Alpha-Refine: Boosting Tracking Performance by Precise Bounding Box Estimation

Bin Yan[1]*, Xinyu Zhang[1]*, Dong Wang[1]†, Huchuan Lu[1,2] and Xiaoyun Yang[3]
[1]School of Information and Communication Engineering, Dalian University of Technology, China
[2]Peng Cheng Laboratory [3]Remark AI
{yan_bin,zhangxy71102}@mail.dlut.edu.cn, {wdice, lhchuan}@dlut.edu.cn, xyang@remarkholdings.com

Abstract

Visual object tracking aims to precisely estimate the bounding box for the given target, which is a challenging problem due to factors such as deformation and occlusion. Many recent trackers adopt the multiple-stage strategy to improve bounding box estimation. These methods first coarsely locate the target and then refine the initial prediction in the following stages. However, existing approaches still suffer from limited precision, and the coupling of different stages severely restricts the method's transferability. This work proposes a novel, flexible, and accurate refinement module called Alpha-Refine (AR), which can significantly improve the base trackers' box estimation quality. By exploring a series of design options, we conclude that the key to successful refinement is extracting and maintaining detailed spatial information as much as possible. Following this principle, Alpha-Refine adopts a pixel-wise correlation, a corner prediction head, and an auxiliary mask head as the core components. Comprehensive experiments on TrackingNet, LaSOT, GOT-10K, and VOT2020 benchmarks with multiple base trackers show that our approach significantly improves the base tracker's performance with little extra latency. The proposed Alpha-Refine method leads to a series of strengthened trackers, among which the ARSiamRPN (AR strengthened SiamRPNpp) and the ARDiMP50 (AR strengthened DiMP50) achieve good efficiency-precision trade-off, while the ARDiMPsuper (AR strengthened DiMPsuper) achieves very competitive performance at a real-time speed. Code and pretrained models are available at https://github.com/MasterBin-IIAU/AlphaRefine.

1. Introduction

Precise box estimation is indispensable for a successful tracker. Early trackers usually solve this problem by multi-scale search [1, 6, 35, 3] or sampling-then-regression strategy [37, 29], which are inaccurate and greatly limit the performance of the trackers. For obtaining more robust and precise tracking results, many state-of-the-art trackers [40, 12, 7, 2] adopt a multiple-stage tracking strategy, which introduces additional tracking stages for more precise box estimation. These trackers first coarsely locate the target and then refine the initial result in the additional tracking stages to get more precise box prediction. However, this box estimation can still be improved, even for state-of-the-art trackers. An oracle experiment verifies this opinion. We set the tracker's search region always centering at the ground truth so that the performance will be mainly determined by the capacity of box estimation. Table 1 shows that with the aforementioned oracle setting, state-of-the-art trackers' AUC scores are still far from perfection, indicating unsatisfying box estimation, although some methods (*e.g.* ATOM [7], DiMP [2]) have built-in box estimation modules. In contrast, given the perfectly centered search region, the proposed Alpha-Refine achieves significantly better performance, demonstrating that Alpha-Refine's superiority in box estimation.

Table 1. Oracle experiment on LaSOT. The center of the search region is always set at the center of the ground truth, reflecting the box estimation capacity of these methods. The best three results are marked in **red**, **green** and **blue** bold fonts respectively.

Oracle	AUC	P_{Norm}	P
SiamRPNpp[23]	**0.682**	**0.829**	**0.745**
ATOM[7]	0.580	0.686	0.604
DiMPsuper[2]	**0.693**	**0.799**	**0.734**
ECO[6]	0.496	0.666	0.533
AlphaRefine	**0.762**	**0.902**	**0.919**

Additionally, most of refinement methods in existing trackers [40, 12, 7, 2] are weak in transferability, because their training is coupled with other components. Extra retraining is required if above refinement modules are applied to new base trackers. As opposed to these methods, Alpha-

*Equal contribution.
†Corresponding Author: Dr. Dong Wang, wdice@dlut.edu.cn

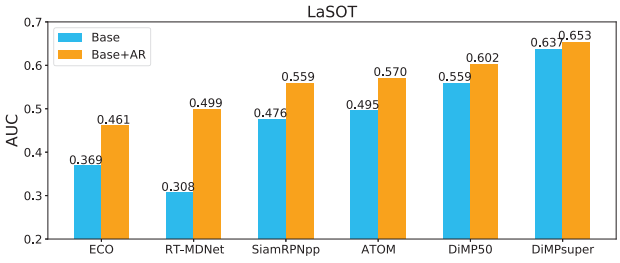

Figure 1. Performance improvement of our Alpha-Refine module on LaSOT. 'Base': the base tracker; 'Base+AR': the base tracker with Alpha-Refine (AR). This figure shows that all base trackers are significantly improved by the proposed AR module.

Refine is trained independently and can be directly applied to any existing trackers in a plug-and-play style, requiring no extra training or modification of the base tracker.

In this work, a series of design options are investigated and compared. Specifically, we assess multiple feature fusion modules and prediction heads. We also explore to use an auxiliary mask head, which introduces pixel-level supervision into the training. We find that extracting and maintaining precise spatial information is the key to precise box estimation. To this end, we finally adopt a pixel-wise correlation as well as a key-point style prediction head for better maintaining and utilizing the detailed spatial information. Additionally, an auxiliary mask head is used, which encourages the network to extract more detailed spatial information, leading to more precise box estimation. Moreover, if we reserve the mask head at the inference stage, Alpha-Refine will enable the base trackers to predict the mask of the object, satisfying scenarios where the mask is required. The design options will be discussed in Section 3 and verified in Section 4.

To verify the effectiveness of our Alpha-Refine module, we choose six famous base trackers: ECO [6], RT-MDNet [16], SiamRPNpp [23], ATOM [7], DiMP [2], and DiMPsuper [2], on multiple tracking benchmarks, namely, LaSOT [11], GOT-10K [14], TrackingNet [28] and VOT2020 [19]. Take Fig. 1 as an example, experimental results show that our proposed refinement module improves the base trackers' performance significantly. Compared with its competitors (*i.e.* IoU-Net [7, 2] and SiamMask [42]), Alpha-Refine's performance also surpasses them by a large margin.

The proposed Alpha-Refine method leads to a series of strengthened trackers, among which the AR-SiamRPN (Alpha-Refine strengthened SiamRPNpp) and ARDiMP50 (Alpha-Refine strengthened DiMP50) achieve good efficiency-precision trade-off, while ARDiMPsuper (Alpha-Refine strengthened DiMPsuper) achieves state-of-the-art performance at a real-time speed on multiple benchmarks.

2. Related Works

Early Box Estimation. Early box estimation methods are mainly scale estimation, which can be summarized into two categories: multiple-scale search and sampling-then-regression strategies. Most correlation-filter-based trackers [13, 6, 35] and SiamFC [1] adopt the former strategy. Specifically, these trackers construct search regions with different sizes, then compute correlation with the template, and finally determine the size of the target as the size-level where the highest response locates. Multiple-scale search is coarse and time-consuming due to its fixed-aspect-ratio prediction and heavy image pyramid operation. Another type of method first generates several bounding box samples, then uses some methods to choose the best one, and finally apply regression on it to obtain more accurate results. SINT [37], MDNet [29] and RT-MDNet [16] are three representative trackers that exploit this approach.

Modern Box Estimation. As deep learning techniques become mature, several high-performance scale estimation approaches are developed and can be categorized into the following classes: RPN-based [24, 51, 23], Mask-based[42, 26], IoU-based [7, 2], and Anchor-free-based [45, 4]. RPN-based methods learn a region proposal network [31], which determines whether the current anchor contains the target and makes refinement to the target simultaneously. SiameseRPN-series trackers [24, 51, 23] utilize the RPN-based mechanism as the core component and achieve great success in recent years. Mask representation is more accurate, and the ability to predict mask is quite beneficial to precise box estimation. SiamMask [42] and D3S [26] belong to this class, which obtain higher precision than the Siamese tracker that can only predict boxes. IoU-based approaches learn a network to predict the overlap between candidate boxes and groundtruth. During the inference phase, this strategy optimizes candidate boxes by gradient-ascent, and therefore obtains more precise results. ATOM [7] and DiMP [2] fully exploit this method and surpass traditional correlation-filter trackers by a large margin. In the past years, anchor-free philosophy has become quite popular in the object detection field [22, 49, 18, 38]. SiamFC++ [45] introduces this structure into object tracking field and therefore achieves state-of-the-art performance. The CGACD method [9] designs a corner-based box estimation for object tracking which wisely adopts soft-argmax to decode the corner heat map into box coordinates. The corner-based version of Alpha-Refine adopts a similar box representation, and experiments demonstrate that this design retains more precise spatial information for a refinement module.

Refinement Modules. Many state-of-the-art trackers [40, 12, 7, 2, 21] apply a multiple-stage tracking strategy to obtain accurate and robust results. This approach first locates the target coarsely and then utilizes a refinement module to refine results from the previous stage. SPM [40]

and Siamese Cascaded RPN [12] adopt a light-weight relation network [36] and stacked RPNs [31], respectively, as the refinement module to further increase trackers' discriminative power and precision. However, the two refinement modules have to be trained together with their previous Siamese tracker in an end-to-end manner; this procedure limits their flexibility of combining with other base trackers. ATOM [7] and DiMP [2] first use an online classification module to locate the target and then draw some random samples around it. Finally, they deploy a modified IoU-Net [15] to maximize the overlap between these samples and groundtruth to obtain more precise bounding boxes. This modified IoU-Net can be trained separately from the base tracker. Thus, the IoU-Net has good transferability but its precision can still be greatly improved. Notably, the winners of VOT2019 [21] utilize SiamMask [42] as a refinement module [21]. Similar to IoU-Net [15] mentioned before, SiamMask [42] can be combined with any base tracker. However, SiamMask is designed as an independent tracker rather than a refinement module, which is not suitable and not economical to refine other trackers. Considering previous refinement modules' weak transferability and limited accuracy, we propose a novel, flexible, and accurate refinement module named Alpha-Refine.

3. Alpha-Refine

Alpha-Refine is a refinement module which is able to efficiently refine the base tracker's outputs and significantly improve the tracking performance. We detail the network architecture (Fig. 2), design options, and training process as follows.

3.1. Network Architecture

Fig. 2 shows the overall architecture of the proposed Alpha-Refine module. This module adopts the Siamese architecture with two input branches, namely, the *reference branch* and the *test branch*. A parameter-shared backbone is applied to both branches for feature extraction. Features extracted from two branches are aggregated by a fusion module, which is typically a correlation module (*e.g.* naive-correlation, depth-wise correlation, pixel-wise correlation).

The fused feature is further processed by some convolutional layers, producing the features for the prediction head. An auxiliary mask head can be added parallel to the box head to introduce pixel-level supervision into training. The output of the mask head can be used for scenarios that also require a mask result.

To function as a refinement module, the reference branch is initialized by the first frame with the ground truth. In the current frame, the test branch extends the base tracker's prediction into a concentric search region of two times the size, from which Alpha-Refine predicts a finer result. Alpha-Refine can be combined with arbitrary trackers in a plug-and-play style and improve their performance.

Notably, compared with independent trackers, the size of Alpha-Refine's search region is roughly two times the size of the object, which is smaller than normal trackers (four times in most cases). The smaller search region can depress the cluttered background and enable the model to focus on more detailed spatial information, which is beneficial to precise localization. Small search region also lowers the computation cost, so that Alpha-Refine can improve the base tracker with little latency increase. Alpha-Refine is not capable of tracking by itself because of the small search region light-weight design. A complete base tracker is always needed.

3.2. Feature Fusion

Most methods with Siamese architecture aggregate features of the template and the search region using the coarse naive correlation [1, 24, 51] or depth-wise correlation [23, 42]. As shown in Fig. 3, both naive correlation or depth-wise correlation take the whole template feature as the kernel to correlate with the search region feature, making adjacent sliding window on the feature map producing similar response and blur the spatial information. As a refinement module, Alpha-Refine requires the feature fusion module to maintain as much spatial information as possible. Thus, the popular naive nor depth-wise correlation is not suitable for Alpha-Refine.

In this work, we adopt pixel-wise correlation [43] for high-quality feature representation. We denote $K \in \mathbb{R}^{C\times H_0\times W_0}$ and $S \in \mathbb{R}^{C\times H\times W}$ as features of the template and the search region. Pixel-wise correlation first decomposes K into H_0W_0 small kernels $K_j \in \mathbb{R}^{C\times 1\times 1}$ and then uses them to compute correlation separately to obtain correlation maps $C \in \mathbb{R}^{H_0W_0\times H\times W}$. The process can be described as

$$C = \{C_j | C_j = K_j * S\}_{j\in\{1,\dots,H_0\times W_0\}}, \qquad (1)$$

where $*$ denotes naive correlation.

In contrast to naive or depth-wise correlation, pixel-wise correlation takes each part of the target features as a kernel. Pixel-wise correlation ensures that each correlation map encodes information of a local region on the target while avoiding an extremely large correlation window from blurring the feature.

Fig. 5 shows the computation process of three correlation methods, and Fig. 4 shows some fusion outputs. Fig. 4(c) indicates that naive convolution can only roughly represent the center location of the object while losing most of the shape and scale information. Fig. 4(d) illustrates that depth-wise correlation has to encode the blurred location into channels, which is less explainable and inefficient. By contrast, Fig. 4(e) shows that pixel-wise correlation is better

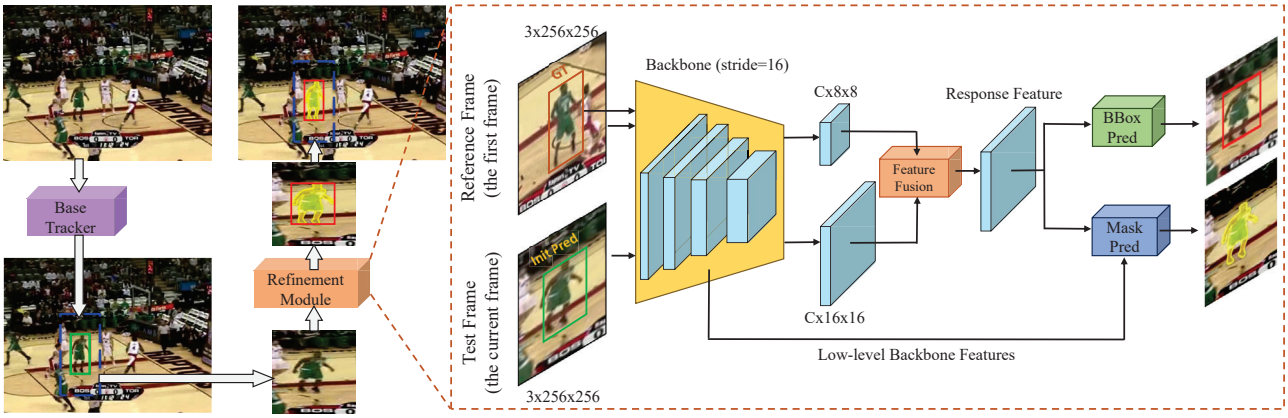

Figure 2. Overall architecture of the proposed Alpha-Refine. Better viewed in color with zoom-in.

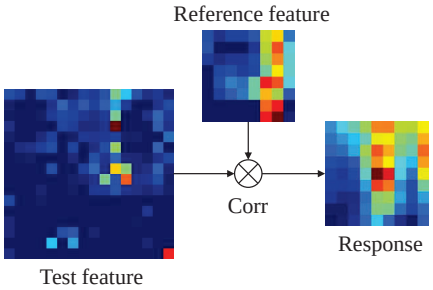

Figure 3. Illustration of blur effect. Depth-wise correlation or naive correlation may blur the spatial information.

at retaining the target's boundary and other detailed spatial information.

3.3. Prediction Heads

We explore two ways of predicting the bounding box: directly regressing the box coordinates and predicting two corner points[1] from two heatmaps. For regressing the box coordinates, we evaluate RPN style and RCNN style designs. For predicting the corners, we evaluate the key-point style design.

RPN Style Box Head. One way of regressing the bounding box is the RPN Style Box Head. Fig. 6(a) shows the diagram. Similar to the FCN structure of one-stage object detectors, a 4D box coordinates together with a confidence score is predicted at each location. The box with the highest score is regarded as the tracking result.

However, we notice some drawbacks of using this method in the refinement module. In this method, each box prediction is generated by an individual feature point, which requires this feature point to summarize the information in its receptive field and encode spatial information into the channels, so that a single feature point can make a prediction by itself. However, different feature points have varying receptive fields, making them good at representing various parts of the object. The RPN style method ignores the relationship between feature points at different locations, not fully utilizing the information contained in the spatial distribution of the feature map. Additionally, this strategy has inconsistency because most precise box predictions may have a low score. In the experiment, we implement this strategy by stacking four Conv-BN-ReLU layers followed by a prediction layer. For simplicity, we directly regress four distances from the feature point location to four edges of the bounding box. Another four Conv-BN-ReLU layers are used to predict the confidence score.

RCNN Style Box Head. Similar to the second stage of Faster-RCNN [31], this RCNN Style method reduces the feature map into a vector and estimates the bounding box of the object with some fully connected layers. Fig. 6(b) shows the diagram. Compared with the RPN style method, this method utilize the whole feature map rather than individual feature points. Apparently, this method crashes spatial information when reducing the feature map, indicating that it is not suitable for a refinement module. For this strategy, we use a bounding box head containing four stacked Conv-BN-ReLU layers in our experiment, followed by a global average pooling layer and a fully-connected layer, which predicts four coordinates of the bounding box.

Corner Head. Recently, keypoints detection techniques have become popular in the object detection field, producing several state-of-the-art methods [22, 49, 10, 50]. In our experiment, we implement a corner head with four stacked Conv-BN-ReLU layers, followed by a Conv layer predicting two heatmaps, which represent top-left corner and bottom-right corner respectively. Different from methods like CornerNet [22], we do not upsample the feature map for the computation issue, resulting in a coarse-grained heatmaps. We apply soft-argmax [27] to the heatmaps to make the discrete heatmaps precisely describe the position

[1]The top-left corner and the bottom-right corner

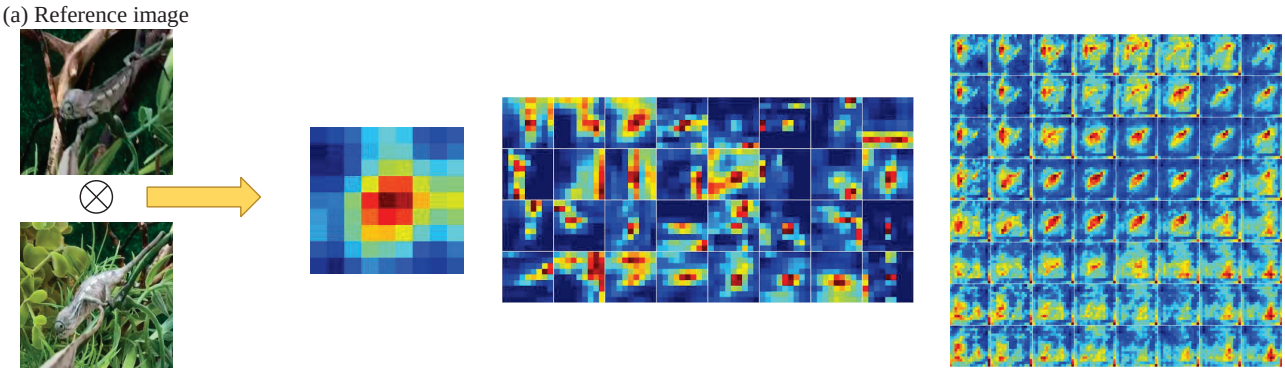

Figure 4. Comparison among different correlation responses. (a) and (b) denote the reference branch and the test branch, respectively. (c), (d), and (e) are correlation result of naive, depth-wise (the first 32 channels of 256), and pixel-wise correlations, respectively.

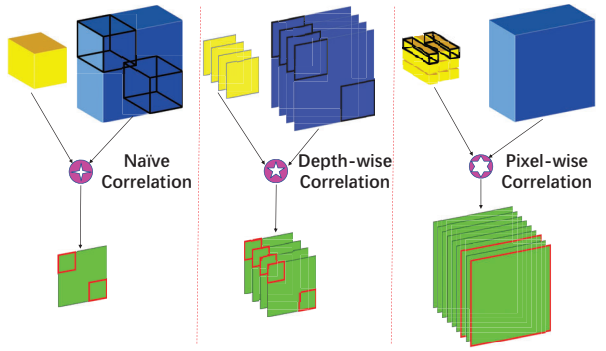

Figure 5. Comparison among different correlation methods and the illustration of the non-local module. From left to right, naive, depth-wise, and pixel-wise correlations are demonstrated. The black-edged cubes or squares represent sliding kernels. The red edged ones represent corresponding correlation maps.

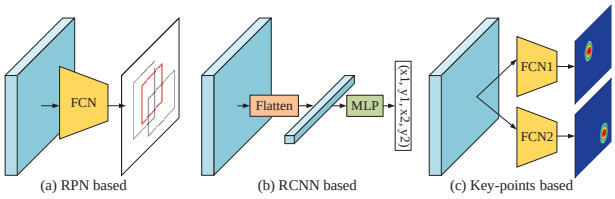

Figure 6. Options for predicting box results. (a) RPN style box prediction. (b) RCNN style box prediction. (c) Corner prediction.

of the corner point. The soft-argmax operation enable our model to predict continuous values from discrete heatmaps. It encodes the box estimation into the distribution of confidence (heat) scores, avoiding the inconsistency problem in the RPN style head. The key-point style method retrains the natural spatial structure of the feature map, avoiding encoding spatial information into channels, which is desirable for Alpha-Refine.

Auxiliary Mask Head. As Alpha-Refine is a module for precisely estimating the bounding box, additional detailed shape information would be helpful. To this end, we add an auxiliary mask head parallel to the box head, which introduces pixel-level supervision into training. When the box head is trained with the auxiliary mask head, the network is encouraged to extract more detailed spatial information which is required by the mask head and facilitates precise box estimation. In addition, supervision from mask annotation also teach the model to better discriminate foreground and background, which is required by the segmentation task and also beneficial to tracking. Some previous works [42, 26] also demonstrate that mask prediction is quite beneficial for improving tracking performance, especially on benchmarks (*e.g.*, VOT [20, 21]) that adopt rotated bounding box labels. In this work, the mask head is implemented as a U-Net [32] style decoder, which gradually upsample the feature map while fusing them with low-level features from the backbone until the resolution is the same as the input image, and a mask is predicted from the last layer. At the inference stage, the mask head is by default disabled to speed up Alpha-Refine. For scenarios requiring pixel-level prediction, the mask head can be activated, producing mask prediction as the output.

3.4. Training

Training Set Construction. We use the training splits of LaSOT [11] and GOT-10K [14], ImageNet VID [33], ImageNet DET [33], COCO [25], Youtube-VOS [44], and some segmentation datasets [46, 41, 34] to train the Alpha-Refine. Given a video sequence, two stochastic frames F_{ref} and F_{test} with an interval of less than 50 frames are first selected. The input of the reference branch is obtained by cropping F_{ref} at the center of the ground truth with two times the size of the ground truth box. The input of the test branch is obtained by cropping F_{test} randomly centered around the ground truth, with a jittered size. Specifically, we randomly translate and scale the ground truth of F_{test} to

obtain the region to be cropped. With the following equations:

$$[h, w] = [2h^{GT}, 2w^{GT}] \times e^{\mathbf{N} f_s^{test}} \tag{2}$$

$$O_{\max} = \sqrt{hw} \times f_c^{test} \tag{3}$$

$$[c_x, c_y] = [c_x^{GT}, c_y^{GT}] + (\mathbf{U} - 0.5) \times O_{\max} \tag{4}$$

we obtain the region centering at $[c_x, c_y]$ with size $[h, w]$. $[c_x^{GT}, c_y^{GT}, h^{GT}, w^{GT}]$ is the ground truth bounding box. f_s^{test} and f_c^{test} are two scalar factors corresponding to scale and center, respectively. We use $[f_s^{test}, f_c^{test}] = [0.25, 0.25]$ in our experiments. $\mathbf{N}$ and $\mathbf{U}$ represent the $2D$ standard normal distributed random variable and $2D$ uniform random variable respectively. The cropped images are resize into 256×256 as the inputs of Alpha-Refine.

Training Approach. For the box output (*i.e.* output of RPN style, RCNN syle, Key-Point style Heads), the mean squared error L_{box} is used. All predictions are converted into coordinate vectors of the format [left-most, top-most, right-most, bottom-most] and compared with ground truth to obtain the mean squared error. For the mask output, a binary cross-entropy loss L_{mask} is used. The total loss L is the weighted sum of two losses.

$$L = L_{box} + \lambda L_{mask}, \tag{5}$$

where $\lambda = 1000$ is used in the experiments. We train Alpha-Refine for 40 epochs, each of which consists of 500 iterations on eight Nvidia 2080Ti GPU with a batch size of 32 per GPU (32×8 samples per iteration in total). Considering the abundance of the training data, we do not freeze any parameters of the backbone. The Adam optimizer [17] is applied and the learning rate halves every 8 epochs.

4. Experiments

We implement our algorithm with the Pytorch [30] deep learning library. In this section, we verify the effectiveness of Alpha-Refine by performing comprehensive experiments on many popular tracking benchmarks: LaSOT [11], TrackingNet [28], GOT-10K [14], and VOT2020 [19] together with six representative and state-of-the-art base trackers (including ECO [6], RT-MDNet [16], ATOM [7], SiamRPNpp [23], DiMP50 [2], and DiMPsuper [2]) to demonstrate our Alpha-Refine's capacity of boosting the trackers' performance. Besides, we evaluate our design options with SiamRPNpp [23] as the base tracker and determine the effects of different settings of our Alpha-Refine. ResNet-34 is used as backbone by default if not otherwise specified. All experiments of our trackers run five times, and the results are obtained by average.

Table 2. Comparison results on the *LaSOT test* set. 'Base': the base tracker; and 'Base+AR': the base tracker with Alpha-Refine. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. Numbers are shown in percentage (%).

Method	Base			Base+AR		
	AUC	P_{Norm}	P	AUC	P_{Norm}	P
ECO	36.9	43.5	36.4	46.1	50.8	46.0
RT-MDNet	30.8	36.0	30.1	49.9	63.1	50.7
SiamRPNpp	47.6	54.7	47.2	55.9	62.2	57.4
ATOM	**49.5**	**56.0**	**49.1**	**57.0**	**63.0**	**58.1**
DiMP50	**55.9**	**63.3**	**55.3**	**60.2**	**66.8**	**61.7**
DiMPsuper	**63.7**	**72.5**	**65.6**	**65.3**	**73.2**	**68.0**

4.1. Representative Visual Results

Figure 7 provides some representative visual results regarding different refinement module. We can see that our Alpha-Refine module facilitates the tracker obtaining more precise bounding boxes than IoU-Net and SiamMask.

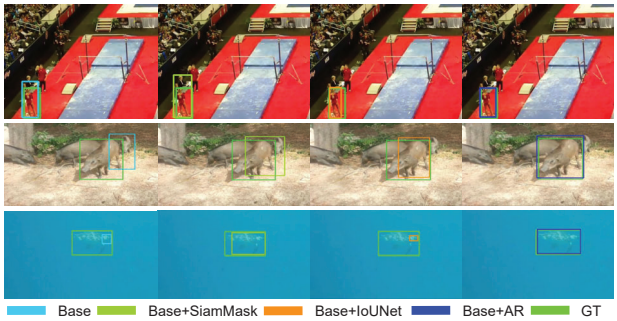

Figure 7. Visual Comparison of Alpha-Refine and other refinement modules. From left to right, we present the origin prediction of the SiamRPNpp base tracker, and refined results obtained by SiamMask [42], IoU-Net [7, 2], our Alpha-Refine.

4.2. Evaluation on LaSOT

LaSOT [11] is a recent large-scale tracking benchmark, which consists of 1400 challenging videos (1120 for training and 280 for testing). In this work, we follow the one-pass evaluation, using Success (AUC), Normalized Precision (P_{Norm}), and Precision (P), to compare different trackers without and with Alpha-Refine. Table 2 shows that our Alpha-Refine (AR) module consistently and significantly improves the base trackers in all evaluation metrics. Especially for RT-MDNet, the improvement of the AUC score is up to 19%. As shown in Table 3 the previous best tracker is Siam R-CNN [39], which obtains a 64.8% AUC score but merely runs around 5 *fps*. in contrast, ARDiMPsuper achieves the best record (AUC: 65.3%), while maintaining a real-time speed.

Latency and Speed. Table 4 reports the latency and speed performance of different trackers without and with Alpha-Refine, showing that our Alpha-Refine module introduces few computation loads (merely about 5-6ms every frame), while significantly improving the tracking accuracies (see Table 2).

Table 3. Comparison state-of-the-art results on the *LaSOT test* set. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. Numbers are shown in percentage (%). More results are available at https://github.com/MasterBin-IIAU/AlphaRefine.

Method	**ARDiMPsuper** (ours)	SiamRCNN [39]	**ARDiMP50** (ours)	PrDiMP [8]	LTMU [5]	DiMP50 [2]	Ocean [48]	**ARSiamRPN** (ours)	SiamAttn [47]	SiamFC++ [45]
AUC(%)	**65.3**	**64.8**	**60.2**	59.8	57.2	56.8	56.0	56.0	56.0	54.4
Speed(fps)	33	5	46	30	13	59	25	50	45	90

Table 4. Latency and speed of different methods. The tracking speed is measured using frame per second (fps).

Method	**Base**		**Base+AR**		Δt
	latency	fps	latency	fps	
ECO	13.3ms	75.2	18.9ms	52.9	+5.6ms
RTMDNet	14.3ms	69.9	20.1ms	49.8	+5.7ms
ATOM	16.8ms	59.5	22.1ms	45.2	+5.3ms
SiamRPNpp	14.9ms	67.1	20.0ms	50.0	+5.1ms
DiMP50	16.7ms	59.9	21.9ms	45.7	+5.2ms
DiMPsuper	25.2ms	39.7	30.4ms	32.9	+5.2ms

4.3. Ablation Studies

In this subsection, we conduct ablation studies of our Alpha-Refine (AR) module using SiamRPNpp [23] as the base tracker, evaluated on the LaSOT [11] test set.

Table 5. Analysis of different head options. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. Numbers are shown in percentage (%).

Method	AUC(%)	$\mathrm{P_{Norm}}$(%)	P(%)
SiamRPNpp	47.6	54.7	47.2
+AR_{rpn}	50.2	55.5	51.2
+AR_{rcnn}	48.9	54.2	46.9
+AR_{c}	**54.6**	**60.3**	**55.3**
+AR_{rpn+m}	**53.7**	**60.3**	**54.7**
+AR_{rcnn+m}	51.6	58.1	52.3
+AR_{c+m}	**55.9**	**62.2**	**57.4**

Head Options. The head option is a very important component in this work, since it is directly related with the final output. Table 5 reports the performance of the SiamRPNpp+AR tracker with different head options. The symbols $\mathrm{AR_{rpn}}$, $\mathrm{AR_{rcnn}}$ and $\mathrm{AR_{c}}$ denote the Alpha-Refine module with RPN style box head, RCNN style box head and KeyPoint style corner head, respectively. '+m' stands for the auxiliary mask head used during training. From Table 5, we have the following two conclusions: (1) all adopted box estimation heads improve the original SiamRPNpp method, and the corner head performs much better than the other two; and (2) the auxiliary mask head further makes additional improvements, and the combination of the corner and mask heads obtains the best performance. Thus, we chose $\mathrm{AR_{c+m}}$ as our final module in this work.

Feature Fusion Options. Table 6 compares the SiamRPNpp+AR variants using different feature fusion options (naive, depthwise, or pixelwise in Sec 3.2), where the prediction head is determined based on aforementioned discussions. The results show that the adopted pixelwise correlation performs the best, indicating that the pixelwise correlation is better at extracting and maintaining spatial information than the depthwise correlation or naive correlation.

Table 6. Analysis of different feature fusion types. Naive indicates the typical feature correlation between reference and test branches. Numbers are shown in percentage (%).

Method	AUC(%)	$\mathrm{P_{Norm}}$(%)	P(%)
Pixelwise	**55.9**	**62.2**	**57.4**
Depthwise	54.8	60.8	55.8
Naive	53.1	59.4	53.9

Table 7. Comparison of different refinement modules. The best result is marked in **red** bold fonts.

Method	AUC(%)	$\mathrm{P_{Norm}}$(%)	P(%)
SiamRPNpp	47.6	54.7	47.2
+IoU-Net	48.8	55.6	47.8
+SiamMask	50.3	54.7	48.7
+AR	**55.9**	**62.2**	**57.4**

Comparison with Different Refinement Modules. We compare our Alpha-Refine (AR) with two recent refinement modules (IoU-Net presented in [7, 2] and SiamMask proposed in [42]), and report the results in Table 7. Our Alpha-Refine module surpasses IoU-Net and SiamMask by a large margin.

Different Backbones. We investigate the Alpha-Refine module with different backbones and reports the comparison results in Table 8. When the ResNet-18 backbone is used, the latency of our AR model is very low but the corresponding performance is also 7.4% higher than the original SiamPRNpp. As the backbone goes deeper, the AUC score is better but the speed is slower. In this work, we choose ResNet-34 as the default backbone to balance accuracy and speed.

Table 8. Accuracy and Speed Comparison of SiamRPNpp+AR with different backbones.

Method	AUC(%)	fps	latency	Δt
SiamRPNpp	47.6	67.1	14.9ms	
+ AR(ResNet-50)	56.2	46.5	21.5ms	6.6ms
+ AR(ResNet-34)	55.9	50.0	20.0ms	5.1ms
+ AR(ResNet-18)	55.0	52.4	19.1ms	4.2ms

4.4. Evaluation on Other Benchmarks

TrackingNet. TrackingNet [28] is a popular large-scale

Table 9. Comparison results on the *TrackingNet test* set. 'Base': the base tracker; and 'Base+AR': the base tracker with Alpha-Refine. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. Numbers are shown in percentage (%).

Method	Base			Base+AR		
	AUC	P_{Norm}	P	AUC	P_{Norm}	P
ECO	61.2	71.0	55.9	75.1	80.0	71.4
RT-MDNet	58.4	69.4	53.3	76.0	81.0	72.3
ATOM	70.3	77.1	64.8	77.7	82.5	74.5
SiamRPNpp	**73.3**	**80.0**	**69.4**	**78.8**	**83.7**	**76.4**
DiMP50	**74.0**	**80.1**	**68.7**	**79.5**	**84.1**	**76.5**
DiMPsuper	**77.6**	**82.5**	**72.6**	**80.5**	**85.6**	**78.3**

short-term tracking benchmark. We evaluate various methods on its *test* set, which contains 511 sequences. For the *test* set, only groundtruth of the first frame is given and participants need to submit their results to the evaluation server. Table 9 shows that our Alpha-Refine module improves different base trackers by a large margin. ARDiMPsuper obtains 80.5% in the main AUC metric, which is slightly worse than the previous best tracker (Siam R-CNN [39]: 81.2% in AUC). However, ARDiMPsuper runs approximately 32.9 *fps*, being six times faster than Siam R-CNN.

GOT-10K. GOT-10K [14] is a recent large-scale dataset, which contains 10K sequences for training and 180 for testing. We submit the tracking outputs to the official evaluation server and obtain the comparison results (*i.e.*, AO and SR_T) in Table 10. On one hand, our Alpha-Refine module consistently and significantly improves the base trackers in all evaluation metrics. On the other hand, ARDiMPsuper achieves 70.1% in the main AO metric, which performs much better and runs much faster than the previous best tracker (Siam R-CNN [39]: 64.9% in AO, 72.8% in SR_T, and 59.7% in $SR_{0.75}$).

Table 10. Comparison results on the *GOT-10K test* set. 'Base': the base tracker; and 'Base+AR': the base tracker with Alpha-Refine. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. Numbers are shown in percentage (%).

Method	Base			Base+AR		
	AO	$SR_{0.5}$	$SR_{0.75}$	AO	$SR_{0.5}$	$SR_{0.75}$
ECO	41.3	43.8	13.4	56.7	64.8	46.1
RT-MDNet	35.0	35.8	9.2	56.1	63.7	46.9
ATOM	**53.5**	**62.2**	**37.8**	**63.1**	**71.1**	**55.8**
SiamRPNpp	51.8	61.7	32.4	61.5	69.6	46.9
DiMP50	**60.3**	**71.8**	**46.0**	**65.4**	**74.3**	**58.5**
DiMPsuper	**67.2**	**78.8**	**59.3**	**70.1**	**80.0**	**64.2**

VOT2020. VOT2020 [19] includes 60 challenging videos with high-quality mask-based ground truth. This benchmark takes expected average overlap (EAO) as the main ranking metric, which simultaneously considers the trackers' accuracy and robustness. The evaluation on VOT2020 has two settings: Baseline and real-time. The real-time requires the trackers to predict bounding boxes no slower than the video frame rate (20 fps in the official toolkit). The results of different trackers are shown in Table 11. We can see that the proposed Alpha-Refine module improves the base trackers in terms of EAO significantly. Besides, Figure 8 demonstrates that our AR strengthened method ARDiMPsuper obtains the best performance in the Real-Time setting.

Table 11. Comparison results on the VOT2020 benchmark. 'Base': the base tracker; and 'Base+AR': the base tracker with Alpha-Refine. The best three results are marked in **red**, **green** and **blue** bold fonts, respectively. The main EAO metric is reported.

Method	Base		Base+AR	
	Baseline	Real Time	Baseline	Real Time
RT-MDNet	0.248	0.247	0.371	0.356
SiamRPNpp	0.254	0.254	0.395	0.395
ECO	**0.280**	**0.276**	**0.426**	**0.426**
ATOM	0.275	0.279	0.416	0.414
DiMP50	**0.286**	**0.278**	**0.444**	**0.438**
DiMPsuper	**0.314**	**0.311**	**0.471**	**0.478**

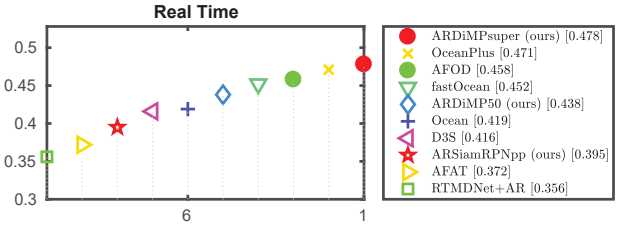

Figure 8. State-of-the-art evaluation on VOT2020. Our ARDiMPsuper obtains the best result in the real-time setting.

5. Conclusion.

In this work, we propose a novel Alpha-Refine method for visual tracking, which is an accurate and general refinement module to effectively improve the tracking performance of different types of trackers in a plug-and-play style. By exploring multiple design options, we find that extracting and maintaining precise spatial information is the key to the precise box estimation. Alpha-Refine finally adopts a precise pixel-wise correlation layer, a Key-Point style prediction head, and an auxiliary mask head. Finally, we apply the Alpha-Refine model to six well-known and top-performed trackers and conduct numerous evaluations on four popular benchmarks. The experimental results demonstrate that our Alpha-Refine could consistently improve the tracking performance with few computational loads.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under Grant nos. 62022021, 61806037, 61872056, and 61725202, and in part by the Science and Technology Innovation Foundation of Dalian under Grant no. 2020JJ26GX036.

References

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, 2016. 1, 2, 3

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 1, 2, 3, 6, 7

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 1

[4] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 2

[5] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *CVPR*, 2020. 7

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017. 1, 2, 6

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *CVPR*, 2019. 1, 2, 3, 6, 7

[8] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, 2020. 7

[9] Fei Du, Peng Liu, Wei Zhao, and Xianglong Tang. Correlation-guided attention for corner detection based visual tracking. In *CVPR*, 2020. 2

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In *ICCV*, 2019. 4

[11] 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 *CVPR*, 2019. 2, 5, 6, 7

[12] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019. 1, 2, 3

[13] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. In *ICVS*, 2008. 2

[14] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2019. 2, 5, 6, 8

[15] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018. 3

[16] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time MDNet. In *ECCV*, 2018. 2, 6

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[18] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. FoveaBox: Beyound anchor-based object detection. *TIP*, 2020. 2

[19] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Martin Danelljan, Alan Lukezic, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernandez, and et al. The eighth visual object tracking vot2020 challenge results. In *ECCVW*, 2020. 2, 6, 8

[20] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *ECCVW*, 2018. 5

[21] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, et al. The seventh visual object tracking vot2019 challenge results. In *ICCVW*, 2019. 2, 3, 5

[22] Hei Law and Jia Deng. CornerNet: detecting objects as paired keypoints. In *ECCV*, 2018. 2, 4

[23] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 1, 2, 3, 6, 7

[24] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 2, 3

[25] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[26] Alan Lukezic, Jiri Matas, and Matej Kristan. D3S - A discriminative single shot segmentation tracker. In *CVPR*, 2020. 2, 5

[27] Diogo C. Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. *Computers & Graphics*, 2019. 4

[28] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 2, 6, 7

[29] Hyeonseob Nam and Bohyung Han. Learning multi–domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 2

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R–CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 3, 4

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large scale visual recognition challenge. *IJCV*, 2015. 5

[34] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *TPAMI*, 2015. 5

[35] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018. 1, 2

[36] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 3

[37] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 1, 2

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019. 2

[39] Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. Siam R-CNN: visual tracking by re-detection. In *CVPR*, 2020. 6, 7, 8

[40] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. SPM-tracker: Series-parallel matching for real-time visual object tracking. In *CVPR*, 2019. 1, 2

[41] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017. 5

[42] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 2, 3, 5, 6, 7

[43] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. RANet: Ranking attention network for fast video object segmentation. In *ICCV*, 2019. 3

[44] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. YouTube-VOS: Sequence-to-sequence video object segmentation. In *ECCV*, 2018. 5

[45] Yinda Xu, Zeyu Wang, Zuoxin Li, Yuan Ye, and Gang Yu. SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 2020. 2, 7

[46] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013. 5

[47] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R. Scott. Deformable siamese attention networks for visual object tracking. In *CVPR*, 2020. 7

[48] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020. 7

[49] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2, 4

[50] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 4

[51] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 2, 3