Appendix

In this supplementary material, we first provide some details mentioned in main text in Sec.A. We then report more exploration studies of our method in Sec.B and provide some visualizations in Sec.C.

A. Experiment Details

A.1. Training Details

ImageNet Pre-training. We follow DeiT [**?**, **?**] and apply random cropping, random horizontal flipping, label-smoothing regularization, mixup, CutMix, and random erasing as data augmentations. We initialize the weights with a truncated normal distribution. During training, we employ AdamW with a momentum of 0.9, a mini-batch size of 128, and a weight decay of 5×10^2 to optimize models. The initial learning rate is set to 1×10^3 and decreases following the cosine schedule. All models are trained for 300 epochs from scratch on 8 V100 GPUs. For test in ImageNet [**?**], we apply a center crop on the validation set, where a 224×224 patch is cropped to evaluate the classification accuracy.

Finetune on Tracking. We finetune the whole model on the tracking datasets. In particular, for each pair of search/template images from the training dataset, we compute the losses based on the classification and regression outputs from the prediction head. We use standard cross-entropy loss for the classification loss: all pixels within the ground-truth box are regarded as positive samples and the rest are negative. We use GIoU [**?**] loss and L_1 loss for the regression loss. We load the pre-trained SBT parameters. We use 8 tesla V100 GPUs and set the batch size to be 20 for each GPU. Since our model does not have batch normalization, it is not sensitive to the batch size. The batch size can be flexibly adjusted based on the hardware. The search area factor of template and search image is set to 2 and 4, respectively. For GOT-10k training set, the sample pairs of each epoch is $50,000$. The learning rate is set to be 10^{-4} for the feature extraction network, and 10^{-3} for the rest. The learning rate decays at the $30_{th}, 50_{th}$ epoch. We finetune the model for 100 epochs. For full-dataset training, it includes the train subsets of LaSOT [**?**], GOT-10K [**?**], COCO2017 [**?**], and TrackingNet [**?**]. All the forbidden sequences defined by the VOT2019 challenge are abandoned. The pairs of training images in each iteration are sampled from one video sequence or constructed by a static image. On static images, we also construct an image pair by applying data augmentation like flip, brightness jittering and target center jittering.

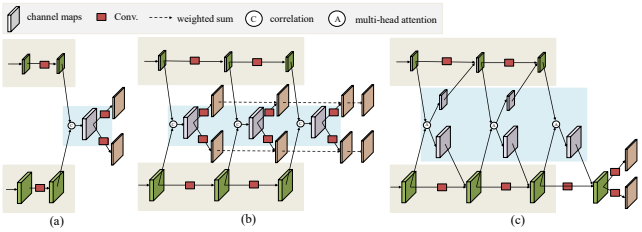

Figure 1. (a): a simple Siamese tracking baseline; (b): Siamese tracking baseline with layer-wise aggregation; (c): correlation-embedded structure in SBT.

Training loss. To validate the generality of our framework, we adopt a vanilla anchor-free prediction head following [**?**] which employs the standard binary cross-entropy loss for classification, which is defined as

$$\mathcal{L}_{cls} = -\sum_{j}[y_j \log(p_j) + (1-y_j)\log(1-p_j)], \quad (1)$$

where y_j denotes the ground-truth label of the j-th feature token, $y_j = 1$ denotes foreground, and p_j denotes the predicted confidence value belong to the foreground. For regression, we apply two kinds of loss: ℓ_1-norm loss $\mathcal{L}_1(.,.)$ and the generalized IoU loss $\mathcal{L}_{GIoU}(.,.)$ [**?**]. The regression loss is as follows:

$$\mathcal{L}_{reg} = \sum_{j} \mathbb{1}_{\{y_j=1\}}[\lambda_G \mathcal{L}_{GIoU}(b_j, \hat{b}) + \lambda_1 \mathcal{L}_1(b_j, \hat{b})], \quad (2)$$

where $y_j = 1$ denotes the positive sample, b_j denotes the j-th predicted bounding box, and $\hat{b}$ denotes the normalized ground-truth bounding box. We set $\lambda_G = 5$ and $\lambda_1 = 7$ and 12 for classification loss in our experiments.

Inference Details. For SBT tracker, during inference, the regression head and classification head generate two response maps which embed estimated size shapes and location confidence values, respectively. The maximum confidence value and its bounding box size are chosen to be final predicted target. The template and search image size are chosen to 128×128 and 256×256, respectively. To validate the effectiveness of our feature network, no other tricks such as template update and online module are adopted.

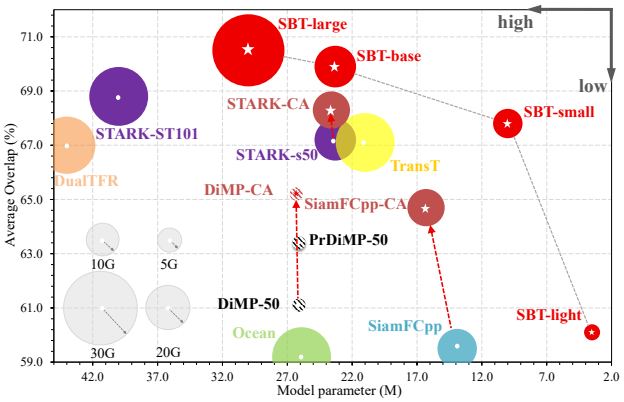

Figure 2. Comparison to the state-of-the-art methods on the GOT10k dataset. The radius of a circle represents the number of FLOPs of the model. Multiple trackers (with suffix "CA") can benefit from our correlation-aware features. We do not show the Flops of DCF methods because of the online learning.

Ablation Details. In Fig. 1, It shows the difference among a simple Siamese tracking baseline, Siamese tracking baseline with layer-wise aggregation and correlation-embedded structure in SBT. For the Siamese pipeline, the template features are center-cropped with he spatial size of 7×7, and then perform correlation with the search features. For cross attention, the template features dose not need to be cropped for the strong global modelling of attention scheme. The channel dimension of the template and search features are adjusted to 256 by a convolutional layer, which is the same with SiamRPNpp [**?**].

A.2. Correlation-Aware Trackers

SiamFCpp-CA: SiamFCpp [**?**] is a recent typical Siamese tracker. We replace the GoogLeNet [**?**] with the modified SBT-small version. **DiMP-CA:** DiMP [**?**] is a modern DCF tracker.. For training and inference, we feed the search and template image pair to modified SBT-base to obtain the correlation-aware search features, then it replaces the original search features extracted by ResNet [**?**]. **STARK-CA:** STARK [**?**] is a recent strong Transformer-based tracker. We replace the ResNet-50 [**?**] in STARK with modified SBT-small. **STM-CA:** STM [**?**, **?**] is a video object segmentation method. We replace the original ResNet-50 [**?**] to prove that our network can also be used in pixel-wise tracking.

A.3. Performance, Model Size and Flops

As shown in Fig. 2, we provide a comprehensive comparison in GOT-10k in terms of AO, model size and computation Flops between our methods and other trackers.

Setting	model	Pre.	Low	Mid	High	SR_{50}	SR_{75}	AO
A_1	SBT-base	✓	B6	B8	B10	75.3	58.6	65.0
A_2	SBT-base	✓	B8	B9	B10	70.8	52.9	61.0
A_3	SBT-base	✓	B6	B9	B10	73.1	53.9	63.2
A_4	SBT-base	✓	B4	B9	B10	75.2	58.0	64.6
A_5	SBT-base	✓	B2	B9	B10	75.7	59.3	65.7

Table 1. Position pattern studies on SBT-base model. Results are from GOT-10k test set. B6 denotes the 6^{th} block in third stage. Pre. denotes whether use pre-trained weights or not.

Setting	model	Pre.	N1	N2	N3	SR_{50}	SR_{75}	AO
B_1	SBT-base	✗	4	4	12	72.3	54.9	62.3
B_2	SBT-base	✗	4	4	14	74.4	57.3	64.0
B_3	SBT-base	✗	2	2	10	72.4	52.4	61.5
B_4	SBT-base	✗	2	2	13	72.3	54.1	62.0
B_5	SBT-base	✗	2	2	15	73.4	55.7	63.1

Table 2. Block pattern studies on GOT-10k test set. N1 denotes the block number in the first model stage. other settings including interleaved CA block and channel dimension are the same.

B. More Studies on Model Variants

We further choose the base version of SBT to explore the impacts of model variants on tracking performance.

B.1. Position Pattern.

In Tab. 1, we set the total number of EoC-CA block to be 3 and ablate their position pattern during inference. When the last two correlation operation are set to the 9_{th} and 10_{th} block, moving the first correlation block to the earlier position achieves better results (63.2% of A_3 vs. 64.6% of A_4 vs. 65.7% of A_5). It clearly validates that there is a strong correlation between earliest EoC-CA position and tracking performance. It also points out our key insight that early-cross generates target-dependent features which help tracker to see better. From the comparison among 65.0% of A_1, 61.0% of A_2 and 63.2% of A_3, the interleaved EoC-SA/EoC-CA pattern outperforms the Siamese-style and earlier-cross design. It also enlighten us to choose a interleaved design pattern of SBT tracking. The underlying reason is related to the feature representation: EoC-SA block can refine the template/search feature after the correlation, resulting in a more expressive feature space for matching. We also point out that the position pattern can be flexibly designed based on the requirements from future researcher/engineer.

B.2. Block Number.

In Tab. 2, we keep other network factors be the same and ablate their block numbers in different stage. 64.0% of B_2 achieves the best tracking performance which has moderate number of blocks $(4/4)$ in the shallow model stage and the second large number of blocks (14) in the deep stage. It indicates that a moderate size of blocks in shallow stage is necessary. Putting the most of blocks in the third stage is

Setting	model	Pre.	C1	C2	C3	SR_{50}	SR_{75}	AO
C_1	SBT-base	✗	64	128	256	74.8	55.1	63.6
C_2	SBT-base	✗	32	64	128	69.8	46.5	58.7
C_3	SBT-base	✗	96	192	384	74.3	55.3	63.7
C_4	SBT-base	✗	128	256	512	73.0	55.2	62.8
C_5	SBT-base	✗	32	64	512	72.0	51.1	61.1
C_6	SBT-base	✗	64	128	512	72.8	52.7	61.8
C_7	SBT-base	✗	128	256	512	74.1	55.8	63.6

Table 3. Channel dimension studies on GOT-10k. C1 denotes the channel dimension in the first model stage. other settings including interleaved CA block and block number are the same.

more practical since the spatial size of features is reduced. Obviously, the tracking performance is highly related the size of overall model. Finally, it enlightens us that we can design different versions of SBT trackers to meet the requirements on speed and model size. It is also quite practical and easy to be implemented by stack different number of EoC blocks.

B.3. Channel Dimension.

In Tab. 2, we keep other network factors be the same and ablate their channel dimension in different stage. Since the channel dimension is quite essential to the model size of SBT, it is vital to investigate their impacts on tracking performance. It is natural that larger dimension in each stage can achieve a better performance which also increases the model size (63.6% of C_1, 63.7% of C_3 and 63.6% of C_7). It is quite interesting to observe that the gap between shallow stage and deep stage is harmful to the performance. When the shallow stage is set to have 64 and 128 channel dimension, then the third stage with 512 dimension performs worse than that with 256 dimension(63.6% of C_1 vs. 61.8% of C_6). It indicates that we should adopt a progressive increment design for the channel dimension in SBT. Moreover, too much channels for the shallow stage encoding seems not helpful but increases the model size a lot (63.6% of C_1 vs.63.6% of C_7). In summary, it is more practical to choose some channel dimension numbers which are widely seen in CNN networks. We can also modify the channel dimensions based on the requirements on different speed and performance.

C. Visualization Result

C.1. Attention Visualization

As shown in Fig. 3, the attention weights focus on the background context of the search area in the shallow stage. We also vividly observe that it effectively suppresses non-target features in the search image layer by layer. It clearly illustrates that our attention block can discriminate the distractors to some extent. In the last block, the attention weight is changed to an uniform distribution which indicates that the search features are ready to the prediction networks. Our Single Branch Transformer (SBT) network allows the features of the two images to deeply interact with each other at the stage of feature extraction which can have dynamic instance-varying behaviors.

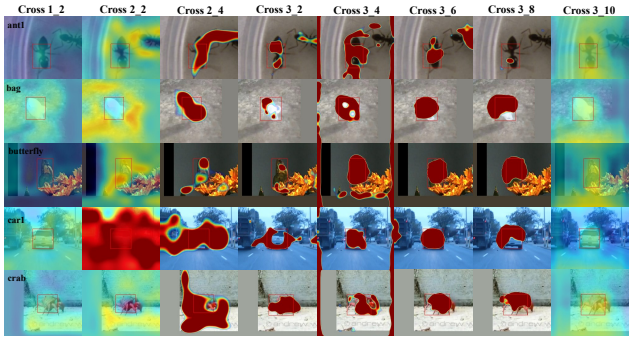

Figure 3. Cross attention map in each stage of SBT tracker.

Figure 4. Visualization of classification map on three cases. (a) denotes suitable template with distractors. (b) denotes template drift with closely attached distractors. (c) denotes template drift with distractors on the edge.

C.2. Response Visualization

As shown in Fig. 4, we visualize the hard case (basketball video) with numerous distractor objects. Our correlation-aware features can discriminate the distractors in a fine-grained level. When the template drifts, our SBT tracker can also tend to make a more reasonable choices. When time index is 214, the man with white clothes are suppressed in SBT while has higher reponse value in other three models. The distractor object with green clothes is almost the same the target which can not be identified by human. Thus, it is reasonable to have high response values for the tracker. The other three models perform worse than SBT. When the time index is 464, though the distractor object with green clothes is similar to the target, but can be identified by the white number on the clothes. SBT identifys this case successfully while other three fails. It clearly indicates that SBT can have more fine-grained discriminative ability among those appearance-based methods.

Figure 5. Failure case. SBT tracker (base) fails on the case of occluded target/out of search region.

C.3. Failure Case

When the target object is occluded with distractor objects, together with appearance changes, the pairwise tracking pipeline is hard to figure out the target. It is also commonly seen in many Siamese trackers. Therefore, our framework struggles to handle the heavy occlusion (e.g., Fig 5) or out-of-view. Another potential limitation of our work is that modern scientific computation packages are not friendly to fast attention computation.

C.4. TSNE Visualization of Features

In Fig. 6 and Fig. 7, we visualize the TSNE of features from our target-dependent network and standard target-unaware Siamese extraction network. When the our SBT network goes deeper, the features belonging to the target (green) become more and more separated from the background and distractors (pink). In the meantime, the search features from the Siamese extraction are totally target-unaware which heavily rely on the separated correlation step to discriminate the targets from background.

C.5. Visualization of Correlation-Aware Trackers

In Fig. 8, we visualize the tracking results of SiamFCpp (first row), SiamFCpp-CA (second row), our SBT tracker (third row) on the challenging sequences from OTB100. We can see that SBT shows stronger discriminative ability and better accuracy throughout tracking. The reponse map of SBT is more centralized and much higher comparing to the background pixels which shows the tracker preserves more spatial information and more discriminative towards the disctractor objects. We can also observe that SiamFCpp-CA has more stronger discriminative ability than original SiamFCpp with standard Siamese extraction network towards distractor objects and background clutters. The response map from our correlation-aware features are more discriminative towards background clutters and more suitable for a instance-level task.

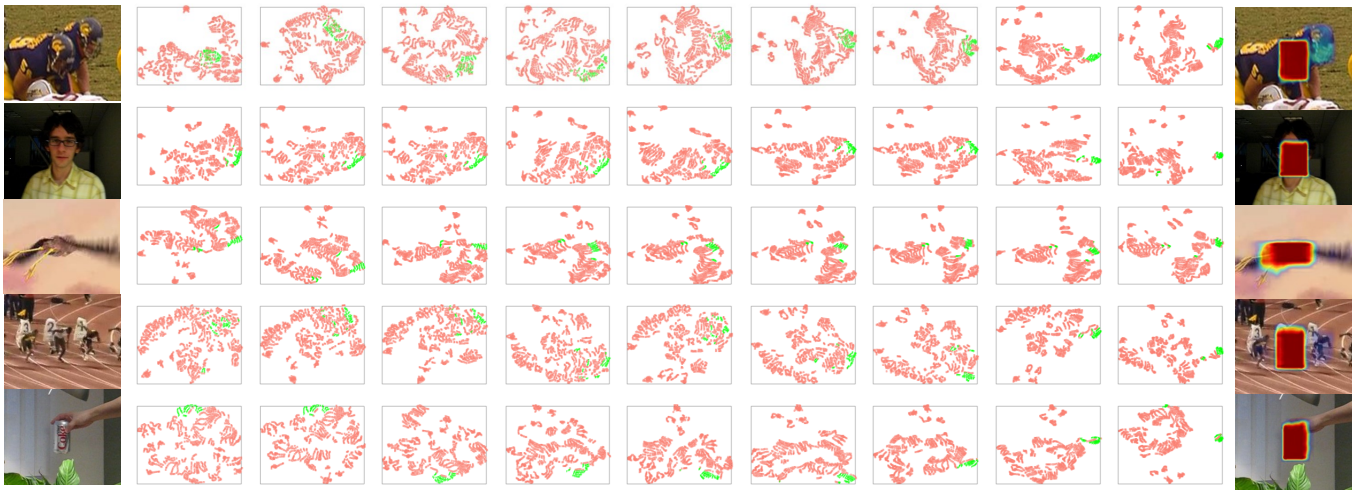

Figure 6. TSNE [**?**] visualizations of search features in correlation-embedded SBT tracker when feature networks go deeper.

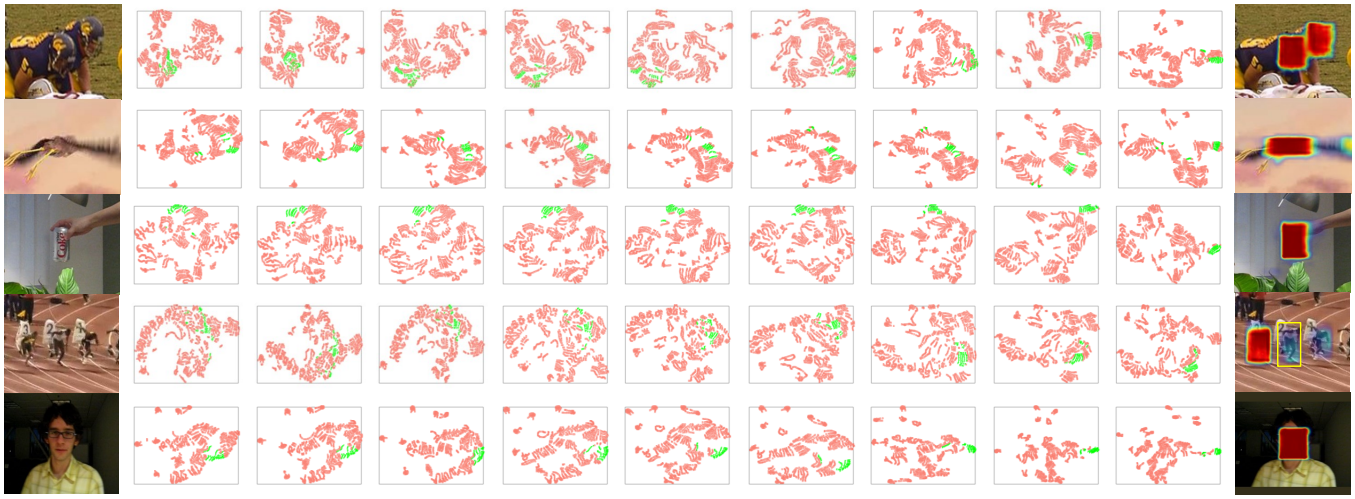

Figure 7. TSNE [**?**] visualizations of search features in SBT tracker with Siamese-like extraction when feature networks go deeper.

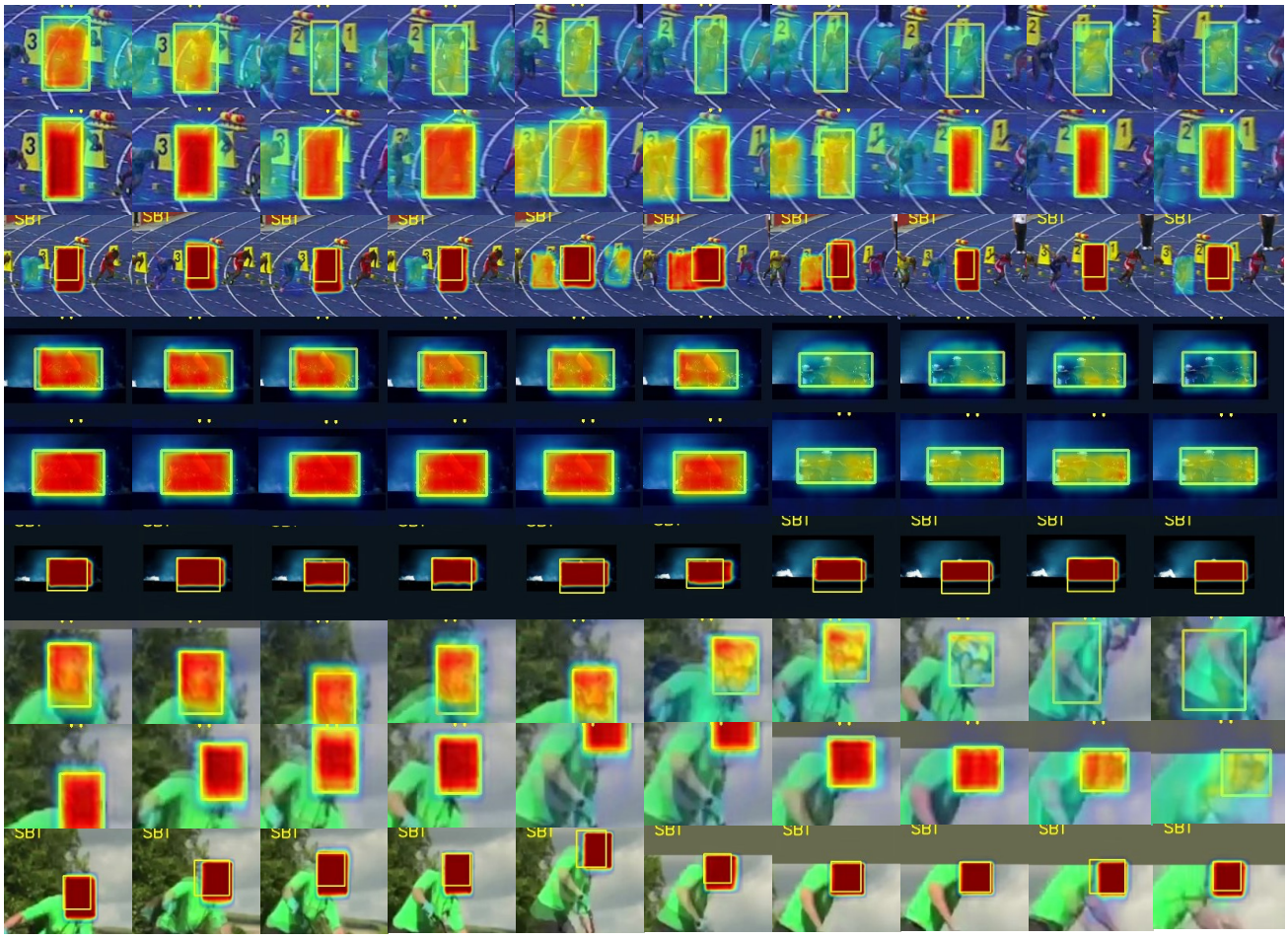

Figure 8. Visualization tracking results of SiamFCpp (first row), SiamFCpp-CA (second row), our SBT tracker (third row) on the challenging sequences from OTB100. We can see that SBT shows stronger generalization ability and better accuracy throughout tracking. SiamFCpp-CA has more stronger discriminative ability than original SiamFCpp with standard Siamese extraction network towards distractor objects and background clutters. Best viewed with zooming in.