zhang, Lijun Wang, Dong Wang, Jinqing Qi, and Huchuan Lu. Learning regression and verification networks for long-term visual tracking. *International Journal of Computer Vision (IJCV)*, 129(9):2536–2547, 2021. 8

[54] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020. 7

[55] 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)*, June 2021. 2, 7, 8

[56] Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1, 2, 8