

Article

Iterative Multiple Bounding-Box Refinements for Visual Tracking

Giorgio Cruciata , Liliana Lo Presti *  and Marco La Cascia * 

Dipartimento di Ingegneria, University of Palermo, 90128 Palermo, Italy; giorgio.cruciata@unipa.it

* Correspondence: liliana.lopresti@unipa.it (L.L.P.); marco.lacascia@unipa.it (M.L.C.)

Abstract: Single-object visual tracking aims at locating a target in each video frame by predicting the bounding box of the object. Recent approaches have adopted iterative procedures to gradually refine the bounding box and locate the target in the image. In such approaches, the deep model takes as input the image patch corresponding to the currently estimated target bounding box, and provides as output the probability associated with each of the possible bounding box refinements, generally defined as a discrete set of linear transformations of the bounding box center and size. At each iteration, only one transformation is applied, and supervised training of the model may introduce an inherent ambiguity by giving importance priority to some transformations over the others. This paper proposes a novel formulation of the problem of selecting the bounding box refinement. It introduces the concept of non-conflicting transformations and allows applying multiple refinements to the target bounding box at each iteration without introducing ambiguities during learning of the model parameters. Empirical results demonstrate that the proposed approach improves the iterative single refinement in terms of accuracy and precision of the tracking results.

Keywords: visual tracking; deep tracking; iterative bounding box refinement



Citation: Cruciata, G.; Lo Presti, L.; La Cascia, M. Iterative Multiple Bounding-Box Refinements for Visual Tracking. *J. Imaging* **2022**, *8*, 61. <https://doi.org/10.3390/jimaging8030061>

Academic Editor: Marco Porta

Received: 20 December 2021

Accepted: 22 February 2022

Published: 3 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Visual object tracking aims to automatically locate a target in subsequent frames, generally by estimating the bounding box that encloses the target on the image plane [1]. In contrast to the object detection problem, where instances of predefined object classes are located on an image, in object tracking the target is often located in a class-agnostic way by considering only the information provided in an initial frame (for instance, the frame where the target first appears). Although it has been widely studied, visual tracking remains a challenging problem in real-world scenarios due to target occlusions, pose and appearance changes, and illumination variations [2].

Nowadays, the best performance in visual tracking is achieved by employing deep learning [2,3]. MDNet, proposed by the authors of [4], is a tracking-by-detection and regression algorithm that classifies into target/background a set of bounding boxes sampled every frame around the last known target location. The bounding box with the highest classification confidence score is later adjusted through regression. The deep classification model is trained in a multi-domain way. Despite the fact that MDNet is not the most recent tracker, it still achieves state-of-the-art performance on the famous OTB benchmark [5]. MDNet has two main limitations: One is related to the sampling and classification at each frame of several bounding boxes to select the optimal one; the other limitation is related to the use of a regression model to refine the selected bounding box.

Recent works have tried to improve MDNet by formulating the search of the optimal bounding box either as an iterative process where a discrete sequence of bounding box refinements is predicted to locate the target, as done in [6–10], or as the problem of regressing the bounding box coordinates at each frame as in [11].

In [6,12,13], at each frame, the bounding box is iteratively refined by applying shift (move right/left/up/down on the image plane) and scale (reduce/enlarge the bounding box size) refinements. The identity transformation is also included to account for the cases in which the bounding box must be accepted as it is. To implement such a strategy, the deep model takes as input the image patch corresponding to the currently estimated target bounding box, and provides as output the probability associated with each of the possible bounding box refinements. The one with the highest probability value is applied to the bounding box, and the process is iterated until a maximal number of iterations or the null transformation is selected. Figure 1A summarizes the process.

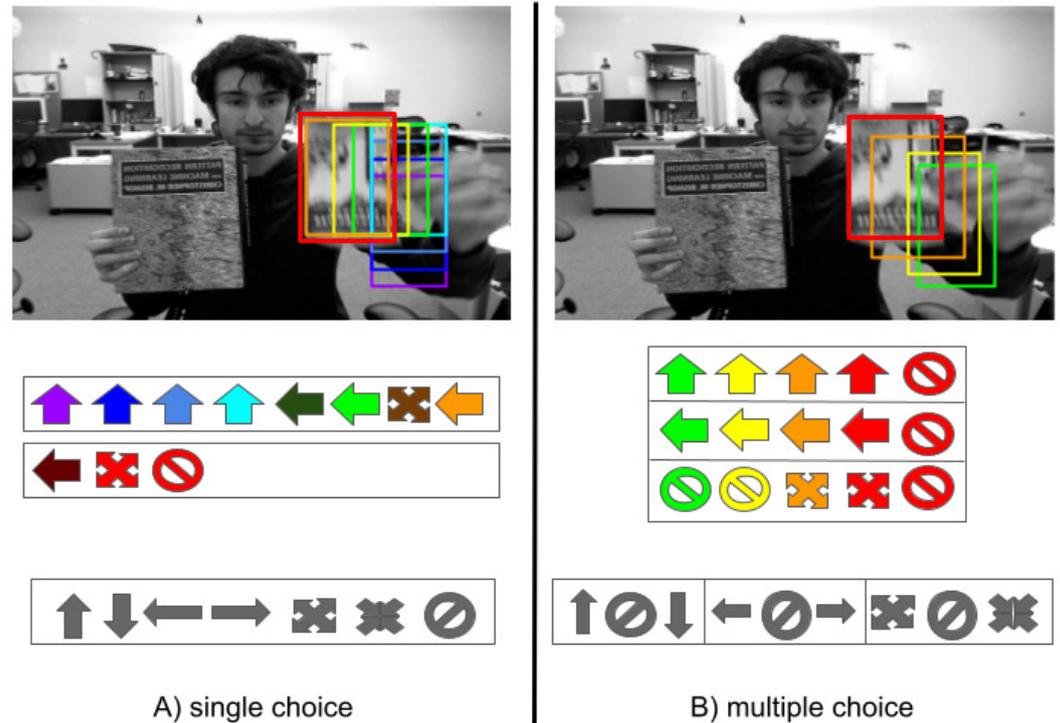


Figure 1. (A) In iterative single refinement, starting from an initial bounding box, a sequence of transformations to the bounding box is predicted and applied to locate the target. (B) With multiple refinements, a sequence of multiple non-conflicting transformations is predicted and applied.

This paper focuses on these kinds of approaches. We note that the above iterative process has several drawbacks. First, only one transformation can be applied at each iteration, thus leading to a higher computational cost to find the optimal bounding box. Second, the strategy introduces an ambiguity during the learning of the model parameters. Indeed, supervised training of the model is performed by providing the target patch and the type of bounding box refinement that should be applied to improve the tracking result. As shown in Figure 2, the effect of applying a transformation can be measured by estimating the intersection-over-union (IoU) value of the resulting bounding box and the ground-truth. Very often, more than a transformation would result in improved target localization (in Figure 2 the transformations “move up”, “move left”, and “enlarge”), but this is generally ignored by the supervised training procedure in [6,7,13]. Indeed, when using the 1-hot encoding schema to select the refinement, it has been given priority importance to a bounding box refinement over another by considering the first transformation with the highest IoU. One of the main contributions of this work is to allow the selection of multiple transformations. During training, multiple 1-hot encoding is used to indicate the transformations that might be applied to improve the bounding box.

Finally, we note that bounding box transformations may cancel each other. We call such refinements “conflicting transformations” (i.e., shifting the bounding box to the

left and to the right). As reported in [13], refinements canceling each other may induce cycling behaviors.

In this paper, we deal with all the above issues by proposing a novel formulation of the problem of selecting the bounding box refinement (see Figure 1B). The contributions of this work can be summarized as follows:

- We introduce the novel concept of conflicting refinements and train the model to deal with them;
- We formulate the problem in such a way that multiple non-conflicting transformations can be applied at each iteration, leading to a speed up of the tracking process. As a consequence, a higher number of composite transformations can be handled by the model without any increases in its complexity;
- We limit the ambiguity during the training procedure by avoiding giving importance priority to some refinements over the others (see Figure 2).

To demonstrate our ideas empirically, we implemented our own tracking algorithm by adopting simple models and approaches common to several state-of-the-art tracking algorithms. We will make our tracker implementation publicly available to the scientific community. Our goal is not to achieve the highest accuracy and precision values in tracking. Instead, we aim to study the effect of formulating the problem of selecting the best target bounding box refinements in a different way.

In Section 2, we summarize the main characteristics of deep tracking approaches. In Section 3, we present our novel formulation of the problem. In Section 4, we provide details about the implementation we used to demonstrate our ideas. Finally, in Section 5, we discuss our contribution and contrast it with the state-of-the-art on the OTB benchmark. We also report the performance of our tracker implementation on the VOT benchmarks for completeness of results. Finally, in Section 6, we present conclusions and future work.

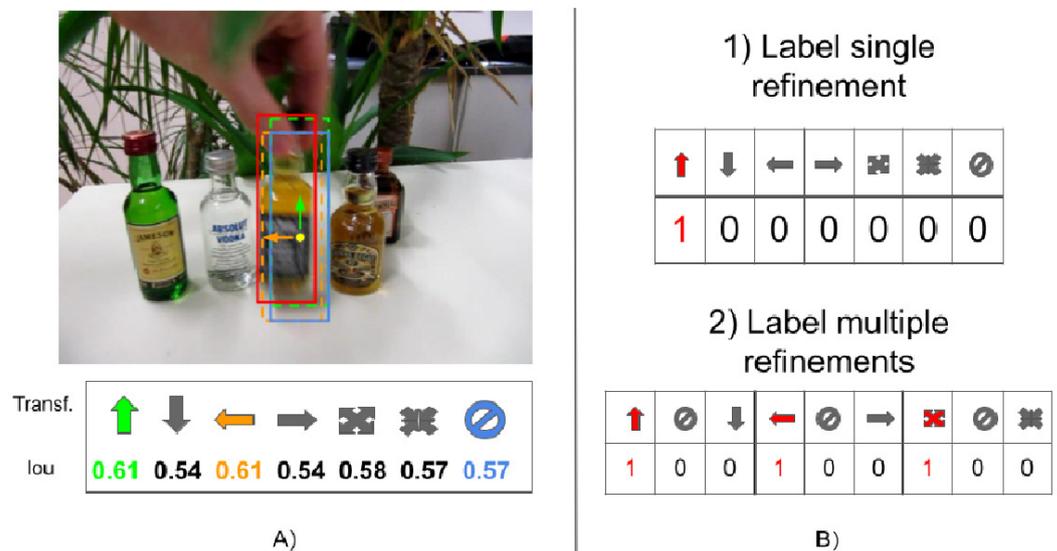


Figure 2. Image (A) shows the IoU values calculated after each of the possible refinements. Image (B) shows how 1-hot-encoding is used to annotate the data in single and multiple bounding box refinements.

2. Related Work

Given a sequence of T images $\{F_1; F_2; \dots; F_T\}$, visual tracking is the problem of detecting the location b_t on the image plane of a target moving in the environment over time or, more formally, estimating $b_t = (x_t; y_t)$ with $t \in [1; T]$ indexing the image frames. Often, b_t also includes the width w_t and height h_t of the bounding box enclosing the target on the image. Many methods have been proposed in literature for tracking by using traditional approaches like linear dynamical systems, Kalman Filter [14,15], re-identification and data

association [16,17], and correlation filter [18] together with hand-crafted features [2,19]. Recently, deep learning has proven to be particularly effective in extracting features for the recognition and detection of objects [20,21], and has been used to solve visual tracking problems [2]. In the following, we first briefly describe the different neural architectures and strategies commonly adopted in visual tracking. Then, we discuss methods that refine iteratively the target bounding box.

2.1. Network Architectures and Output

In general, it is possible to identify two categories of networks used in visual tracking: template-based and template-free neural networks. The former, such as the one proposed in [22], are composed of two or more branches that usually process the target template and the search area, and generally run in real-time without online parameter adaptation. These methods aim at locating the target depicted in the template image within the search area image. To account for target appearance changes, the template is in general updated online. The latter, such as the one proposed in [4], are composed of one branch. These methods do not need any templates but require online model adaptation to account for the target's appearance changes. As online parameter adaptation is computationally costly, these models are generally slower but more accurate.

Regardless of the adopted neural architecture, three main strategies are used: tracking-by-detection, tracking-by-regression, and tracking-by-detection and regression. Tracking-by-detection approaches [23,24] are classification methods, which aim to discriminate between target and background. A large number of candidate target bounding boxes are drawn around the last known target location. The one yielding to the highest classification score is selected. As a result, performance is closely linked to the sampling strategy.

Tracking-by-regression approaches [22,25] use regression to locate the bounding box in subsequent frames by minimizing an objective function such as least-squared error.

Tracking-by-detection and regression methods [4,6,26–28] are hybrid approaches whose goal is detecting the most similar region to the target, and then refining the region through regression.

Recent approaches [29–32] might be included in the class of tracking-by-segmentation. These methods aim at tracking objects while producing a mask of the target object for each frame of the processed video. The work in [33], which focuses on the transformation of appearance features into motion-attentive representations, is also related.

Our work focuses on tracking-by-detection and regression. The method refines the bounding box by discrete transformations as in [6,7]. As one refinement of the bounding box cannot be enough to detect the target, refinements are made in an iterative way on the same frame. At each iteration, the tracking model not only predicts the refinement to apply, but also classifies the bounding box image into target/background and returns a confidence score as in tracking-by-detection strategies. This score is used to detect drifting of the tracker and decide whether to adopt a re-detection method purely based on tracking-by-detection strategies. In this sense, at each iteration, the model has to decide whether to regress the bounding box (by applying discrete transformations) or re-detect the target (by classification of multiple candidate bounding boxes).

2.2. Iterative Bounding Box Refinements

When iterative bounding box refinements are used, at each iteration the candidate target bounding box is transformed such that the target object is progressively more and more at the center of the image patch [12,13]. Numerous works in tracking used this formulation to find the bounding box of the tracked object [6–10]. The problem can be formulated as a Markov Decision Process and, while reinforcement learning is often used, these models generally require a supervised training procedure.

Often, the set of possible refinements are treated as categories and the model predicts the probabilities associated with each of them. When preparing the training set, given a target bounding box and its perturbed version, it is necessary to associate the transformation

that the network has to predict when the perturbed bounding box is given in input. Among all transformations, the one maximizing the IoU of the transformed bounding box and the ground-truth is chosen.

In [13], the method for object detection in [12] is adapted to perform visual tracking. Given an initial detection of the target, a pre-trained deep network (derived from the VGG-16 model in [34]) is used to extract features from the bounding box. The deep model is trained to shift and scale the bounding box to recenter the target. It has been noted that the model may sometimes start to select pairs of bounding box transformations that cancel each other. To overcome this issue, the bounding box is randomly perturbed whenever cycling behavior is detected.

In the work proposed in [6], an action-decision network (ADNet) is used to predict a probability distribution over the possible bounding box transformations. The network takes as input the cropped target image and the history of the last 10 selected bounding box transformations represented by a one-hot encoding schema. The network is pre-trained in a supervised way and also estimates a confidence score to decide when the tracking process must be re-initialized. At test time, a supervised adaptation of the latest fully connected layers is performed to make the model more robust to appearance changes. ADNet was later improved in IADNet [35]. During training, a multi-domain learning strategy [4] is used. An online adaptive update strategy based on meta-learning is used to estimate the most appropriate model parameters such that the parameters are closer to the optimal ones.

The work proposed in [7], RDNet, adopts two Siamese network-based models receiving as input the crop of the image corresponding to the currently estimated bounding box and a search region image both obtained from the current frame. The first model chooses among different shifting transformations; the second one selects a scaling transformation. Similarly to RDNet, we also treat shifting and scaling transformations separately, but we adopt one multi-branch network and formulate the problem as one selecting multiple non-conflicting transformations.

The model TSAS proposed in [36] uses a cascade of two networks to predict how to shift the target bounding box. The first network, trained in a supervised way, predicts the best bounding box transformation to locate the target; the second network assesses the quality of the predicted transformation. Finally, a regressor is used to refine the resulting bounding box. In our model, one branch provides the confidence of the model on the current tracking result and no postprocessing of the bounding-box is used to improve the results.

3. Multi-Refinements of the Bounding Box

At time t , the candidate target bounding box is defined as a 4-dimensional vector $b_t = [x_t, y_t, w_t, h_t]$, where (x_t, y_t) is the center coordinates and w_t, h_t represent the width and height of the bounding box, respectively. Given an image frame F_t , the image patch p_t is obtained by cropping F_t based on the bounding box b_t through the selection patch function $f_b(\cdot)$. The function can also include pre-processing steps to adapt the resulting image patch to the network input:

$$p_t = f_b(b_t, F_t). \quad (1)$$

Let V be the number of basic linear transformations that can be used to refine the bounding box b . These transformations define the space Φ of allowed discrete bounding box transformations. Let φ be a subset of k transformations $\varphi = \{\phi_1, \phi_2, \dots, \phi_k\} \subseteq \Phi$. A transformation $\theta(\cdot) \in \Phi$ is *conflicting* with the transformations in φ if

$$b = \gamma_i(\theta(b)) \quad (2)$$

where γ_i indicates any sequence of transformations in φ . In other words, bounding box changes operated through the refinement θ are canceled out by some of the refinements in φ .

If instead, for any sequences γ_i

$$b \neq \gamma_i(\theta(b)), \tag{3}$$

then $\theta(\cdot)$ is *non-conflicting* with the transformations in φ .

Keeping in mind the above definitions, we propose to split the bounding box refinements into N groups such that each transformation in one group is not in conflict with the transformations in all the other ones. We include within each of these groups a “null transformation”, namely, the identity transformation. In the following, the obtained groups are named *non-conflicting* groups.

As an example, let us consider the following discrete bounding box transformations: $\Phi = \{Left(\Delta), Right(\Delta), Up(\Delta), Down(\Delta)\}$, where the name of the transformation indicates the shifting direction of the bounding box, while Δ represents the number of pixels. A possible partition into two non-conflicting groups, G_h and G_v , is $G_h = \{Left(\Delta), Right(\Delta)\}$ and $G_v = \{Up(\Delta), Down(\Delta)\}$.

Output Layer for Non-Conflicting Refinements

To deal with multiple non-conflicting refinements of the bounding box, we need to properly design the output layer of the deep model used within the tracking strategy. Figure 3 shows, on the top, the typical deep model used for iterative bounding box refinements [6,12,13]. At each iteration, the model takes as input an image patch and provides the probabilities of each of the V transformations $\phi_i \in \Phi$. Only the transformation with the highest probability is applied.

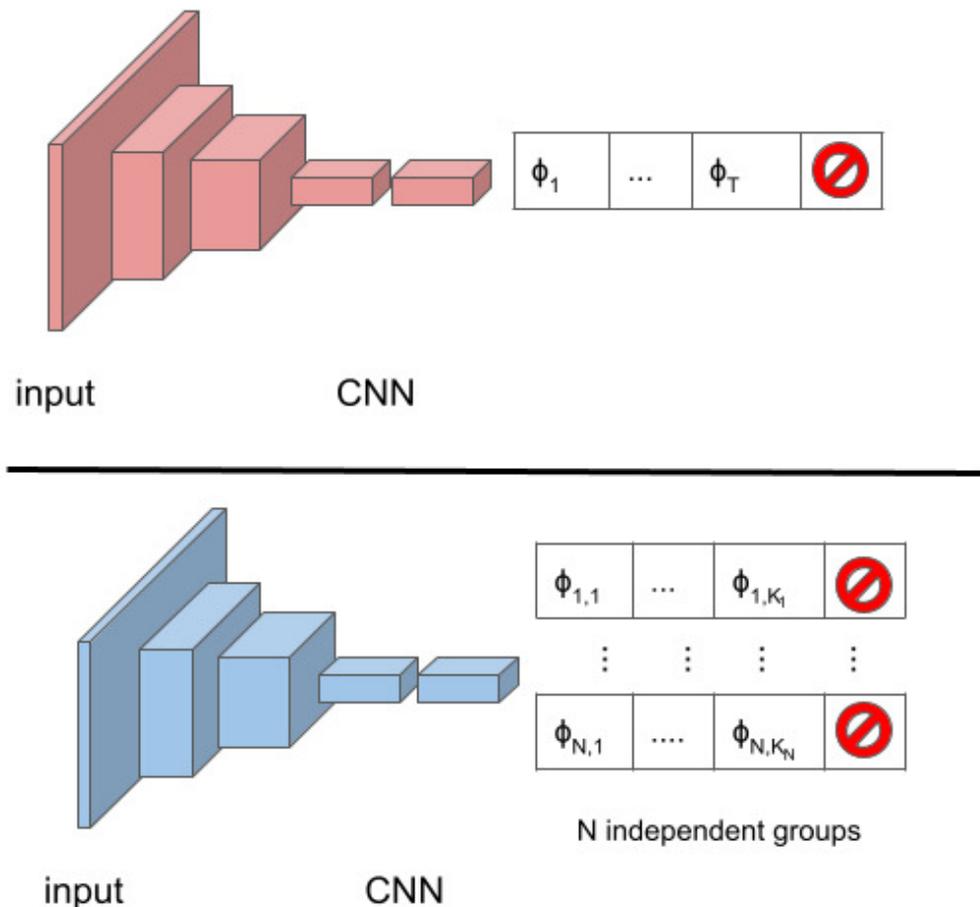


Figure 3. On the top, the model estimates the probability distribution over all the transformations. On the bottom, the model estimates N probability distributions, one for each group of transformations.

On the bottom, Figure 3 shows how the model needs be adapted to deal with N non-conflicting transformation groups. The network will provide N probability distributions over the k_i transformations belonging to each group, with i varying in $[1, N]$. The distributions are computed independently from each other, and N different non-conflicting transformations are applied, one from each group.

4. Tracking by Iterative Multi-Refinements

To demonstrate our idea, we implemented our own tracker. Our tracking architecture is template-free, and requires online fine-tuning of the parameters to adapt the model to the target appearance changes over time. As shown in Figure 4, our tracker takes advantage of one network with multiple output branches sharing the convolutional layers and the first dense layer. The first N output branches, namely, the subnet Transformation-Net, estimate N probability distributions over the refinements within the N non-conflicting groups of transformations. In the figure, $N = 3$. The last output branch network, namely, the subnet Confidence-Net, provides a confidence score of the classification of the image patch into background/target.

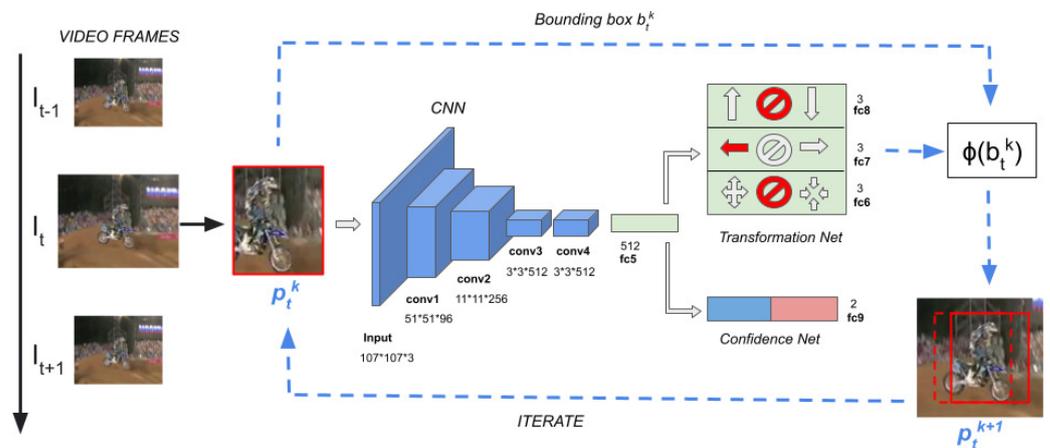


Figure 4. Architecture of our model. On the left, the arrow indicates the flow of frames. At time t , starting from the bounding box estimated at time $t - 1$, the method iterates a sequence of multiple transformations highlighted in red on the right side. The transformations are applied to the bounding box, which is used to get a new image patch to feed the model.

At time t , starting from the target bounding box estimated at time $t - 1$, our method uses the networks to estimate a sequence of linear transformations of the bounding box to locate the target in the image frame. The process ends when, for each non-conflicting transformation group, the null transformation is selected or a maximum number of iterations is reached.

The final confidence score is used to assess if the tracking failed, and in this case, a re-detection procedure based on particle filter is used. The entire tracking procedure is presented in Algorithm 1. In the following, we present details about the various steps performed by our tracker.

Algorithm 1 Online tracking algorithm.**Input:** Pre-trained CNN (w_1, \dots, w_5)Initial target b_0^* Sequence of V frames $\{F_0; F_1; \dots; F_t; \dots, F_V\}$ **Output:** Estimated sequence of target bounding-boxes $\{b_t^*\}_{t=1}^V$

```

1: Randomly initialize parameters  $w_6-w_9$ 
2: Draw training samples  $X_0$  around  $b_0^*$ ,  $M_S.push(X_0)$ ,  $M_L.push(X_0)$ 
3: Update parameters  $w_4-w_9$ 
4: for  $t = 1, \dots, V$  do
5:    $b_t^0 \leftarrow b_{t-1}^*$ 
6:   for  $k = 1, \dots, max\_iter$  do
7:      $p_t^k = f_b(b_t^{k-1}, F_t)$ 
8:     Select  $N$  refinements  $\varphi_i$  based on Transformation-Net( $p_t^k$ )
9:     Apply  $\varphi_i$  with  $i \in [1, N]$  to the bounding-box to estimate  $b_t^k$ 
10:   end for
11:   Evaluate confidence score  $q^* = \text{Confidence-Net}(f_b(b_t^k, F_t))$ 
12:   if  $q^* > 0.5$  then
13:      $b_t^* \leftarrow b_t^k$ 
14:     Draw sample  $X_t$  around  $b_t^*$ 
15:     Update  $M_S$  and  $M_L$  by adding  $X_t$  and limiting their size
16:   end if
17:   if  $q^* \leq 0.5$  then (failure!) apply Re-detection to find  $b_t^*$ 
18:     Evaluate confidence score  $q^* = \text{Confidence-Net}(f_b(b_t^*, F_t))$ 
19:   end if
20:   if  $q^* \leq 0.5$  then Update  $w_4 - w_9$  by using  $M_S$ 
21:   else if  $t \bmod 10 = 0$  then Update  $w_4 - w_9$  by using  $M_L$ 
22:   end if
23: end for

```

4.1. Designing Non-Conflicting Refinements

In our implementation, each basic transformation depends on parameters calculated from the size of the current bounding box, similarly to what has been done in [6]. In particular, at the k -th iteration, we consider the values $\Delta w_t^k = \sigma \cdot w_t^{k-1}$ and $\Delta h_t^k = \sigma \cdot h_t^{k-1}$, where σ is a constant value equals to 0.03. These parameters represent, in pixels, the adjustments to the center coordinates or the width/height of the bounding box. Based on our experiment, considering the structure of the adopted CNN (a VGG-M), σ equals to 0.03 is the minimum value for the network to notice the effect of the transformations to the bounding box.

We also consider $N = 3$ non-conflicting groups of refinements, each including two basic transformations. The first group, $G_h = \{Left(\Delta w_t^k), Right(\Delta w_t^k)\}$, shifts the bounding box to the left or to the right by adding/subtracting the value Δw_t^k to the coordinate x_t^{k-1} .

The second group, $G_v = \{Up(\Delta h_t^k), Down(\Delta h_t^k)\}$, shifts the bounding box up or down by adding/subtracting the value Δh_t^k to the coordinate y_t^{k-1} .

Finally, $G_s = \{Enlarge(\Delta w_t^k, \Delta h_t^k), Reduce(\Delta w_t^k, \Delta h_t^k)\}$, re-scales the bounding box to decrease or increase its size by adding/subtracting the value Δw_t^k and Δh_t^k to the bounding box width and height, respectively. Each of the above groups is augmented to also include the null transformation. These groups were created starting from the transformations used in ADNet [6], separating them into non-conflicting groups and excluding double shifts. The introduction of double shifts, namely, shifting transformations with a doubled value of σ , did not produce significant improvements in our experiments.

4.2. Deep Model

The structure of our network is described in Figure 4. Similarly to former approaches [4,6], the first four convolutional layers (conv1–conv4), with weights ($w_1 - w_4$), are taken from

the VGG-M network [37]. The input layer is adapted to our input dimension and subsequent feature maps; we used the same input size as in [4]. The fully-connected layer fc5, with weights w_5 , is shared by both the Confidence- and Transformation-Net. Layers fc6–fc8, with weights $(w_6 - w_8)$, provide the probability distributions for each of the transformation groups we defined and apply softmax activation functions. Layer fc9, with weights w_9 , belongs to the Confidence-Net; with 2 units, it applies a softmax activation function.

4.3. Supervised Training of the Model

Offline training: An offline training procedure is used to learn the parameters $(w_1 - w_5)$. This procedure is based on multi-domain learning [4] where layers (conv1–fc5) dealing with domain-independent information (such as motion blur, illuminations changes, and scale variations) and domain-specific layers (fc6–fc9) are treated differently. In particular, while the former are shared among all videos, the latter are initialized and trained for each video. The number of domains is equal to the number of videos contained in the training dataset. At each training iteration we use $X = [X_c, X_b]$, where X_c indicates the data used to train the Confidence-Net, and X_b indicates the data used to train the Transformation-Net. The entire model is trained by alternating the training of the Transformation-Net (while freezing the Confidence-Net) and the training of the Confidence-Net (while freezing the Transformation-Net).

Online training: At test time, parameters $(w_6 - w_9)$ are randomly initialized for each video sequence to be adapted online to the target appearance changes, and parameters $(w_1 - w_3)$ are fixed and not trained online to speed up the online training, and to limit overfitting of the network. Online parameter adaptation is done every s frames ($s = 10$) and, whenever a tracking failure has been detected, a re-detection step is performed. A failure is detected whenever the model predicts a confidence score lower than 0.5. Inspired by the works in [4,6], we update the parameters by using a long-term memory M_L every s frames. This memory stores random samples from the last $mem_L = 1000$ frames. In case of tracking failures, to speed up the model adaptation to the current target appearance, we update the parameters by using a short-term memory M_S . This memory stores random samples from the last $mem_S = 20$ frames.

4.4. Sample Generation

Considering N groups of k transformations (including the identity one), there are overall k^N composite transformations. In our implementation, $k^N = 3^3 = 27$. To train the Transformation-Net, we used balanced mini-batches of 81 samples where the 27 composite transformations were equally represented. A grid sampling approach over the 4-dimensional space with a discrete uniformly distributed random step has been used.

To generate balanced mini-batches of 64 samples for training the Confidence-Net, we referred mainly to the technique used by [4]. We used the sampling methods reported in the public code, which slightly differs from the one described in the paper. The sampling is based on normal distributions whose mean and variance depend on the bounding box estimated at the previous iteration. If the sample comes from the first frame, it is considered positive if it yields to $IoU > 0.7$. Otherwise, it is considered positive if the predicted confidence value is > 0.5 . Furthermore, we used hard negative sampling, meaning that we randomly selected a large number of negative samples from the short memory M_S and select 32 samples with the highest positive classification score. This procedure improves the discriminative abilities of the Confidence-Net.

4.5. Implementation Details

Whenever a failure occurs, similarly to the works in [4,6], we sample 256 candidate target bounding boxes with the same schema adopted to generate mini-batches for training the Confidence-Net. The confidence score of these bounding boxes is evaluated, and the candidate with the highest score is selected as the predicted target bounding box.

As loss function, we adopted the categorical cross-entropy. The learning rate is fixed to 0.001 and the network is trained with SGD with a momentum of 0.9 and weight decay parameter of 0.0005. During offline training of the model, we used 287 domains, that is, 287 videos from the ALOV300 dataset [38]. In particular, for a number of 3 iterations, we sampled 5 frames from each video. For each frame we trained the model for 5 iterations. Training has been done alternating the domains.

At the first frame, layers (conv4-fc9) are trained for 30 iterations on samples generated based on the known target. The Transformation-Net is initialized by sampling 6 times all the possible composite transformations, for a total of 162 samples for each iteration. Online adaptation of the parameters was done for 10 iterations. During tracking, at each frame, 15 negative samples and all transformed bounding boxes with a confidence score > 0.5 were stored in the long and short memories.

5. Experimental Results

Our goal, in this paper, is to demonstrate that iterative approaches to refine the bounding box have several drawbacks that can be overcome by allowing multiple non-conflicting refinements to the bounding box at each iteration. Therefore, we run two kinds of experiments: one to show the usefulness of our proposed approach, the other to compare our tracking strategy to state-of-the-art approaches on publicly available benchmarks. All experiments were conducted on a machine equipped with: 32 GB RAM, GPU RTX 2070 8GB RAM. Our prototype has been implemented in Python by using Tensorflow and runs at 5 fps on the GPU.

5.1. Single vs. Multiple Transformation Groups

We performed experiments by keeping the tracking strategy fixed and by varying the output layer of the deep model (single vs multiple transformation groups). As training of the model is important, we also test our approach by varying the model. To this purpose, we used the pre-trained parameters of the ADNet model [6], modified the last layer and compared single vs. multiple transformation groups. Finally, we tested how different training strategies of our model can affect the results. Experiments have been run on the OTB 100 benchmark [5].

All results are reported in Figure 5. Configurations with “NOT” directly use the initial parameters of VGG-M for the layers (conv1–4), while the fully connected layers (fc5–fc9) are initialized by random noises. Configurations with “ADNet” load the ADNet parameters.

For the above configurations, no offline training is performed and only online learning at test time is done.

Configurations with “MT” use multiple bounding box refinements, in contrast to “ST” where a single transformation is applied at each iteration. Finally, “MD” indicates that multi-domain learning is adopted to pre-train the model (offline learning), “SD” indicates a more classical training procedure where all videos are used to train all model layers.

As shown in Figure 5, models adopting the multiple bounding box refinements achieve higher performance than the corresponding ones with single refinements. However, the training strategy largely affects the performance of the method. Offline multi-domain learning allows achieving higher results. The method that yields to the highest performance uses ADNet parameters and the proposed refinement approach. We note that in terms of precision and success, we achieve higher results than the one published in [6] (0.88 and 0.646, respectively). On the other side, results with single refinements are almost identical, meaning that different implementation choices in the online tracking strategy may have little impact on performance.

Our offline-trained model differs from the ADNet especially because ADNet uses a reinforcement learning approach after the supervised training of the model.

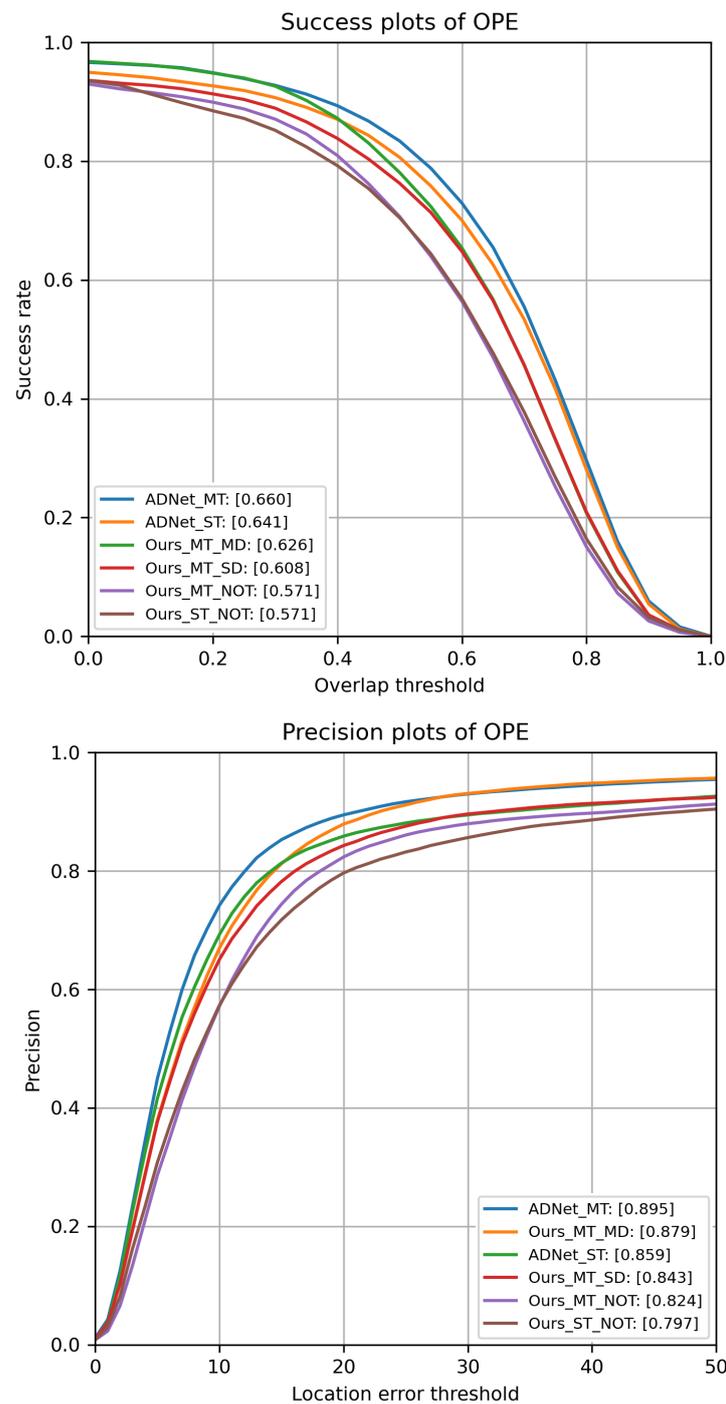


Figure 5. Success and Precision plots on OTB-100 (One-Pass Evaluation (OPE)). Overlap threshold and location error threshold indicate the threshold values used to compute the ROC curves. For the precision plot, the scores in the legend indicate the mean precisions when the location error threshold is 20 pixels. For the success plot, the scores indicate the area under curve (AUC).

5.2. Comparison on OTB and VOT

Table 1 shows the results achieved by our tracker in terms of precision (P), success (AUC), and frame rate (FPS). We used Got10K-toolkit [39] to run all the experiments. Compared to approaches using iterative bounding box refinements (in column Iter), such as TSAS [36] and ADNet, our model (Ours_MT_MD) achieves better/comparable results. Our ADNet-based tracker (ADNet_MT) achieves better/comparable results than those in [7,35,40], which also use iterative refinements.

Table 1. Comparison on OTB-100. Iter indicates approaches using iterative refinements. $P(20px)$ indicates the mean precisions when the location error threshold is 20 pixels. AUC of IoU is the area under the curve of the success plot computed by considering the IoU values.

| Algorithm | P(20px) | AUC of IoU | FPS | Iter |
|------------------------|---------|------------|-----|------|
| Retina-MAML [26] | 0.926 | 0.712 | 40 | |
| VITAL [24] | 0.918 | 0.682 | 2 | |
| SiamRPN++ [41] | 0.915 | 0.696 | 35 | |
| ECO [42] | 0.910 | 0.691 | 8 | |
| MDNet [4] | 0.909 | 0.678 | 1 | |
| RDNet [7] | 0.903 | 0.673 | 4 | x |
| ADNet_MT (ours) | 0.895 | 0.660 | 5 | x |
| Hier. T. [40] | 0.894 | 0.651 | 23 | x |
| IADNET [35] | 0.894 | 0.651 | 3 | x |
| ADNet [6] | 0.88 | 0.646 | 2 | x |
| ATOM [28] | 0.879 | 0.667 | 30 | |
| Ours_MT_MD | 0.879 | 0.626 | 5 | x |
| TSAS [36] | 0.861 | 0.651 | 20 | x |
| ACT [11] | 0.855 | 0.622 | 30 | |
| TD3T [43] | 0.821 | 0.616 | 23 | |
| A3CTD [44] | 0.717 | 0.535 | 50 | |
| GOTURN [22] | 0.565 | 0.425 | 125 | |

Figure 6 shows some samples from three videos belonging to OTB. In the images at the first row, both our method and ADNet are unable to adapt to the actual target shape. This is because rescaling of height and width is done jointly and not separately. However, our method (red bounding-boxes) seems to center better the target. In the second row, both ADNet and our method are sensitive to large and abrupt camera motion. In the third row, where the target is among several instances of the same class (several players), ADNet drifts while our tracker is able to follow the target.

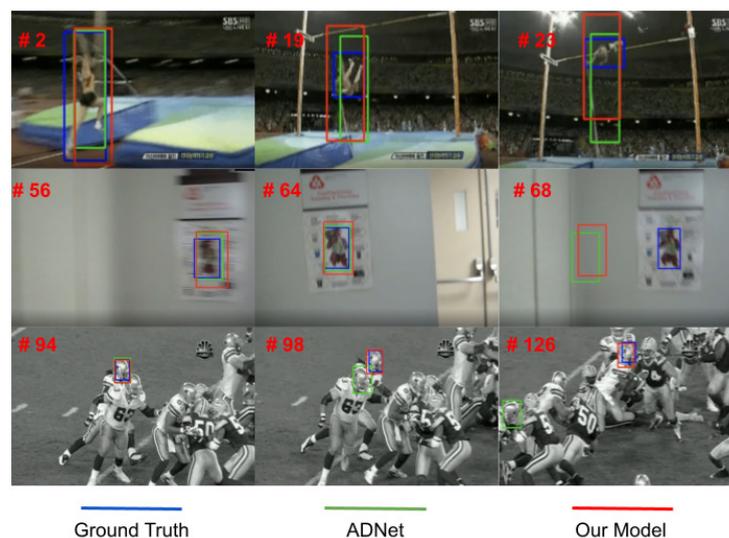


Figure 6. The figure shows some qualitative results of our tracker (red bounding-boxes) vs. ADNet (green bounding-boxes). Ground-truth is shown in blue.

We also compare our tracker on VOT2016 [45], VOT2018 [46], and VOT2019 [47] by adopting Expected Average Overlap (EAO), Accuracy, and Robustness as metrics, and by using the official VOT toolkit. Unfortunately, ADNet was trained on VOT data, and we could not test our modified ADNet model on this benchmarks. Results are reported in Tables 2–4, respectively.

Table 2. Comparison on VOT-2016.

| Algorithm | EAO | Accuracy | Robustness |
|-------------------|-------|----------|------------|
| D3S [31] | 0.493 | 0.66 | 0.131 |
| UpdateNet [48] | 0.481 | 0.61 | 0.21 |
| SiamRPN++ [41] | 0.464 | 0.64 | 0.20 |
| DiMP-50 [49] | 0.440 | 0.597 | 0.153 |
| SPM [50] | 0.434 | 0.62 | 0.21 |
| ATOM [28] | 0.43 | 0.61 | 0.18 |
| ECO [42] | 0.374 | 0.54 | 0.72 |
| Ours_MT_MD | 0.372 | 0.557 | 0.53 |
| RDNet [7] | 0.364 | 0.54 | 0.72 |
| CCOT [51] | 0.331 | 0.54 | 0.238 |

Table 3. Comparison on VOT-2018.

| Algorithm | EAO | Accuracy | Robustness |
|-------------------|-------|----------|------------|
| D3S [31] | 0.489 | 0.64 | 0.15 |
| Ocean-off [52] | 0.467 | 0.598 | 0.169 |
| Retina-MAML [26] | 0.452 | 0.604 | 0.159 |
| DiMP-50 [49] | 0.440 | 0.597 | 0.153 |
| SiamRPN++ [41] | 0.414 | 0.600 | 0.234 |
| ATOM [28] | 0.401 | 0.590 | 0.204 |
| UPDT [53] | 0.378 | 0.536 | 0.184 |
| Ours_MT_MD | 0.372 | 0.56 | 0.44 |
| DRT [54] | 0.356 | 0.519 | 0.201 |

Table 4. Comparison on VOT-2019.

| Algorithm | EAO | Accuracy | Robustness |
|-------------------|-------|----------|------------|
| Retina-MAML [26] | 0.313 | 0.57 | 0.366 |
| ATOM [28] | 0.292 | 0.603 | 0.411 |
| SiamRPN++ [41] | 0.292 | 0.58 | 0.446 |
| SiamMask [29] | 0.287 | 0.594 | 0.461 |
| Ours_MT_MD | 0.232 | 0.513 | 0.72 |

Among all the approaches using iterative bounding box refinements, only RDNet reports results on VOT2016. As shown in Table 2, our method shows improvements over RDNet in all the three metrics.

Overall, the comparison among algorithms adopting iterative bounding box refinements confirms that dealing with conflicting transformations and allowing multiple refinements at each iteration helps improve the tracking results. We also note that the performance of some algorithms such as SiamRPN++, ATOM has decreased from VOT-2016 to VOT-2018. For instance, accuracy of SiamRPN++ is 0.64 and 0.59 on VOT-2016 and VOT-2018, respectively. As for ATOM, accuracy is 0.61 and 0.6 on VOT-2016 and VOT-2018, respectively. The accuracy of the proposed algorithm is 0.557 and 0.56 on VOT-2016 and VOT-2018, respectively. Therefore, despite accuracy slightly decreased also for our method, these results suggest that our method improves in terms of stability.

6. Conclusions and Future Work

This work focused on tracking strategies where the target bounding box is refined iteratively by applying a sequence of transformations. It proposed a novel formulation such that, given an image patch based on the currently estimated target bounding box, the model returns a set of N probability distributions over bounding box transformations.

The method can apply multiple non-conflicting refinements at each iteration without introducing ambiguity during learning, i.e., without giving priority to some transformations over the others.

Experimental results show that the proposed iterative multi-refinement approach is superior to the single-refinement one, independently of the model/training strategies adopted. Overall, the proposed approach is competitive with respect to other state-of-the-art approaches that iteratively refine the target bounding box. In future work, we will try to improve the offline training procedure, as it largely impacts the tracking results. In particular, we plan to adopt a meta-learning approach to accelerate the learning of the model parameters.

Author Contributions: Conceptualization, G.C., L.L.P., and M.L.C.; methodology, G.C. and L.L.P.; software, G.C.; validation, G.C.; formal analysis, G.C., L.L.P. and M.L.C.; investigation, G.C., L.L.P. and M.L.C.; resources, L.L.P. and M.L.C.; data curation, G.C.; writing—original draft preparation, G.C., L.L.P. and M.L.C.; supervision, L.L.P. and M.L.C.; project administration, G.C. and L.L.P.; funding acquisition, L.L.P. and M.L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by MIUR grant number PRIN I-MALL 2017BH297_004 and Italian PON IDEHA grant ARS01_00421.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Marvasti-Zadeh, S.M.; Cheng, L.; Ghanei-Yakhdan, H.; Kasaei, S. Deep learning for visual tracking: A comprehensive survey. *IEEE Trans. Intell. Transp. Syst.* **2021**. [\[CrossRef\]](#)
2. Li, P.; Wang, D.; Wang, L.; Lu, H. Deep visual tracking: Review and experimental comparison. *Pattern Recognit.* **2018**, *76*, 323–338. [\[CrossRef\]](#)
3. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. PAMI* **2013**, *35*, 1798–1828. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Nam, H.; Han, B. Learning multi-domain convolutional neural networks for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4293–4302.
5. Wu, Y.; Lim, J.; Yang, M.H. Online object tracking: A benchmark. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013; pp. 2411–2418.
6. Yun, S.; Choi, J.; Yoo, Y.; Yun, K.; Choi, J.Y. Action-driven visual object tracking with deep reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2239–2252. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Jiang, Y.; Han, D.K.; Ko, H. Relay dueling network for visual tracking with broad field-of-view. *IET Comput. Vis.* **2019**, *13*, 615–622. [\[CrossRef\]](#)
8. Ren, L.; Yuan, X.; Lu, J.; Yang, M.; Zhou, J. Deep reinforcement learning with iterative shift for visual tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 684–700.
9. Huang, C.; Lucey, S.; Ramanan, D. Learning policies for adaptive tracking with deep feature cascades. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 105–114.
10. Zhang, W.; Song, K.; Rong, X.; Li, Y. Coarse-to-fine uav target tracking with deep reinforcement learning. *IEEE Trans. Autom. Sci. Eng.* **2018**, *16*, 1522–1530. [\[CrossRef\]](#)
11. Chen, B.; Wang, D.; Li, P.; Wang, S.; Lu, H. Real-time Actor-Critic Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 318–334.
12. Caicedo, J.C.; Lazebnik, S. Active object localization with deep reinforcement learning. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2488–2496.
13. Jiang, Y.; Shin, H.; Ko, H. Precise regression for bounding box correction for improved tracking based on deep reinforcement learning. In Proceedings of the 2018 IEEE ICASSP, Calgary, AB, Canada, 15–20 April 2018; pp. 1643–1647.
14. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng.* **1960**, *82*, 35–45. [\[CrossRef\]](#)

15. Monteleone, V.; Lo Presti, L.; La Cascia, M. Pedestrian tracking in 360 video by virtual PTZ cameras. In Proceedings of the 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), Palermo, Italy, 10–13 September 2018; pp. 1–6.
16. Lo Presti, L.; Sclaroff, S.; La Cascia, M. Object matching in distributed video surveillance systems by LDA-based appearance descriptors. In *International Conference on Image Analysis and Processing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 547–557.
17. Lo Presti, L.; Morana, M.; La Cascia, M. A data association algorithm for people re-identification in photo sequences. In Proceedings of the 2010 IEEE International Symposium on Multimedia, Taichung, Taiwan, 13–15 December 2010; pp. 318–323.
18. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. PAMI* **2014**, *37*, 583–596. [[CrossRef](#)] [[PubMed](#)]
19. Cruciata, G.; Lo Presti, L.; La Cascia, M. On the Use of Deep Reinforcement Learning for Visual Tracking: A Survey. *IEEE Access* **2021**, *9*, 120880–120900. [[CrossRef](#)]
20. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
21. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
22. Held, D.; Thrun, S.; Savarese, S. Learning to track at 100 fps with deep regression networks. In Proceedings of the ECCV: European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 749–765.
23. Tao, R.; Gavves, E.; Smeulders, A.W. Siamese instance search for tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1420–1429.
24. Song, Y.; Ma, C.; Wu, X.; Gong, L.; Bao, L.; Zuo, W.; Shen, C.; Lau, R.W.; Yang, M.H. Vital: Visual tracking via adversarial learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–24 June 2018; pp. 8990–8999.
25. Song, K.; Zhang, W.; Lu, W.; Zha, Z.J.; Ji, X.; Li, Y. Visual Object Tracking via Guessing and Matching. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 4182–4191. [[CrossRef](#)]
26. Wang, G.; Luo, C.; Sun, X.; Xiong, Z.; Zeng, W. Tracking by instance detection: A meta-learning approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6288–6297.
27. Fan, H.; Ling, H. Sanet: Structure-aware network for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 42–49.
28. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. Atom: Accurate tracking by overlap maximization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4660–4669.
29. Wang, Q.; Zhang, L.; Bertinetto, L.; Hu, W.; Torr, P.H. Fast online object tracking and segmentation: A unifying approach. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1328–1338.
30. Zhang, Z.; Hua, Y.; Song, T.; Xue, Z.; Ma, R.; Robertson, N.; Guan, H. Tracking-assisted weakly supervised online visual object segmentation in unconstrained videos. In Proceedings of the 26th ACM international conference on Multimedia, Seoul, Korea, 22–26 October 2018; pp. 941–949.
31. Lukezic, A.; Matas, J.; Kristan, M. D3S-A discriminative single shot segmentation tracker. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 7133–7142.
32. Yao, R.; Lin, G.; Xia, S.; Zhao, J.; Zhou, Y. Video object segmentation and tracking: A survey. *ACM Trans. Intell. Syst. Technol. (TIST)* **2020**, *11*, 1–47. [[CrossRef](#)]
33. Zhou, T.; Wang, S.; Zhou, Y.; Yao, Y.; Li, J.; Shao, L. Motion-attentive transition for zero-shot video object segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 13066–13073.
34. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
35. Huang, D.; Kong, L.; Zhu, J.; Zheng, L. Improved action-decision network for visual tracking with meta-learning. *IEEE Access* **2019**, *7*, 117206–117218. [[CrossRef](#)]
36. Teng, Z.; Zhang, B.; Fan, J. Three-step action search networks with deep q-learning for real-time object tracking. *Pattern Recognit.* **2020**, *101*, 107188. [[CrossRef](#)]
37. Chatfield, K.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv* **2014**, arXiv:1405.3531.
38. Smeulders, A.W.; Chu, D.M.; Cucchiara, R.; Calderara, S.; Dehghan, A.; Shah, M. Visual tracking: An experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *36*, 1442–1468.
39. Huang, L.; Zhao, X.; Huang, K. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1562–1577. [[CrossRef](#)] [[PubMed](#)]
40. Zhong, B.; Bai, B.; Li, J.; Zhang, Y.; Fu, Y. Hierarchical tracking by reinforcement learning-based searching and coarse-to-fine verifying. *IEEE Trans. Image Process.* **2018**, *28*, 2331–2341. [[CrossRef](#)] [[PubMed](#)]
41. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4282–4291.

42. Danelljan, M.; Bhat, G.; Shahbaz Khan, F.; Felsberg, M. Eco: Efficient convolution operators for tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6638–6646.
43. Zheng, S.; Wang, H. Real-Time Visual Object Tracking Based on Reinforcement Learning with Twin Delayed Deep Deterministic Algorithm. In Proceedings of the International Conference on Intelligent Science and Big Data Engineering, Nanjing, China, 17–20 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 165–177.
44. Dunnhofer, M.; Martinel, N.; Luca Foresti, G.; Micheloni, C. Visual Tracking by means of Deep Reinforcement Learning and an Expert Demonstrator. In Proceedings of the IEEE/CVF ICCVW, Seoul, Korea, 27–28 October 2019.
45. Kristan, M.; Leonardis, A.; Matas, J. The Fourth Visual Object Tracking VOT2016 Challenge Results. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016.
46. Kristan, M.; Leonardis, A.; Matas, J.; Felsberg, M.; Pflugfelder, R.; Cehovin Zajc, L.; Vojir, T.; Bhat, G.; Lukezic, A.; Eldesokey, A.; et al. The Sixth Visual Object Tracking VOT2018 Challenge Results. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
47. Kristan, M.; Matas, J.; Leonardis, A.; Felsberg, M.; Pflugfelder, R.; Kamarainen, J.K.; Cehovin Zajc, L.; Drbohlav, O.; Lukezic, A.; Berg, A.; et al. The Seventh Visual Object Tracking VOT2019 Challenge Results. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 27–28 October 2019.
48. Zhang, L.; Gonzalez-Garcia, A.; Weijer, J.v.d.; Danelljan, M.; Khan, F.S. Learning the model update for siamese trackers. In Proceedings of the IEEE/CVF ICCV, Seoul, Korea, 27 October–2 November 2019; pp. 4010–4019.
49. Bhat, G.; Danelljan, M.; Gool, L.V.; Timofte, R. Learning discriminative model prediction for tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 6182–6191.
50. Wang, G.; Luo, C.; Xiong, Z.; Zeng, W. Spm-tracker: Series-parallel matching for real-time visual object tracking. In Proceedings of the IEEE/CVF Conf. on CVPR, Long Beach, CA, USA, 15–20 June 2019; pp. 3643–3652.
51. Danelljan, M.; Robinson, A.; Khan, F.S.; Felsberg, M. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 472–488.
52. Zhang, Z.; Peng, H.; Fu, J.; Li, B.; Hu, W. Ocean: Object-aware anchor-free tracking. *arXiv* **2020**, arXiv:2006.10721.
53. Bhat, G.; Johnander, J.; Danelljan, M.; Khan, F.S.; Felsberg, M. Unveiling the power of deep tracking. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018; pp. 483–498.
54. Sun, C.; Wang, D.; Lu, H.; Yang, M.H. Correlation tracking via joint discrimination and reliability learning. In Proceedings of the IEEE Conference on CVPR, Salt Lake City, UT, USA, 18–23 June 2018; pp. 489–497.