Improving Accuracy and Generalization for Efficient Visual Tracking

Ram Zaveri Shivang Patel Yu Gu Gianfranco Doretto
West Virginia University
Morgantown, WV 26506, USA
{rz0012, sap00008, yugu, gidoretto}@mix.wvu.edu

Abstract

Efficient visual trackers overfit to their training distributions and lack generalization abilities, resulting in them performing well on their respective in-distribution (ID) test sets and not as well on out-of-distribution (OOD) sequences, imposing limitations to their deployment in-the-wild under constrained resources. We introduce SiamABC, a highly efficient Siamese tracker that significantly improves tracking performance, even on OOD sequences. SiamABC takes advantage of new architectural designs in the way it bridges the dynamic variability of the target, and of new losses for training. Also, it directly addresses OOD tracking generalization by including a fast backward-free dynamic test-time adaptation method that continuously adapts the model according to the dynamic visual changes of the target. Our extensive experiments suggest that SiamABC shows remarkable performance gains in OOD sets while maintaining accurate performance on the ID benchmarks. SiamABC outperforms MixFormerV2-S by 7.6% on the OOD AVisT benchmark while being 3x faster (100 FPS) on a CPU.

1. Introduction

Tracking a single object, given the location at the first frame, has been an ongoing challenge in the vision community for decades. Most recent approaches provide reasonably good performance [9, 17, 19, 55, 61], especially when benchmarked on *in-distribution (ID)* datasets, *i.e.*, on the testing portion of the same datasets used for training. However, they incur high computational costs and hardware constraints, making their deployment "in-the-wild" in mobile, autonomous, and IoT applications still challenging.

The best-performing Transformer-based trackers operate between 0.4 to 4 frames per second (FPS) on a CPU [9, 58], which is considered "slower than real-time" in many applications. Siamese tracking approaches provide the highest speed. FEAR-XS [6] can operate at 100 FPS on a CPU, whereas an efficient Transformer-based approach, MixFormerV2-S [12], operates at 37 FPS on a CPU. Despite the significant progress on efficient trackers, they still need to catch up when tested on *out-of-distribution (OOD)* datasets, *i.e.*, those that were not used during training.

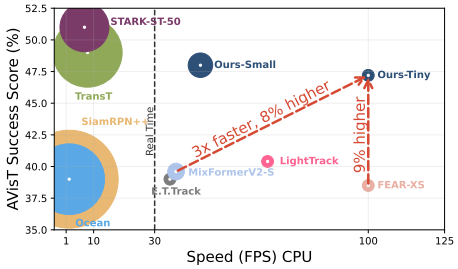

Figure 1. Comparison of our trackers with others on the AVisT [45] dataset on a CPU. We show the success score (AUC) (vertical axis), speed (horizontal axis), and relative number of FLOPs (circles) of the trackers. Our trackers outperform other efficient trackers in terms of both speed and accuracy.

A recently proposed benchmark, AVisT [45], involves tracking objects under extreme visibility conditions that are common in-the-wild but not in most current training sets. High-performing tracking approaches tend to struggle when tested on AVisT, showing very significant performance deterioration. For instance, MixFormerV2-S exhibits remarkable performance with AUC of 58.7% on an in-distribution benchmark like GOT-10k [22]; however, it struggles on the OOD benchmark AVisT, with an AUC of 39.6%. Therefore, the trade-off between the need of computational resources and *OOD generalization* abilities of visual trackers is still unsatisfactory for their deployment in-the-wild under resource constraints.

In this work we aim at significantly improving the trade-off mentioned above. We design a new Siamese tacker that preserves the high speed, reduced memory and computing requirements of the Siamese family while improving OOD generalization to near SOTA-level performance. From the architectural point of view, we make two key contributions. First, we better facilitate the visuo-temporal bridge between the static image template representing the target, and the search region image at current time. While [6] popularized the use of a dual-template, which we also adopt, we introduce the use of a *dual-search-region*. This will allow the tracker to stay anchored to the initial target rep-

resentation while better latching onto its dynamic appearance variations. Second, we design a new learnable layer, the *Fast Mixed Filtration*, that acts as an efficient filtration method for enhancing the relevant components of the combination of the representations forming the dual-template, as well as the dual-search-region. This is important because, given also the reduced representational capacity of smaller backbones used by efficient trackers, directly fusing the representations of the dual-template does not necessarily improve performance [6].

From the learning point of view we make two additional contributions. First, we introduce a new *transitive relation loss* to help bridge the visuo-temporal similarities of the filtered representations of the dual-template and the dual-search-region, so that the relevant relational differences between them can be effectively leveraged for tracking purposes. Second, we more directly address the OOD generalization issue by tackling the dynamic distribution shifts while doing inference. As shown in many test-time adaptation (TTA) approaches for classification [33, 41, 44, 46, 49, 52], shifts in Batch-Normalization (BN) statistics are majorly responsible for performance degradation under OOD testing. We introduce a *dynamic TTA (DTTA)* approach specifically tailored to tracking. It is backward-free, thus lightweight computationally, and aims at dynamically updating the BN statistics while keeping them anchored to the source statistics. To the best of our knowledge this is the first work that uses TTA for single object visual tracking.

Combining the contributions above lead even our smallest and most efficient tracker, S-Tiny, to surpass relevant SOTA approaches on numerous benchmarks. Most notably, on the AVisT benchmark, S-Tiny achieves the AUC of 47.2% while running at 100 FPS on a CPU, outperforming MixFormerV2-S by 7.6% while being almost 3x faster. See Figure 1. An extensive set of experiments with multiple datasets and other approaches shows additional compelling results in support of our method.

2. Related Works

Efficient Tracking. Practical applications require object trackers to be efficient as well as accurate. Siamese-based [27,40] trackers [3,11,21,29,30,54,60,63,65,67] are efficient as they use a separate two stream feature-extraction framework. LightTrack [62] and FEAR [6] introduced lightweight siamese-based trackers, however, lack accurate performance. Transformer-based approaches [9, 17, 55, 61] show reasonable accuracy, however, they lack efficiency as they utilize computationally heavy attention layers. To alleviate that, E.T.Track [5] incorporates an efficient Exemplar Transformer block on the prediction heads. While HCAT [8] uses multiple hierarchical cross-attention blocks with feature sparsification, HiT [24], instead, leverages a lightweight hierarchical transformer backbone to achieve improved accuracy and speed. MixformerV2 [12] uses distillation to reduce the number of FLOPs, whereas SMAT [18] uses separable mixed attention to maintain the accuracy on their one-stream transformer networks. Since these are smaller networks, they have limited representational capacity and do not generalize well to OOD sets, making them less reliable for tracking "in-the-wild", which we tackle in the proposed framework.

Efficient Attention. [39] proposed separable transformer blocks with convolutional layers to increase efficiency while maintaining accuracy. MobileViTv3 [51] further replaced heavy transformer blocks with their CNN-ViT-based separable attention blocks, and showed considerable performance gain. Originally, CBAM [56], DANet [16], and Polarized Self-Attention [37] explored attention in convolution by computing channel and spatial attentions separately. CBAM [56] and PSA [37] further incorporate a squeeze-and-excite framework [23] to excite relevant features across the channels. The latter is a more powerful and efficient variant of CBAM. This suggests that separability is inevitable for efficient attention blocks; therefore, in this work we propose a simplified fast convolution-based separable attention framework.

Efficient Adaptation. To tackle the dynamic distribution shifts during inference, we focus on Test-Time Adaptation (TTA). A recent popular approach, CoTTA [53] uses data augmentation at test time to generate pseudo-labels and performs distillation for the image classification task. TENT [52] and EATA [44] use model's test-time entropy to update only the BN learnable parameters. DUA [41], Momentum [49], IN [46], and AdaBN [33] use backward-free BN-statistics updates to perform adaptation with maximal efficiency while showing considerably good accuracy. Most TTA approaches experience performance deterioration when used under real-world online applications, except BN-adaptation approaches as they are efficient and reliable [2]. Therefore, we propose an efficient instance-level BN update strategy that continuously adapts the model to follow the dynamic visual changes of the target.

3. Methods

Overview. We introduce a tracker that maintains four data sources. There is the *static image template* I_T that represents an object. The *dynamic image template* I_D instead, represents the object at a time $t - \Delta t$, where t is the current time. There is a *search region image* I_t, where the object is presumed to be located at current time t. Unlike previous trackers, we also maintain a *dynamic search region image* I_S, which is the image of the search region at time $t - \Delta t$ re-centered at the object position, i.e., it contains I_D in the center. So, besides the *dual-template*, (I_T, I_D), our tracker incorporates temporal information also via the *dual-search-region*, (I_S, I_t). Specifically, the static template anchors the

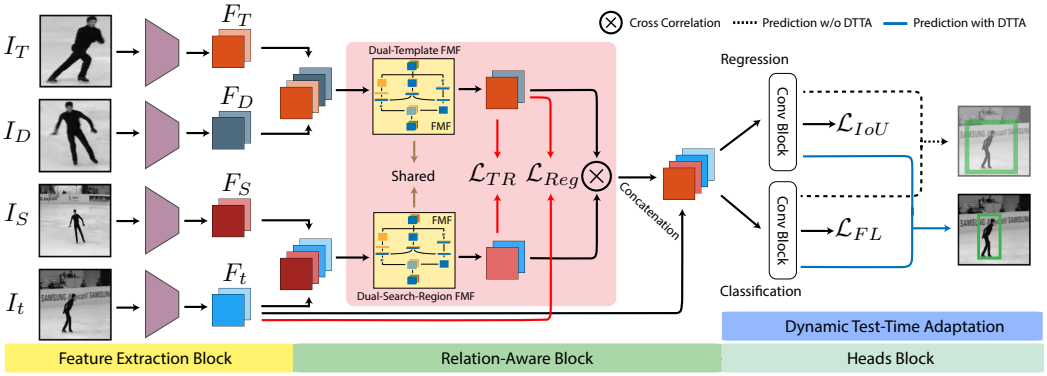

Figure 2. **Overall Architecture.** The Feature Extraction Block uses a readily available backbone to process the frames. The Relation-Aware Block exploits representational relations among the dual-template and dual-search-region through our losses, $\mathcal{L}_{TR}$ and $\mathcal{L}_{Reg}$, where dual-template and dual-search-region representations are obtained via our learnable FMF layer. The Heads Block learns lightweight convolution layers to infer the bounding box and the classification score through standard tracking losses, $\mathcal{L}_{IoU}$ and $\mathcal{L}_{FL}$ respectively. During inference, the tracker adapts to every instance through our Dynamic Test-Time Adaptation framework.

tracker at the object representation at time $t = 0$ (A), the dynamic template and the dynamic search region represent time $t - \Delta t$ (B), and the search region represents time t (C). The dual-template will lead to a boosted object representation that bridges the time gap $t - \Delta t$ (from A to B), while the dual-search-region will lead to a boosted search region representation that bridges the time gap Δt (from B to C). Since we use a siamese architecture and leverage the relations between points in time A and B, and between points B and C, then blend them, we name our approach *SiamABC*.

SiamABC utilizes a feature extraction backbone, a new *Fast Mixed Filtration (FMT)* module, a *Pixel-wise Cross-Correlation* module, and heads for classification scores and bounding box regressions. All the inputs, I_T, I_D, I_S, and I_t, go through the backbone $F(\cdot)$, giving us F_T, F_D, F_S, and F_t respectively. Next, the pair (F_T, F_D), and the pair (F_S, F_t) go through the FMT module, producing $\Omega(F_T, F_D)$ and $\Omega(F_S, F_t)$, respectively. Then, the Pixel-wise Cross-Correlation module, $CC(\cdot, \cdot)$, computes the correlation between $\Omega(F_T, F_D)$ and $\Omega(F_S, F_t)$, boosting their representational relations. The output of $CC(\cdot, \cdot)$ is further processed by the classification head, $CH(\cdot)$, and the bounding box regression head, $BH(\cdot)$, to produce the final tracking output. To learn representations that enable tracking by bridging from A to C, we introduce a new *transitive relation* loss. Finally, to further adapt to dynamic shifts of the input distribution, which are typical when tracking is deployed "in-the-wild", on out-of-distribution data, we endow tracking, for the first time, with a *dynamic backward-free test-time adaptation* approach. See Figure 2.

3.1. Architecture

Feature Extraction Block. For efficiency, we chose the first four layers of FBNetV2 [57] as our Tiny backbone, and the first three layers of ResNet-50 [20] as our Small backbone, all pre-trained on ImageNet [14]. Since the channel and spatial resolution of the backbones can differ, we use an additional convolutional filter (without activation) to match the channel resolution. The backbone takes in the input $x \in \mathbb{R}^{3\times H\times W}$, where $H = W = 128$ for I_T and I_D, and $H = W = 256$ for I_t and I_S. The backbone processes the inputs in parallel, and the weights are shared. This functions in a siamese fashion as described in [3].

Relation-Aware Block. The representations of the dual-template and the dual-search-region are first enhanced by the new Fast Mixed Filtration layer, and then correlated by the Pixel-wise Cross-Correlation module to support tracking.

Fast Mixed Filtration. The dual-template features and the dual-search-region features used by the correlation module $CC(\cdot, \cdot)$ could each simply be the naive concatenation of the respective backbone features in each pair. However, we can potentially improve their combination by processing them with an efficient learnable layer that filters out less useful components while enhancing those important for the task at hand. This should hopefully lead to improved performance. In Figure 4(top), we have shown just that. Especially when tracking in the OOD case, naive feature concatenation severely underperforms the filtered combination.

We consider filtration mechanisms such as self-attention [50], only tailored to convolutional models since they tend to be more efficient on CPU. Polarized Self-Attention (PSA) [37] stands out as it functions as a filter and is a more powerful variant than CBAM [56], with time complexity of $O(CWH)$, where C, W, H, are channel, width, and height of the tensor respectively. Another notable option is the CNN-ViT-based attention block MobileViTv3 [51], based on separable attention [39]. As shown in Table 1, we observed limited performance gain of MobileViTv3 over PSA, with relatively high FLOPs, parameters, and latency. We also noticed that PSA performs several matrix multiplications causing the latency on CPU to still

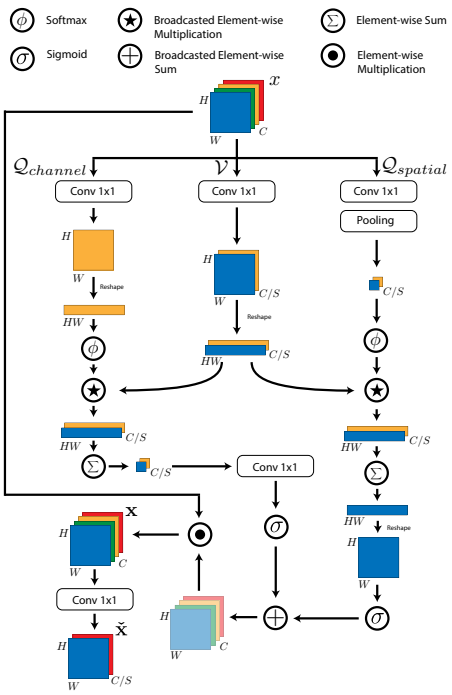

Figure 3. **Fast Mixed Filtration.** This block serves as a lightweight and effective attention mechanism. The input x is filtered to produce the compressed representations $\tilde{\mathbf{x}}$. The broadcast and element-wise operations make this block efficient on CPU.

be considerably high. This motivated the development of our *Fast Mixed Filtration (FMF)*, a new and more efficient mixed filtration method. It is defined as follows

$$\begin{aligned} A_{ch}(x) &= \sigma(W_{ch}(\sum^{HW} W^V(x) \star \phi(W^Q_{ch}(x)))) \ , \\ A_{sp}(x) &= \sigma(\sum^{C} \phi(W^Q_{sp}(x)) \star W^V(x)) \ , \\ \mathbf{x} &= (A_{ch} \oplus A_{sp}) \odot x \ , \end{aligned} \tag{1}$$

where the notation means the following: $\sigma :=$ sigmoid, $\phi :=$ softmax, $W :=$ conv1 $\times$ 1, $\star :=$ broadcasted element-wise multiplication, $\oplus :=$ broadcasted element-wise sum, $\odot :=$ element-wise multiplication, $\sum :=$ element-wise sum, $A_{ch} :=$ channel filter, $A_{sp} :=$ spatial filter, $V :=$ values, $Q :=$ queries, $x :=$ input and $\mathbf{x} :=$ output. Figure 3 depicts the schematic computation of the FMF layer.

Differently than PSA, the improved efficiency of FMF stems from reducing the computation overhead in Equation (1) by limiting the matrix operations to be broadcasted element-wise multiplications and element-wise summations across the vectors, and by setting $W^V_{sp} = W^V_{ch} = W^V$ for both A_{ch} and A_{sp}, which further reduces the number of parameters. Similar to CBAM and PSA, we utilize a squeeze-and-excite framework [23] to excite the relevant features within this block. As shown in Table 1, the latency of FMF is 0.4ms on a CPU, which has decreased from the 0.8ms of PSA, while not experiencing any performance loss.

We use FMF with a squeeze rate $S = 2$ (see Figure 3). In this way, if x is the concatenated representation of the dual-template (i.e., $x_T \doteq F_D \cup F_T$) or of the dual-search-region (i.e., $x_t \doteq F_t \cup F_S$), then $\mathbf{x}$ is the filtered representation with channel dimension $2C$, to which we apply a channel dimension reduction from $2C$ to C, giving us $\tilde{\mathbf{x}}$ (i.e., $\tilde{\mathbf{x}}_T$ or $\tilde{\mathbf{x}}_t$).

Pixel-Wise Cross-Correlation. We combine the filtered dual representations $\tilde{\mathbf{x}}_T$ and $\tilde{\mathbf{x}}_t$ with the block $CC(\cdot)$, which is a pixel-wise cross-correlation. The output is then concatenated with F_t and fed to a one-layer convolution to reduce the number of channels processed by the heads block.

Heads Block. Similar to [6, 66], tracking output is given by a classification head $CH(\cdot)$, and a bounding box regression head $BH(\cdot)$. We use 2 lightweight convolution layers for $CH(\cdot)$ and 4 for $BH(\cdot)$. The last layer of $CH(\cdot)$ has only 1 channel and predicts the foreground/background confidence score for the object, whereas the last layer of $BH(\cdot)$ has 4 channels, each responsible for predicting separate values (x_{min}, y_{min}, x_{max}, and y_{max} for the object bounding box in the frame at time t), which is why we choose a higher number of convolution blocks for $BH(\cdot)$. Following [6], we keep the spatial resolution of these maps to 16×16.

3.2. Training Losses

Transitive Relation Loss (TRL). We help focussing the filtered representations of the dual-template and the dual-search-region on providing relational information that will aid the downstream tasks. To this end, we recognize that $\Omega(F_D, F_T)$ and $\Omega(F_t, F_S)$ should be "similar", since all the inputs contain information about the object. We introduce a loss $\mathcal{L}_{TR}$ to encourage that. At the same time, given the high built in similarity between F_D and F_S (since $I_D \subset I_S$), to avoid learning representations that ignore the static template (F_T) and search region (F_t) information, we also add a regularization loss $\mathcal{L}_{Reg}$ that pulls the representations close to F_t. $\mathcal{L}_{Reg}$ is applied between $\Omega(F_D, F_T)$ and F_t. $\mathcal{L}_{TR}$ and $\mathcal{L}_{Reg}$ are defined as

$$\begin{aligned} \mathcal{L}_{TR} &= \mathcal{D}(\Omega(F_D, F_T), \Omega(F_t, F_S)) \ , \\ \mathcal{L}_{Reg} &= \mathcal{D}(\Omega(F_D, F_T), F_t) \ , \\ \mathcal{D}(x_1, x_2) &= \frac{1}{2}(D(h_1(x_1), h_2(x_2)) + D(h_1(x_2), h_2(x_1))), \\ D(z_1, z_2) &= 1 - \frac{z_1}{||z_1||_2} \cdot \frac{z_2}{||z_2||_2} \ , \end{aligned} \tag{2}$$

where $||\cdot||_2$ is the ℓ_2-norm, h_1 and h_2 are MLP projection heads used only during training, and $D(\cdot, \cdot)$ calculates the cosine distance. Moreover, $\mathcal{D}(\cdot, \cdot)$ is computed as in [7], where we implement the stop-gradient operation to avoid degenerated solutions. We refer to the pair $\mathcal{L}_{TR}$ and $\mathcal{L}_{Reg}$ as the *transitive relation loss (TRL)* since it is meant to bridge the similarities between template and search region

Table 1. Comparison of FLOPs, number of parameters, and latency when using MobileViTv3 [51], Polarized Self-Attention [37] and Fast Mixed Filtration, with their performances on AVisT [45] and LaSOT [15].

Attention	FLOPs (G.)(↓)	Params (M.)(↓)	Latency (ms)(↓)			AVisT		LaSOT	
			CPU	GPU	Nano	AUC(↑)	OP50(↑)	AUC(↑)	Prec.(↑)
MobileViTv3 [51]	0.220	0.862	2.7	1.3	6.5	0.435	0.490	0.568	0.588
PSA [37]	0.051	0.526	0.8	0.25	4.3	0.457	0.529	0.567	0.588
FMF (ours)	0.034	0.395	0.4	0.2	3.5	0.458	0.529	0.572	0.592

images with the aid of the dynamic components so that the relevant relational differences can be highlighted by the downstream blocks for task purposes. The TRL loss improves tracking performance, especially when both components are used. See Figure 4(middle). Notably, the AUC on AVisT [45] improves from 43.7% to 45.8%.

Total Tracking Loss. We use standard losses for the regression and classification heads. We use the IoU loss [48], $\mathcal{L}_{IoU}$, for the bounding box regression, $BH(\cdot)$, and the focal loss [35], $\mathcal{L}_{FL}$, for the classification head, $CH(\cdot)$. We refer to [35, 48] for their definition. The offline training of the tracker is therefore based on the *total tracking loss*

$$\mathcal{L} = \lambda_{IoU}\mathcal{L}_{IoU} + \lambda_{FL}\mathcal{L}_{FL} + \lambda_{TR}\mathcal{L}_{TR} + \lambda_{Reg}\mathcal{L}_{Reg}, \tag{3}$$

where we set λ_{IoU}, λ_{FL}, λ_{TR}, and λ_{Reg} to 1.5, 1.5, 0.5, and 0.5, respectively.

Dynamic Update. Different strategies can be implemented for when to update the dynamic image template and dynamic search region image. Table 6 reports our case-study focussing on parameter-free strategies, which suggests that different strategies are specific to their approaches and are not generally applicable. A simple yet effective strategy that gave us reliable performance is based on maintaining a running average of the classification scores $\overline{\rho}_t$

$$\overline{\rho}_t = (1-\lambda_D)\overline{\rho}_{t-1} + \lambda_D \rho_t \ , \tag{4}$$

where, ρ_t is the score at time t, and λ_D is a momentum parameter, which we set to 0.25. We also start a counter, C, that when it reaches, let us say $N = 60$ frames (≈ 2 seconds), we compare the current classification score with the running average, and if $\rho_t > \overline{\rho}_{t-1}$, then we update the dynamic image components and reset the counter, otherwise we repeat the test at the next iteration. This strategy is parameterless, uses limited computational resources, and is effective.

3.3. Dynamic Test-Time Adaptation

To increase tracking performance, especially at OOD test-time, we introduce a dynamic test-time adaptation procedure based on a batch normalization (BN) correction tailored specifically to tracking. We applied this strategy to the classification and bounding box regression heads. First, BN layers are generally computed by the following equations:

$$BN(x) = \gamma\frac{x - E(x)}{\sqrt{Var(x)}} + \beta, \tag{5}$$

$$\begin{aligned} \overline{\mu}_t &= (1-\alpha)\overline{\mu}_{t-1} + \alpha\mu_t, \\ \overline{\sigma}^2_t &= (1-\alpha)\overline{\sigma}^2_{t-1} + \alpha\sigma^2_t, \end{aligned} \tag{6}$$

where x is the input feature, $E(x)$ and $Var(x)$ are the expected value and variance of x. γ and β are learnable parameters for scaling and shifting. The BN layers keep track of the running mean and variance through Equation (6), where μ_t and σ_t are current expected value and variance respectively, and $\overline{\mu}_t$ and $\overline{\sigma}_t$ are used for $E(x)$ and $Var(x)$, respectively. α is the momentum parameter. Estimating learnable parameters γ and β during testing would require a backward pass, with great detriment to the speed. Therefore, we propose a method that dynamically updates the BN statistics $E(x)$ and $Var(x)$ during testing, which has negligible computational overhead for maintaining speed.

Prior works have explored BN adaptation for classification purposes, with some using a backward pass for adaptation [44, 52], while others introduced backward-free adaptation [33, 41, 46, 49]. However, none are truly applicable for tracking. As we show in Table 5, applying a new Instance-Norm (IN) layer [46] is expensive, and slows down the speed twofold without noticeable improvement. Additionally, the batch size for tracking remains 1, which is too small to make any significant improvement with [49] that applies weighted momentum based on the target batch-size. Given the scale of our architecture and limited number of parameters, by replacing batch statistics with instance statistics, AdaBN [33] does not improve the performance either. DUA [41] uses source statistics as a prior for the incoming task but does not stay anchored to the source statistics, resulting in target statistics drifting away from the original distribution, causing performance drop. Therefore, we propose that BN statistics should be updated with weighted instance statistics while remaining anchored to the source statistics. This results in the following strategy

$$\begin{aligned} \overline{\mu}_{I,t} &= (1-\lambda_{BN})\overline{\mu} + \lambda_{BN}\mu_{I,t} \ , \\ \overline{\sigma}^2_{I,t} &= (1-\lambda_{BN})\overline{\sigma}^2 + \lambda_{BN}\sigma^2_{I,t} \ , \end{aligned} \tag{7}$$

where $\overline{\mu}$ and $\overline{\sigma}^2$ are the final running mean and variance of the model trained on the source data, respectively, and $\mu_{I,t}$, and $\sigma^2_{I,t}$ are mean and variance calculated from the instance at time t, respectively. $\overline{\mu}_{I,t}$ and $\overline{\sigma}^2_{I,t}$ are updated based on the current instance and used for feature normalization at time t. λ_{BN} is set to 0.1. This backward-free *dynamic test-time adaptation (DTTA)* strategy is efficient with negligible difference in latency as shown in Table 5.

Table 2. Comparative study on VOT2020 Benchmark [28]. Red, blue, and green colors describe the best three CPU real-time trackers whereas bold suggests the best CPU non-real-time tracker.

	CPU non-real-time Methods			CPU real-time Methods					
Trackers	STARK-ST50 [61]	STARK-S50 [61]	DiMP [4]	HCAT [8]	E.T.Track [5]	LightTrack [62]	ATOM [13]	MixFormerV2-S [12]	**S-Tiny**
EAO	**0.308**	0.280	0.274	**0.276**	0.267	0.242	**0.271**	0.258	**0.291**
Accuracy	**0.478**	0.477	0.457	**0.455**	0.432	0.422	**0.462**	-	**0.491**
Robustness	**0.799**	0.728	0.740	**0.747**	**0.741**	0.689	**0.734**	-	**0.741**
FPS (CPU)	6	7	**14**	**60**	35	**67**	30	37	**100**
FPS (GPU)	66	66	**127**	**300**	108	170	240	**420**	**425**
FPS (Nano)	10	**12**	10	**24**	10	**17**	13	**40**	**40**

Table 3. Comparative Study with other SOTA approaches on various benchmarks including AVisT [45], NFS30 [25], UAV123 [42], TrackingNet [43], GOT-10k [22], and LaSOT [15]. Red, blue, and green colors describe the best three CPU real-time trackers whereas bold suggests the best CPU non-real-time tracker.

Methods	Out-of-Distribution (OOD) test sets							In-Distribution (ID) test sets							FPS (↑)		
	AVisT [45]			NFS30 [25]		UAV123 [42]		TrackingNet [43]			GOT-10k [22]		LaSOT [15]				
	AUC	OP50	OP75	AUC	Prec.	AUC	Prec.	AUC	P_{norm}	Prec.	AO	$SR_{0.50}$	AUC	Prec.	CPU	GPU	Nano
CPU non-real-time Methods																	
STARK-ST50 [61]	0.511	0.592	**0.391**	0.652	-	0.691	-	0.813	0.861	-	0.680	0.777	0.666	-	**7**	66	10
Ocean [66]	0.389	0.436	0.205	0.573	0.706	0.574	-	-	-	-	0.611	0.634	0.505	0.517	2	70	18
SiamRPN++ [29]	0.390	0.435	0.212	0.596	**0.720**	0.593	-	0.733	0.800	-	-	-	0.503	0.496	1.4	145	10
SiamMask [54]	0.358	0.401	0.185	-	-	-	-	-	-	-	-	-	-	-	4	**308**	20
SiamBAN [11]	0.376	0.432	0.217	0.594	-	0.631	0.833	-	-	-	-	-	0.514	0.598	4	300	**24**
SeqTrack-L384 [9]	-	-	-	0.662	-	0.685	-	**0.855**	**0.895**	**0.858**	0.748	0.819	**0.725**	0.793	0.4	15	-
MixFormerV2-B [12]	-	-	-	-	-	**0.699**	**0.921**	0.834	0.881	0.816	0.739	-	0.706	**0.808**	**7**	130	15
DropMAE [58]	-	-	-	-	-	-	-	0.841	0.889	-	**0.759**	**0.868**	0.718	0.780	4	98	10
TransT [10]	0.490	0.564	0.372	0.657	-	0.691	-	0.814	0.867	0.803	0.723	0.824	0.649	0.690	**7**	85	8
OSTrack-256 [64]	-	-	-	0.647	-	0.683	-	0.831	0.878	0.820	0.710	0.804	0.691	0.752	4	98	18
ToMP-50 [38]	**0.516**	**0.595**	0.389	**0.669**	-	0.690	-	0.786	0.862	0.812	-	-	0.676	0.722	**7**	83	6
CPU real-time Methods																	
HiT-Small [24] [1]	-	-	-	0.618	-	0.633	-	**0.777**	0.819	**0.731**	0.626	0.712	0.605	**0.615**	-	-	-
SMAT [18]	**0.447**	**0.507**	**0.313**	**0.620**	**0.746**	**0.643**	**0.839**	**0.786**	**0.842**	**0.756**	**0.645**	**0.747**	**0.617**	**0.646**	34	158	20
E.T.Track [5]	0.390	0.412	0.227	0.570	0.694	0.623	0.806	0.745	0.798	0.698	0.566	0.646	0.589	0.603	35	108	10
MixFormerV2-S [12]	0.396	0.425	0.227	0.610	0.722	0.634	0.837	0.758	0.811	0.704	0.587	0.672	**0.606**	0.604	37	**420**	**40**
HCAT [8]	0.418	0.481	0.263	**0.619**	0.741	0.636	0.805	0.766	**0.826**	0.729	**0.634**	**0.743**	0.590	0.605	**60**	300	**24**
LightTrack [62]	0.404	0.437	0.242	0.565	0.692	0.617	0.799	0.729	0.793	0.699	0.582	0.660	0.522	0.517	**67**	170	17
FEAR-XS [6]	0.387	0.421	0.220	0.486	0.563	0.610	0.816	0.715	0.805	0.699	0.573	0.681	0.535	0.545	**100**	**450**	**40**
S-Tiny	**0.472**	**0.543**	**0.353**	**0.620**	**0.747**	**0.662**	**0.856**	0.741	0.819	0.720	0.614	0.728	0.590	0.607	**100**	**425**	**40**
S-Small	**0.479**	**0.557**	**0.372**	**0.624**	**0.744**	**0.681**	**0.858**	**0.784**	**0.835**	**0.746**	**0.646**	**0.751**	**0.607**	**0.622**	45	400	**30**

4. Experiments

4.1. Implementation Details

Model Details. With the Tiny backbone, SiamABC consists of 2.03M parameters and uses 0.628 GigaFLOPs. We refer to this tracker as SiamABC-Tiny or S-Tiny. With the Small backbone, SiamABC consists of 9.82M parameters and uses 6.81 GigaFLOPs. We refer to it as SiamABC-Small or S-Small. S-Tiny runs at 100 FPS on a CPU, 425 FPS on a GPU, and 40 FPS on our edge device Jetson Orin Nano, while S-Small runs at 45 FPS (CPU), 400 FPS (GPU), and 30 FPS (Nano).

Training. All the code is written in PyTorch [47]. Both models were trained on a single Nvidia RTX A6000 GPU for 20 epochs. We use a batch size of 32 and ADAM optimizer [26] with a learning rate of 10^{-4}. We allow close to 10^6 samples every epoch by randomly sampling sequences and then images from GOT-10k [22], LaSOT [15], COCO2017 [36], and TrackingNet [43]. We randomly sample a template from a sampled sequence. Further, we randomly choose a search sample from the same sequence with an offset of Δ. The dynamic frames are sampled from the interval between the template and the search samples. We set $\Delta = 150$ arbitrarily to facilitate dynamic updates at longer intervals during testing. The input size of the template is 128×128, and 256×256 for the search frames. For standard augmentations, we crop a template with a size increase offset of 0.2 and a search region with an offset of 2.0. We also apply to the search region crops a random scale and shift factor by uniformly drawing samples from (0.65, 1.35) and (0.92,1.08), respectively. We also apply the color augmentation and use the same post-processing as in [3].

Inference. We evaluate all the trackers on two hardware platforms. One is based on an Nvidia RTX 3090 GPU and 12th Gen Intel i9-12900F CPU. The other is an entry-level GPU-based edge device, Nvidia Jetson Orin Nano, which here we abbreviate to 'Nano'. All the FPS numbers were reproduced using these two platforms.

4.2. Comparison with other Trackers

We evaluate our SiamABC trackers on 11 challenging benchmarks: AVisT [45], VOT2020 [28], LaSOT [15], TrackingNet [43], GOT-10k [22], OTB-2015 [59], TC128 [34], UAV123 [42], NFS30 [25], ITB [32], and DTB70

[1] HiT [24] could not release their code and the performance was not ascertained; therefore, we report Hit-Small from their publication as the most comparable tracker. According to those numbers, we outperform Hit-Small in various benchmarks as shown in Table 3.

Table 4. Comparative study on ITB [32], OTB [59], TC128 [34], and DTB70 [31] benchmarks in terms of their AUC score. Red, blue, and green colors describe the best three CPU real-time trackers whereas bold suggests the best CPU non-real-time tracker.

	CPU non-real-time Methods						CPU real-time Methods				
	DropMAE [58]	TransT [10]	STARK [61]	DiMP [4]	SiamRPN++ [29]	Ocean [66]	E.T.Track [5]	LightTrack [62]	ATOM [13]	**S-Tiny**	**S-Small**
ITB [32]	**0.650**	0.547	0.576	0.537	0.441	0.477	-	-	**0.472**	**0.548**	**0.555**
OTB [59]	**0.696**	0.695	0.681	0.684	0.687	0.684	**0.678**	0.662	0.669	**0.709**	**0.713**
TC128 [34]	-	0.596	**0.626**	0.612	0.577	0.557	-	0.550	**0.599**	**0.617**	**0.630**
DTB70 [31]	-	**0.667**	0.638	-	0.569	0.455	-	**0.491**	-	**0.656**	**0.662**
FPS (CPU)	4	8	7	**14**	1.4	2	35	**67**	30	**100**	**45**
FPS (GPU)	85	66	66	127	**145**	70	108	170	**240**	**425**	**400**
FPS (Nano)	10	8	12	10	10	**18**	10	**17**	13	**40**	**30**

Table 5. Comparative study on test-time adaptation (TTA) approaches on AVisT [45] as it involves various extreme distribution shifts with real-world corruptions and ITB [32] as the next most challenging benchmark. The best results are in bold.

Methods	AVisT [45]				ITB [32]				Latency (ms) ↓		
	AUC	OP50	OP75	Prec.	AUC	OP50	OP75	Prec.	CPU	GPU	Nano
No TTA	0.458	0.529	0.340	0.413	0.539	0.659	0.483	0.631	**3.6**	**0.6**	**9.8**
Backward-Based Methods											
TENT [52]	0.460	0.518	0.356	0.417	0.530	0.635	0.472	0.610	9.9	2.9	30.4
ETA [44]	0.459	0.516	**0.360**	0.416	0.525	0.630	0.470	0.606	9.9	2.8	35.6
Backward-Free Methods											
Momentum [49]	0.452	0.513	0.341	0.411	0.540	0.656	0.484	0.632	3.7	0.7	15.1
DUA [41]	0.427	0.484	0.307	0.394	0.516	0.628	0.467	0.591	3.7	0.7	15.1
IN [46]	0.454	0.515	0.342	0.417	0.519	0.628	0.455	0.604	6.6	1.9	23.1
AdaBN [33]	0.456	0.517	0.345	0.414	0.522	0.632	0.461	0.608	3.7	0.7	15.1
DTTA (ours)	**0.472**	**0.543**	0.353	**0.440**	**0.548**	**0.667**	**0.494**	**0.634**	3.7	0.7	15.1

[31].

VOT2020 [28] contains 60 challenging videos and employs EAO (expected average overlap) as its metric alongside the accuracy and robustness. In Table 2, our tracker, S-Tiny, shows remarkable resilience against difficult scenarios in this benchmark, outperforming HCAT [8] (best real-time method) and STARK-S50 [61] by 1.05% and 1.04% EAO respectively while being more than 14x faster than STARK-S50 on a CPU.

AVisT [45] consists of 120 extremely challenging real-world sequences in adverse visibility (see Figure 2 in the supplementary material). In addition to simple occlusion and fast motion, it involves objects under heavy rain, heavy snow, dense fog, sandstorms, hurricanes, etc. In Table 3, we note that this out-of-distribution benchmark highlights a significant performance degradation of SOTA trackers compared to the widely used in-distribution test sets. On the other hand, our trackers, S-Tiny and S-Small, are outperforming HCAT by 5.4% and 6.1% respectively and MixFormerV2-S [12] by 7.6% and 8.3% respectively, showing our approach's improved ability to track in sequences with adverse conditions. Additionally, S-Tiny outperforms SMAT [18] by 2.5% while being almost 3x faster on a CPU.

UAV123 [42] is a benchmark for tracking from an aerial viewpoint involving 123 long video sequences. We outperform HCAT by 2.6% and 4.5% respectively, and SMAT by 1.9% and 3.8% respectively, confirming the OOD generalization ability of our approach. **NFS30** [25] is a benchmark collected with extremely high frame rate of 240 FPS for fast tracking. Similar to other approaches, we use the 30 FPS version of the benchmark for evaluation. Our trackers outperform others also in this OOD test. Please refer to Table 3.

The **LaSOT** [15] benchmark involves 280 long test sequences with 2500 frames per sequence on average. **TrackingNet** [43] is a large benchmark consisting of real-life videos collected from YouTube. There are 511 test videos averging in about 441 frames per sequence. One has to submit the raw data to their evaluation server to obtain the results for a fair evaluation. Similarly, **GOT-10k** [22] is a challenging short term benchmark consisting of 180 test sequences which are evaluated on their server. Most approaches are trained on the large training sets of LaSOT, TrackingNet, and GOT-10k; thus, the test distributions of such benchmarks remain quite similar. Nevertheless, S-Small outperforms most SOTA CPU real-time trackers in these benchmarks while remaining efficient as shown in Table 3. Our approach slightly underperforms SMAT in two ID sets, LaSOT and TrackingNet; however, we note that the trade-off with speed is significant as SMAT runs at 34 FPS (CPU), 158 FPS (GPU), and 20 FPS (Nano), while S-tiny runs at 100 FPS (CPU), 425 FPS (GPU), and 40 FPS (Nano), and S-Small runs at 45 FPS (CPU), 400 FPS (GPU), and 30 FPS (Nano).

As shown in Table 4, we outperform other CPU-based trackers on the **OTB-2015** [59] benchmark with 100 sequences, even surpassing the CPU non-real-time approaches. Another similar benchmark is **TC128** [34] with 128 challenging color sequences, where we outperform other SOTA approaches as well. **DTB70** [31] is another small-scale UAV benchmark involving 70 long sequences. We consistently show improvement here as well. **ITB** [32] is a benchmark with 180 various challenging sequences from many other benchmarks giving an informative evaluation of trackers. ITB is second to AVisT in terms of chal-

Table 6. Case study on parameter-free dynamic updates.

Dynamic Updates (No DTTA)	AVisT AUC	AVisT OP50	LaSOT AUC	LaSOT Prec.	FPS(↑) CPU	FPS(↑) GPU	FPS(↑) Nano
No Updates	0.448	0.511	0.556	0.572	102	430	40
Fixed interval ($N = 60$)	0.445	0.514	0.519	0.533	100	425	40
FEAR-based [6]	0.454	0.517	0.524	0.234	96	405	38
TATrack-based [19]	0.361	0.377	0.414	0.430	60	250	22
Ours	0.458	0.529	0.572	0.592	100	425	40

lenging sequences, and Table 4 confirms a remarkable generalization ability of our approach. We note that DropMAE [58] was pre-tained on additional diverse training data.

4.3. Comparision with Adaptation Approaches

In our S-Tiny model we incorporate TENT [52] and ETA [44] as our TTA baselines with backward-passes, where we use our classification output as self-entropy. Additionally, we also evaluated Momentum [49], DUA [41], IN [46], and AdaBN [33] as our backward-free TTA baselines. We set the batch size to 1. The comparison is shown in Table 5 with two of the most challenging benchmarks, AVisT as it involves multiple adverse scenarios with natural corruptions and ITB as the next most challenging benchmark. TENT and ETA have almost 3x the latency because of the backward basses, and do improve with AVisT. However, we do not observe the same with ITB. Momentum shows negligible improvement, whereas IN, AdaBN, and DUA show performance degradation in both scenarios. When DTTA, our efficient adaptation strategy, is turned on, we notice significant improvement with both benchmarks while having minimal latency.

4.4. Ablation Study

We perform ablation studies on each of the components of S-Tiny used on AVisT as an OOD benchmark and on LaSOT as an ID benchmark. In Figure 4(top), the baseline is obtained by removing the FMF block and the TRL losses, $\mathcal{L}_{TR}$ and $\mathcal{L}_{Reg}$. Next, we add the intermediate dynamic frames to obtain the mix configuration, without FMF, and lastly, we add our FMF block. We clearly observe the impact of the FMF block, especially on the OOD benchmark, where the AUC increases from 41.8% to 43.7%. Next, Figure 4(middle) shows the ablation on the TRL losses, $\mathcal{L}_{TR}$ and $\mathcal{L}_{Reg}$. The addition of either of them improves performance; however, the improvement is more significant when used together. Further, in Table 1 we test the impact of FMF on accuracy compared to PSA [37], observing no noticeable fluctuations in performance. We further evaluate the squeeze rate of our FMF block in Figure 4(bottom). There we notice performance degradation on the OOD benchmark when we do not squeeze the FMF module, but not as much in the ID benchmark, suggesting that squeeze is more important for accurate filtration of OOD sequences. In Table 6, we show a case-study on parameter-free dynamic update strategies, where ours consistently improves over the others.

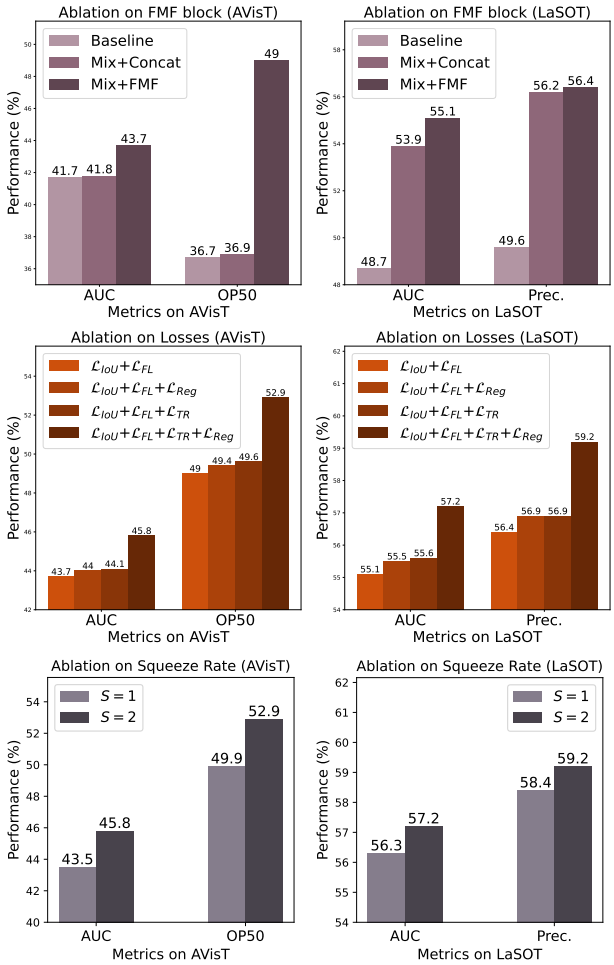

Figure 4. Ablation study on the components of SiamABC-Tiny. Top-row: Ablation on the FMF block. Middle-row: Ablation on TRL losses. Bottom-row: Ablation on squeeze rate.

5. Conclusions

We introduce SiamABC, a new Siamese visual tracker that improves the trade-off between the computational requirements and the OOD generalization ability, thus expanding the horizon of applicability of visual trackers in-the-wild under resource constraints. We have shown that it can be as fast as FEAR-XS, while being significantly more accurate with an evaluation over 11 benchmarks. We have also shown the superior ability of SiamABC in OOD generalization by reaching near-SOTA accuracies on the challenging OOD benchmark AVisT, with a significant improvement over the efficient Transformer-based SOTA methods. We credit this achievement to the four major technical contributions of the approach that include the use of a dual-search-region, the fast filtration layer FMT, the TRL loss, and the introduction, for the first time, of the dynamic TTA during tracking. Promising future extensions of this work may include further development of TTA for tracking, and the adoption of tracking inertia.

Acknowledgments

Research reported in this publication was supported by the National Institute Of Mental Health of the National Institutes of Health under Award Number R44MH125238. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. This material is also based upon work supported by the National Science Foundation under Grants No. 1920920, 2223793.

References

[1] GitHub - got-10k/toolkit: Official Python toolkit for generic object tracking benchmark GOT-10k and beyond — github.com. https://github.com/got-10k/toolkit. 12

[2] Motasem Alfarra, Hani Itani, Alejandro Pardo, Shyma Alhuwaider, Merey Ramazanova, Juan C Pérez, Zhipeng Cai, Matthias Müller, and Bernard Ghanem. Revisiting test time adaptation under online evaluation. *arXiv preprint arXiv:2304.04795*, 2023. 2, 13

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, pages 850–865. Springer, 2016. 2, 3, 6

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6182–6191, 2019. 6, 7

[5] Philippe Blatter, Menelaos Kanakis, Martin Danelljan, and Luc Van Gool. Efficient visual tracking with exemplar transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1571–1581, 2023. 2, 6, 7

[6] Vasyl Borsuk, Roman Vei, Orest Kupyn, Tetiana Martyniuk, Igor Krashenyi, and Jiři Matas. Fear: Fast, efficient, accurate and robust visual tracker. In *European Conference on Computer Vision*, pages 644–663. Springer, 2022. 1, 2, 4, 6, 8

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021. 4

[8] Xin Chen, Ben Kang, Dong Wang, Dongdong Li, and Huchuan Lu. Efficient visual tracking via hierarchical cross-attention transformer. In *European Conference on Computer Vision*, pages 461–477. Springer, 2022. 2, 6, 7

[9] Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14572–14581, 2023. 1, 2, 6

[10] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8126–8135, 2021. 6, 7

[11] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6668–6677, 2020. 2, 6

[12] Yutao Cui, Tianhui Song, Gangshan Wu, and Limin Wang. Mixformerv2: Efficient fully transformer tracking. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 6, 7

[13] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4660–4669, 2019. 6, 7

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[15] Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Harshit, Mingzhen Huang, Juehuan Liu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision*, 129:439–461, 2021. 5, 6, 7, 12

[16] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3146–3154, 2019. 2

[17] Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. Aiatrack: Attention in attention for transformer visual tracking. In *European Conference on Computer Vision*, pages 146–164. Springer, 2022. 1, 2

[18] Goutam Yelluru Gopal and Maria A Amer. Separable self and mixed attention transformers for efficient object tracking. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6708–6717, 2024. 2, 6, 7

[19] Kaijie He, Canlong Zhang, Sheng Xie, Zhixin Li, and Zhiwen Wang. Target-aware tracking with long-term context attention. *arXiv preprint arXiv:2302.13840*, 2023. 1, 8

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[21] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 749–765. Springer, 2016. 2

[22] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021. 1, 6, 7, 12, 13

[23] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer.

Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2, 4

[24] Ben Kang, Xin Chen, Dong Wang, Houwen Peng, and Huchuan Lu. Exploring lightweight hierarchical vision transformers for efficient visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9612–9621, 2023. 2, 6

[25] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1125–1134, 2017. 6, 7

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[27] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015. 2

[28] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking vot2020 challenge results. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 547–601. Springer, 2020. 6, 7, 13

[29] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4282–4291, 2019. 2, 6, 7

[30] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8971–8980, 2018. 2

[31] Siyi Li and Dit-Yan Yeung. Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In *AAAI*, 2017. 7

[32] Xin Li, Qiao Liu, Wenjie Pei, Qiuhong Shen, Yaowei Wang, Huchuan Lu, and Ming-Hsuan Yang. An informative tracking benchmark. *arXiv preprint arXiv:2112.06467*, 2021. 6, 7

[33] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 2, 5, 7, 8

[34] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. *IEEE transactions on image processing*, 24(12):5630–5644, 2015. 6, 7

[35] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 6, 12

[37] Huajun Liu, Fuqiang Liu, Xinyi Fan, and Dong Huang. Polarized self-attention: Towards high-quality pixel-wise regression. *arXiv preprint arXiv:2107.00782*, 2021. 2, 3, 5, 8

[38] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8731–8740, 2022. 6

[39] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 2, 3

[40] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 378–383. IEEE, 2016. 2

[41] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14765–14775, 2022. 2, 5, 7, 8

[42] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 445–461. Springer, 2016. 6, 7

[43] 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 *Proceedings of the European conference on computer vision (ECCV)*, pages 300–317, 2018. 6, 7, 12, 13

[44] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022. 2, 5, 7, 8

[45] Mubashir Noman, Wafa Al Ghallabi, Daniya Najiha, Christoph Mayer, Akshay Dudhane, Martin Danelljan, Hisham Cholakkal, Salman Khan, Luc Van Gool, and Fahad Shahbaz Khan. Avist: A benchmark for visual object tracking in adverse visibility. *arXiv preprint arXiv:2208.06888*, 2022. 1, 5, 6, 7, 13, 14, 15

[46] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018. 2, 5, 7, 8

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[48] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized in-

tersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. 5

[49] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020. 2, 5, 7, 8

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[51] Shakti N Wadekar and Abhishek Chaurasia. Mobilevitv3: Mobile-friendly vision transformer with simple and effective fusion of local, global and input features. *arXiv preprint arXiv:2209.15159*, 2022. 2, 3, 5

[52] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 2, 5, 7, 8

[53] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 2

[54] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019. 2, 6

[55] Xing Wei, Yifan Bai, Yongchao Zheng, Dahu Shi, and Yihong Gong. Autoregressive visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9697–9706, 2023. 1, 2

[56] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 3

[57] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10734–10742, 2019. 3

[58] Qiangqiang Wu, Tianyu Yang, Ziquan Liu, Baoyuan Wu, Ying Shan, and Antoni B Chan. Dropmae: Masked autoencoders with spatial-attention dropout for tracking tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14561–14571, 2023. 1, 6, 7, 8

[59] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 6, 7

[60] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12549–12556, 2020. 2

[61] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10448–10457, 2021. 1, 2, 6, 7

[62] Bin Yan, Houwen Peng, Kan Wu, Dong Wang, Jianlong Fu, and Huchuan Lu. Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15189, 2021. 2, 6, 7, 12

[63] Kai Yang, Zhenyu He, Zikun Zhou, and Nana Fan. Siamatt: Siamese attention network for visual tracking. *Knowledge-based systems*, 203:106079, 2020. 2

[64] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *European Conference on Computer Vision*, pages 341–357. Springer, 2022. 6

[65] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6728–6737, 2020. 2

[66] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 771–787. Springer, 2020. 4, 6, 7

[67] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 101–117, 2018. 2

Algorithm 1 Sampling Strategy

$X \leftarrow rand(X^n) | X \in X^n$ ▷ sequence
$I_T \leftarrow x_i | x_i \in X$ ▷ static template
$I_t \leftarrow x_j | x_j \in X[i, i+\Delta]$ ▷ search-region
$I_S \leftarrow x_k, x_k \in X[i, j]$ ▷ dynamic-search-region
$I_D \leftarrow crop(I_S)$ ▷ dynamic template

A. Further Details on Methods

A.1. Fast Mixed Filtration

We provide more details regarding the operations performed by the FMF block. Please refer to Figure 3 for a schematic representation of FMT. When fusing the pair of feature maps coming either from the dual-template or the dual-search-region, we first concatenate the two feature maps resulting in a representation with $C + C = 2C$ channels dimension. Here, we aim to enhance the most relevant features through this filtration process. We further want to compress the feature space from $2C \times H \times W \rightarrow C \times H \times W$; therefore, we choose a squeeze rate of $S = 2$. Here, W^V and W^Q_{sp} perform the squeeze for V and Q_{sp}, respectively, for the channel and spatial filters, where an average pooling operation is applied to W^Q_{sp} to decrease the spatial dimensions of W^Q_{sp} from $C \times H \times W \rightarrow C \times 1 \times 1$. Moreover, W^Q_{ch} decreases the channel dimensions of Q_{ch} from $2C \times H \times W \rightarrow 1 \times H \times W$. A softmax function, ϕ, is applied to W^Q_{ch} and W^Q_{sp} to produce the channel and spatial filter masks, respectively. Also, broadcast element-wise multiplications, $\star$, and element-wise summation, $\sum$, are then performed between W^V and $\phi(W^Q_{ch})$ and between $\phi(W^Q_{sp})$ and W^V to produce channel and spatial filters of size $C \times 1 \times 1$ and $1 \times H \times W$, respectively. Since there was a squeeze operation on the channel filter, W_{ch} is applied to the channel filter to unsqueeze the channel dimensions from $C \times 1 \times 1 \rightarrow 2C \times 1 \times 1$, followed by a Layer Normalization layer to normalize the feature distributions. Furthermore, a sigmoid operation, σ, is performed on these filters, and then a broadcast element-wise summation, $\oplus$, is performed to form a total filter map of size $2C \times H \times W$. An element-wise multiplication operation is performed between the full filter map and x to enhance the most important features from x to form the final output $\mathbf{x}$. After this mixed-filtration of x, we further readjust the number of channels from $2C \times H \times W \rightarrow C \times H \times W$, giving us $\tilde{\mathbf{x}}$, to perform future correlations.

Finally, like in [22], the classification and regression bounding box heads have been implemented with separable convolutions.

A.2. Training Sampling Strategy

We sample a random sequence from a database as shown in Algorithm 1. We choose a random frame for the template, I_T. Then, we choose another frame randomly for the search-region, I_t, from an interval of size $\Delta = 150$ right after the frame that contains the template. The dynamic-search-region I_S, and I_D, are randomly sampled from the interval with the lower bound of the template frame and the upper bound of the search-region frame. Here, we should note that I_S contains I_D. We randomly choose 300,000 samples from GOT-10k [22], 100,000 from LaSOT [15], 200,000 from COCO2017 [36], and 400,000 from TrackingNet [43] datasets to collect a total of 10^6 samples every epoch.

A.3. Dynamic Sample Update Strategy

Algorithm 2 describes the details of our dynamic sample update strategy with $N = 60$ and $\lambda_D = 0.25$. This strategy is parameter-less and lightweight.

Algorithm 2 Dynamic Sample Update

$t \leftarrow 0$
$x_t \leftarrow$ sequence frame at time t
$I_T \leftarrow x_t, I_S \leftarrow x_t, I_D \leftarrow x_t$
$init\_tracker(I_T)$ ▷ initialize the tracker
$set\_dyn(I_S, I_D)$ ▷ set the dynmaic samples
$C \leftarrow 0$ ▷ counter
$\overline{\rho}_t \leftarrow 1$ ▷ initialize average classification score
while x_{t+1} is available **do**
  $C \leftarrow C + 1$ ▷ increase the counter
  $I_t \leftarrow x_{t+1}$ ▷ search-region
  $\beta_t, \rho_t \leftarrow tracker(I_t)$ ▷ $\beta \leftarrow$ bounding box
    ▷ $\rho \leftarrow$ classification score
  if $C \geq N$ & $\rho_t > \overline{\rho}_t$ **then**
    ▷ $N \leftarrow$ hyperparameter
    $I_S, I_D \leftarrow update(x_{t+1}, \beta_t)$
    $reset\_dyn(I_S, I_D)$
      ▷ reset the dynmaic samples
    $C \leftarrow 0$
  end if
  $\overline{\rho}_t = (1 - \lambda_D)\overline{\rho}_t + \lambda_D \rho_t$
    ▷ $\lambda_D \leftarrow$ hyperparameter
end while

B. Additional Experiments

B.1. Evaluating Benchmarks

We explain the strategies to evaluate our approach against other trackers. The FPS numbers were reproduced using the codebase provided by LightTrack [62]. We follow GOT-10k [22] guidelines and use their toolkit [1] to evaluate

the generic benchmarks. For VOT2020 [28], we use their provided toolkit for the challenge. For GOT-10k [22] and TrackingNet [43], we send the raw results to their servers for a fair evaluation.

B.2. Test-Time Adaptation Baselines

There are no specific Test-Time Adaptation (TTA) approaches available for single-object tracking. Therefore, to compare our Dynamic TTA strategy mentioned in Section 3.3, we implement relevant backward-free and backward-based baselines according to the codebase provided by [2]. We carefully follow the instructions provided by the respective approaches and use them under our online setting. Here, our approach continually adapts to the new domains without forgetting as the BN-statistics stay anchored to the source statistics and model parameters remain frozen. Additionally, our approach continually adapts to each new instance with negligible difference in latency.

B.3. Qualitative Assessment

We show qualitative results on the AviST [45] benchmark, comparing S-Tiny with other trackers in Figure 5. S-Tiny shows remarkable resilience against adverse visibility conditions. We also show qualitative results regarding the ablation assessment of the FMF layer compared to the baseline and the naive concatenation of features in Figure 6, visually showing the significance of our FMF layer.

C. Potential Negative Societal Impact

Our goal is mainly to advance the tracking performance of efficient visual tracking in-the-wild. On the other hand, if misused or used with ill-intent, this technology could potentially raise privacy concerns, especially if used for surveillance purposes, or other illegal activities like for stalking, bullying, etc. However, these are general issues that may arise even with other trackers that have been developed in the past. Our hope is to advance technology, and with regulations on its applications, we believe it should be possible to prevent its misuse.

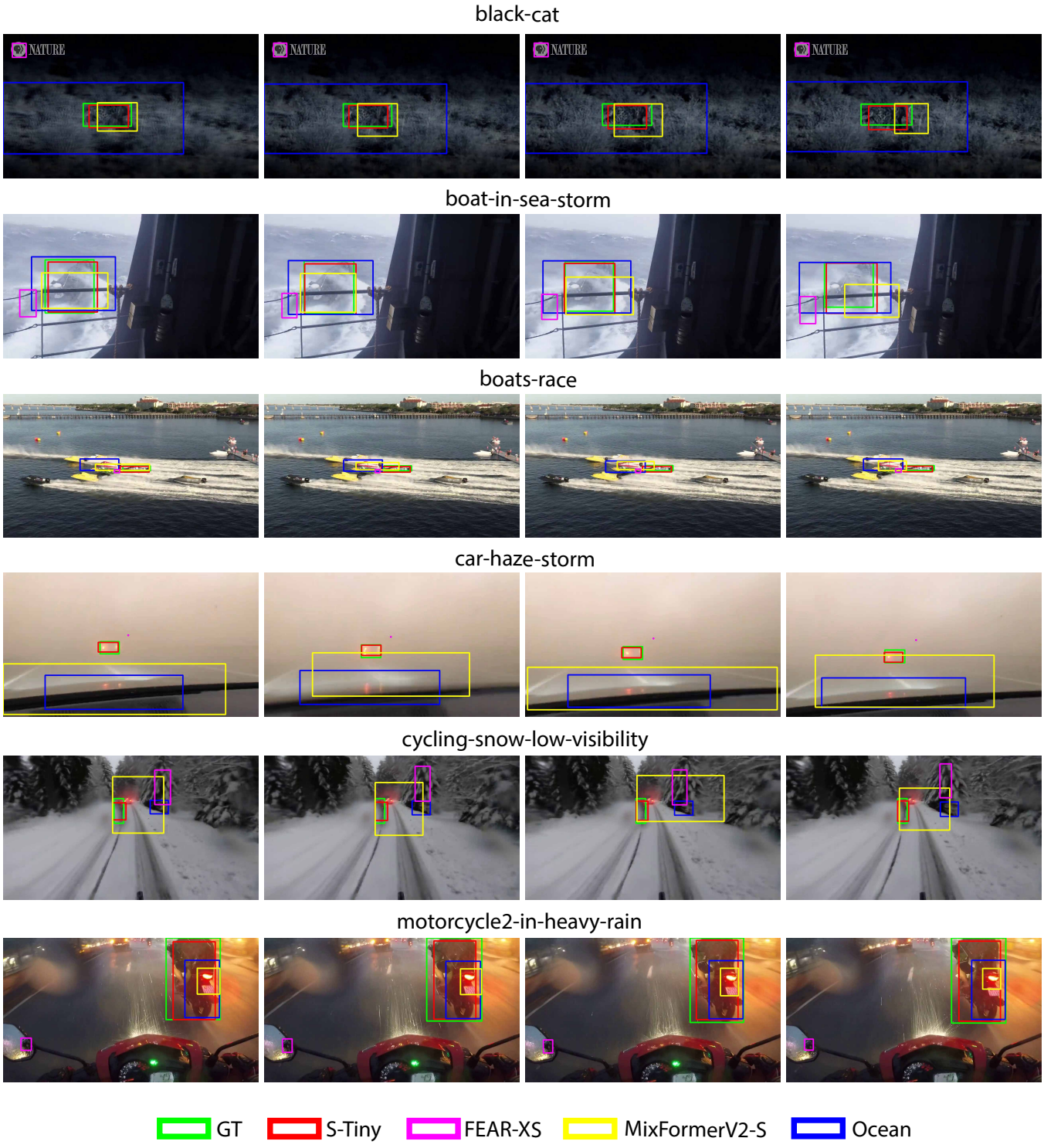

Figure 5. Qualitative comparison on the AVisT [45] dataset with other efficient trackers, and with the further inclusion of Ocean. Under adverse visibility conditions, our tracker, S-Tiny, is relatively stable compared to the others while running at 100 FPS on a CPU.

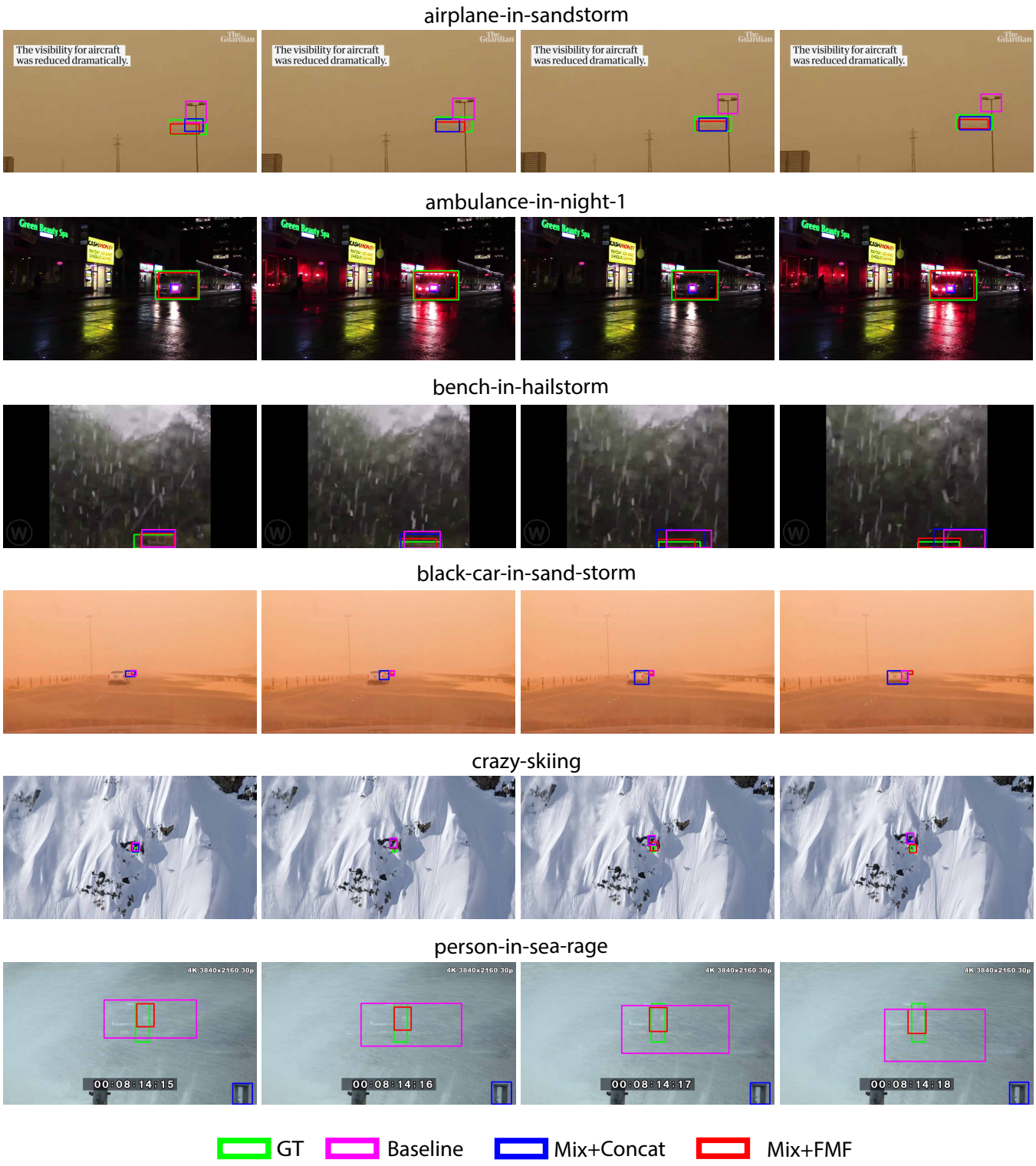

Figure 6. Qualitative results on the ablation study of the FMF layer on the AVisT [45] dataset. We notice the higher stability of our tracker when using FMF in terms of overalps with the ground truth (green), in comparison to the baseline and the naive concatenation.