APSIPA Transactions on Signal and Information Processing, 2022, 11, e27

Original Paper

UHP-SOT++: An Unsupervised Lightweight Single Object Tracker

Zhiruo Zhou[1]*, Hongyu Fu[1], Suya You[2] and C.-C. Jay Kuo[1]

[1] *University of Southern California, Los Angeles, CA, USA*
[2] *DEVCOM Army Research Laboratory, Adelphi, MD, USA*

ABSTRACT

An enhanced version of UHP-SOT called UHP-SOT++ is proposed for unsupervised, lightweight and high-performance single object tracking in this work. Both UHP-SOT and UHP-SOT++ exploit the discriminative-correlation-filters-based (DCF-based) tracker as their baseline and incorporate two new ingredients: (1) background motion modeling and (2) object box trajectory modeling. Their difference lies in the fusion strategy of proposals from three models (i.e., DCF, background motion and object box trajectory models). An improved fusion strategy is adopted by UHP-SOT++ for robust tracking performance against large-scale tracking datasets. Extensive evaluation of state-of-the-art supervised/unsupervised deep and unsupervised lightweight trackers is conducted on four SOT benchmark datasets – OTB2015, TC128, UAV123 and LaSOT. UHP-SOT++ achieves outstanding tracking performance with a small model size and low computational complexity (i.e., operating at a rate of 20 FPS on an i5 CPU even without code optimization). UHP-SOT++ offers an ideal solution in real-time object tracking on resource-limited platforms. Finally, we compare the pros and cons of supervised deep trackers and unsupervised lightweight trackers and provide a new perspective to their performance gap.

Keywords: Object tracking, online tracking, single object tracking, unsupervised tracking.

*Corresponding author: Zhiruo Zhou, zhiruozh@usc.edu.

Received 9 May 2022; Revised 17 July 2022
ISSN 2048-7703; DOI 10.1561/116.00000008

1 Introduction

Video object tracking is one of the fundamental computer vision problems. It finds rich applications in video surveillance [49], autonomous navigation [21], robotics vision [52], etc. Given a bounding box on the target object at the first frame, a tracker has to predict object box locations and sizes for all remaining frames in online single object tracking (SOT) [51]. The performance of a tracker is measured by accuracy (higher success rate), robustness (automatic recovery from tracking loss), computational complexity and speed (a higher number of frames per second of FPS).

Online trackers can be categorized into supervised and unsupervised ones [16]. Supervised/unsupervised trackers based on deep learning, called deep trackers, dominate the SOT field in recent years. The great majority of deep trackers are supervised one. Yet, there are modern unsupervised deep trackers trained by a large amount of data offline developed recently. Examples include ULAST [36], USOT [53], UDT+ [42], LUDT [43] and ResPUL [47].

Deep trackers often use a pre-trained network such as AlexNet [22] or VGG [7] as the feature extractor and do online tracking with extracted deep features [5, 9, 13, 29, 34, 38, 44]. Others adopt an end-to-end optimized model trained by video datasets in an offline manner [23, 24] and could be adapted to video frames in an online fashion [28, 32, 33, 37]. The tracking problem is formulated as a template matching problem in Siamese trackers [4, 18, 23, 24, 39, 46, 55], which is popular because of its simplicity and effectiveness. One recent trend is to apply the Vision Transformer in visual tracking [8, 45].

Although deep trackers offer state-of-the-art tracking accuracy, they do have some limitations. First, they demand large memory space to store the parameters of deep networks due to large model sizes. Second, the high computational power requirement hinders their applications in resource-limited devices such as drones or mobile phones. Third, deep trackers need to be trained with video samples of diverse content. Their capability in handling unseen objects appears to be limited, which will be illustrated in the experimental section. Last, for supervised deep trackers, a large number of annotated tracking video clips are needed in the training, which is a laborious and costly task. In contrast with deep trackers, unsupervised lightweight trackers are attractive in real-time tracking on resource-limited devices because of lower power consumption.

Unsupervised lightweight SOT methods often use discriminative correlation filters (DCFs). They were investigated between 2010 and 2020 [3, 6, 10, 12, 13, 20, 25, 26, 40, 50]. DCF trackers conduct dense sampling around the object box and solve a regression problem to learn a template for similarity matching. Under the periodic sample assumption, matching can be conducted very fast in the Fourier domain. Spatial-temporal regularized correlation filters (STRCF)

[25] adds spatial-temporal regularization to template update and performs favorably against other DCF trackers [9, 11].

An unsupervised lightweight tracker, called UHP-SOT (Unsupervised High-Performance Single Object Tracker), was recently proposed in [54] to address the issues. UHP-SOT used STRCF as the baseline and incorporated two new modules – background motion modeling and trajectory-based object box prediction. A simple fusion rule was adopted by UHP-SOT to integrate proposals from three modules into the final one. UHP-SOT has the potential to recover from tracking loss and offer flexibility in object box adaptation. UHP-SOT outperforms previous unsupervised single object trackers and narrows down the gap between unsupervised and supervised trackers. It achieves comparable performance against deep trackers on small-scale datasets such as TB-50 and TB-100 (or OTB 2015) [48].

This work is an extension of UHP-SOT with new contributions. First, the fusion strategies in UHP-SOT and UHP-SOT++ are different. The fusion strategy in UHP-SOT was simple and ad hoc. UHP-SOT++ adopts a fusion strategy that is more systematic and well justified. It is applicable to both small- and large-scale datasets with more robust and accurate performance. Second, this work conducts more extensive experiments on four object tracking benchmarks (i.e., OTB2015, TC128, UAV123 and LaSOT) while only experimental results on OTB2015 were reported for UHP-SOT in [54]. New experimental evaluations demonstrate that UHP-SOT++ outperforms previous unsupervised SOT methods (including UHP-SOT) and achieves comparable results with deep trackers on large-scale datasets. Since UHP-SOT++ has an extremely small model size, high tracking performance, and low computational complexity (operating at a rate of 20 FPS on an i5 CPU even without code optimization), it is ideal for real-time object tracking on resource-limited platforms. Finally, we compare pros and cons of SiamRPN++ and UHP-SOT++ trackers, which serve as an example of the supervised deep tracker and the unsupervised lightweight tracker, respectively, and provide a new perspective to their performance gap.

The rest of this paper is organized as follows. Related work is reviewed in Section 2. The UHP-SOT++ method is detailed in Section 3. Experimental results are shown in Section 4. Further discussion is provided in Section 5. Concluding remarks are given in Section 6.

2 Related Work

2.1 Visual Tracking

One popular class of unsupervised visual trackers conducts template matching within a search region to generate the response map for object location. The matched template in the next frame is centered at the location that has the highest response. A representative tracker of this class is the DCF tracker. In DCF, the template size is the same as that of the search region so that the Fast Fourier Transform (FFT) could be used to speed up the correlation process. To learn the template, a DCF uses the initial object patch to obtain a linear template via regression in the Fourier domain:

$$\arg\min_{\mathbf{f}} \frac{1}{2} \| \sum_{d=1}^{D} \mathbf{x}^d * \mathbf{f}^d - \mathbf{y} \|^2, \tag{1}$$

where $\mathbf{f}$ is the template to be determined, $\mathbf{x} \in \mathbb{R}^{N_x \times N_y \times D}$ is the spatial map of D features extracted from the object patch, $*$ is the feature-wise spatial convolution, and $\mathbf{y} \in \mathbb{R}^{N_x \times N_y}$ is a centered Gaussian-shaped map that serves as the regression label. Templates in DCFs tend to contain some background information. Furthermore, there exists boundary distortion caused by the 2D Fourier transform. To alleviate these side effects, it is often to weigh the template with a window function to suppress background and image discontinuity.

Unsupervised and supervised deep trackers often adopt the Siamese network as their backbone. The two branches of the Siamese network conduct feature extraction for the template and the search region, respectively. Usually, the object patch of the first frame serves as the template for search in all later frames. Then, the two feature maps are passed to a convolutional layer for correlation calculation to locate the object. The shape and size of the predicted object bounding box are determined by the regional proposal network inside the Siamese network.

2.2 Unsupervised Deep Trackers

There is an increasing interest in learning deep trackers from offline videos without annotations [41]. For example, UDT+ [42] and LUDT [43] investigated cycle learning in video, in which networks are trained to track forward and backward with consistent object proposals. ResPUL [47] mined positive and negative samples from unlabeled videos and leveraged them for supervised learning in building spatial and temporal correspondence. These unsupervised deep trackers reveal a promising direction in exploiting offline videos without annotations. Yet, they are limited in performance. Furthermore, they need the pre-training effort. In contrast, no pre-training on offline datasets is needed in our unsupervised tracker. Recently, an effective data sampling strategy, which samples moving objects in offline training using optical flow and dynamic programming, was adopted by USOT in [53]. The motion cue was leveraged by USOT for object tracking. The difference between USOT and UHP-SOT/UHP-SOT++ is that USOT uses motion to mine samples offline with dense optical flow while we focus on online object tracking with lightweight motion processing. The recent state-of-the-art ULAST [36] improves the cycle training process by further exploiting intermediate training frames and selecting better features and pseudo labels. Yet, it does not target at lightweight applications and still needs heavy backbones and pre-training.

2.3 Spatial-Temporal Regularized Correlation Filters

STRCF is a DCF-based tracker. It has an improved regression objective function using spatial-temporal regularization. The template is initialized at the first frame. Suppose that the object appearance at frame t is modeled by a template, denoted by $\mathbf{f}_t$, which will be used for similarity matching at frame $(t+1)$. By modifying Equation (1), STRCF updates its template at frame t by solving the following regression equation:

$$\arg\min_{\mathbf{f}} \left\{ \frac{1}{2} \| \sum_{d=1}^{D} \mathbf{x}_t^d * \mathbf{f}^d - \mathbf{y} \|^2 + \frac{1}{2} \sum_{d=1}^{D} \| \mathbf{w} \cdot \mathbf{f}^d \|^2 + \frac{\mu}{2} \| \mathbf{f} - \mathbf{f}_{t-1} \|^2 \right\}, \qquad (2)$$

where $\mathbf{w}$ is the spatial weight on the template, $\mathbf{f}_{t-1}$ is the template obtained from time $t-1$, and μ is a constant regularization coefficient. We can interpret the three terms in Equation (2) as follows. The first term is the standard regression objective function of a DCF. The second term imposes the spatial regularization. It gives more weights to features in the center region of a template in the matching process. The third term imposes temporal regularization for smooth appearance change.

To search for the box in frame $(t+1)$, STRCF correlates template $\mathbf{f}_t$ with the search region and determines the new box location by finding the location that gives the highest response. Although STRCF can model the appearance change for general sequences, it suffers from overfitting. That is, it is not able to adapt to largely deformed objects quickly. Furthermore, it cannot recover from tracking loss. The template model, $\mathbf{f}$, is updated at every frame with a fixed regularization coefficient, μ, in standard STRCF.

There is a performance gap between supervised/unsupervised deep trackers and unsupervised DCF trackers. It is attributed to the limitations of DCF trackers such as failure to recover from tracking loss and inflexibility in object box adaptation. Our UHP-SOT++ adopts STRCF as a building module. To address the above-mentioned shortcomings, we have some modification in our implementation. First, we skip updating $\mathbf{f}$ if no obvious motion is observed. Second, a smaller μ is used when all modules agree with each other in prediction so that $\mathbf{f}$ can adapt to the new appearance of largely deformed objects faster.

3 Proposed UHP-SOT++ Method

3.1 System Overview

There are three main challenges in SOT:

1. significant change of object appearance,
2. loss of tracking,
3. rapid variation of object's location and/or shape.

We propose a new tracker, UHP-SOT++, to address these challenges, As shown in Figure 1, it consists of three modules:

1. appearance model update,
2. background motion modeling,
3. trajectory-based box prediction.

UHP-SOT++ follows the classic tracking-by-detection paradigm where the object is detected within a region centered at its last predicted location at each frame. The histogram of oriented gradients (HOG) features as well as the color names (CN) [14] features are extracted to yield the feature map. We choose the STRCF tracker [25] as the baseline because of its efficient and effective appearance modeling and update. Yet, STRCF cannot handle the second and the third challenges well because it only focuses on the modeling of object appearance which could vary a lot across different frames. Generally, the high variety of object appearance is difficult to capture using a single model. Thus, we propose the second and the third modules in UHP-SOT++ to enhance its tracking accuracy. UHP-SOT++ operates in the following fashion. The

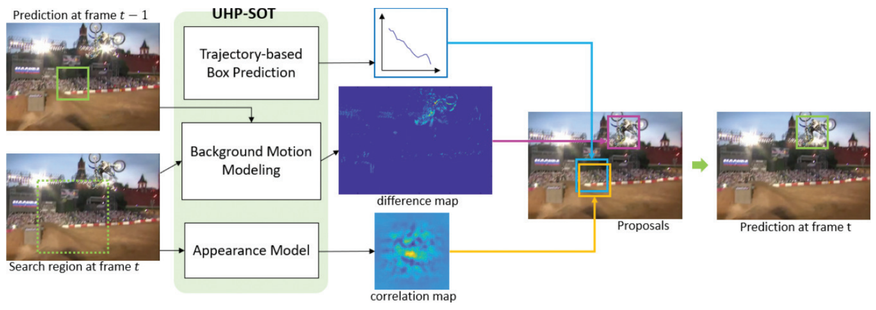
Prediction at frame $t-1$
UHP-SOT
Trajectory-based Box Prediction
Background Motion Modeling
Appearance Model
difference map
correlation map
Search region at frame t
Proposals
Prediction at frame t

Figure 1: The system diagram of the proposed UHP-SOT++ method. It shows one example where the object was lost at time $t-1$ but gets retrieved at time t because the proposal from background motion modeling is accepted.

baseline tracker gets initialized at the first frame. For the following frames, UHP-SOT++ gets proposals from all three modules and merges them into the final prediction based on a fusion strategy.

The STRCF tracker was already discussed in Section 2.3. For the rest of this section, we examine the background motion modeling module and the trajectory-based box prediction module in UHP-SOT in Sections 3.2 and 3.3, respectively. Finally, we will elaborate on the fusion strategy in Section 3.4. Note that the fusion strategies of UHP-SOT and UHP-SOT++ are completely different.

3.2 Background Motion Modeling

We decompose the pixel displacement between adjacent frames (also called optical flow) into two types: object motion and background motion. Background motion is usually simpler, and it may be fit by a parametric model. Background motion estimation [1, 17] finds applications in video stabilization, coding and visual tracking. Here, we propose a 6-parameter model in form of

$$x_{t+1} = \alpha_1 x_t + \alpha_2 y_t + \alpha_0, \tag{3}$$

$$y_{t+1} = \beta_1 x_t + \beta_2 y_t + \beta_0, \tag{4}$$

where (x_{t+1}, y_{t+1}) and (x_t, y_t) are corresponding background points in frames $(t+1)$ and t, respectively, and α_i and β_i, $i = 0, 1, 2$ are model parameters. With more than three pairs of corresponding points, we can determine the model parameters using the linear least-squares method. Usually, we choose a few salient points (e.g., corners) to build the correspondence. We apply the background model to the grayscale image $I_t(x,y)$ of frame t to find the estimated $\hat{I}_{t+1}(x,y)$ of frame $(t+1)$. Then, we can compute the difference map ΔI:

$$\Delta I = \hat{I}_{t+1}(x,y) - I_{t+1}(x,y), \tag{5}$$

which is expected to have small and large absolute values in the background and foreground regions, respectively. Thus, we can determine potential object locations.

While DCF trackers exploit foreground correlation to locate the object, background modeling uses background correlation to eliminate background influence in object tracking. They complement each other. DCF trackers cannot recover from tracking loss easily since it does not have a global view of the scene. In contrast, our background modeling can find potential object locations by removing background.

3.3 Trajectory-based Box Prediction

Given the predicted box centers of the object of the last N frames, $\{(x_{t-N}, y_{t-N}), \dots, (x_{t-1}, y_{t-1})\}$, we calculate $N-1$ displacement vectors $\{(\Delta x_{t-N+1}, \Delta y_{t-N+1}),$

$\ldots, (\Delta x_{t-1}, \Delta y_{t-1})\}$ and apply the principal component analysis (PCA) to them. To predict the displacement at frame t, we fit the first principal component using a line and set the second principal component to zero to remove noise. Then, the center location of the box at frame t can be written as

$$(\hat{x}_t, \hat{y}_t) = (x_{t-1}, y_{t-1}) + (\hat{\Delta x_t}, \hat{\Delta y_t}). \tag{6}$$

Similarly, we can estimate the width and the height of the box at frame t, denoted by $(\hat{w}_t, \hat{h}_t)$. Typically, the physical motion of an object has an inertia in motion trajectory and its size, and the box prediction process attempts to maintain the inertia. It contributes to better tracking performance in two ways. First, it removes small fluctuation of the box in its location and size. Second, when there is a rapid deformation of the target object, the appearance model alone cannot capture the shape change effectively. In contrast, the combination of background motion modeling and the trajectory-based box prediction can offer a more satisfactory solution. For example, Figure 2, shows a frame of the *diving* sequence in the upper-left subfigure, where the green and the magenta boxes are the ground truth and the result of UHP-SOT++, respectively. Although a DCF tracker can detect the size change by comparing correlation scores at five image resolutions, it cannot estimate the aspect ratio change properly. In contrast, as shown in the lower-left subfigure, the

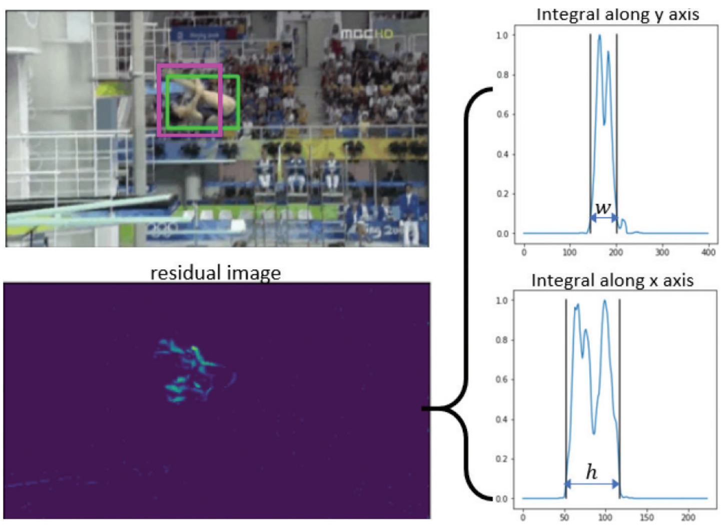

Figure 2: Illustration of shape change estimation based on background motion model and trajectory-based box prediction, where the ground truth and our proposal are annotated in green and magenta, respectively.

residual image after background removal in UHP-SOT++ reveals the object shape. By summing up absolute pixel values of the residual image horizontally and vertically and using a threshold to determine two ends of the box, we have

$$\hat{w} = x_{\max} - x_{\min}, \quad \text{and} \quad \hat{h} = y_{\max} - y_{\min}. \tag{7}$$

Note that raw estimates may not be stable across different frames. Estimates that deviate much from the trajectory of $(\Delta w_t, \Delta h_t)$ are rejected to yield a robust and deformable box proposal.

3.4 Fusion Strategy

We have three box proposals for the target object at frame t: (1) B_{app} from the baseline STRCF tracker to capture appearance change, (2) B_{bgd} from the background motion predictor to eliminate unlikely object regions, and (3) B_{trj} from the trajectory predictor to maintain the inertia of the box position and size. A fusion strategy is needed to yield the final box location and size. We consider a couple of factors for its design.

3.4.1 Proposal Quality

There are three box proposals. The quality of each box proposal can be measured by: (1) object appearance similarity, and (2) robustness against the trajectory. We use a binary flag to indicate whether the quality of a proposal is good or not. As shown in Table 1, the flag is set to one if a proposal keeps proper appearance similarity and is robust against trajectory. Otherwise, it is set to zero.

For the first measure, we store two appearance models: the latest model, $\mathbf{f}_{t-1}$, and an older model, $\mathbf{f}_i$, $i \leq t-1$, where i is the last time instance where all three boxes have the same location. Model $\mathbf{f}_i$ is less likely to be contaminated since it needs agreement from all modules. To check the reliability of the three proposals, we compute correlation scores for the following six pairs: ($\mathbf{f}_{t-1}$, B_{app}), ($\mathbf{f}_{t-1}$, B_{trj}), ($\mathbf{f}_{t-1}$, B_{bgd}), ($\mathbf{f}_i$, B_{app}), ($\mathbf{f}_i$, B_{trj}), and ($\mathbf{f}_i$, B_{bgd}). They provide appearance similarity measures of the two previous models against the current three proposals. A proposal has good similarity if one of its correlation scores is higher than a threshold.

For the second measure, if B_{app} and B_{trj} have a small displacement (say, 30 pixels) from the last prediction, the move is robust. As to B_{bgd}, it often jumps around and, thus, is less reliable. However, if the standard deviations of its historical locations along the x-axis and y-axis are small enough (e.g., 30 pixels over the past 10 frames), then they are reliable.

3.4.2 Occlusion Detection

We propose an occlusion detection strategy for color images, which is illustrated in Figure 3. As occlusion occurs, we often observe a sudden drop in the similarity score and a rapid change on the averaged RGB color values inside the box. A drop is sudden if the mean over the past several frames is high while the current value is significantly lower. If this is detected, we keep the new prediction the same as the last predicted position since the new prediction is unreliable. We do not update the model for this frame either to avoid drifting and/or contamination of the appearance model.

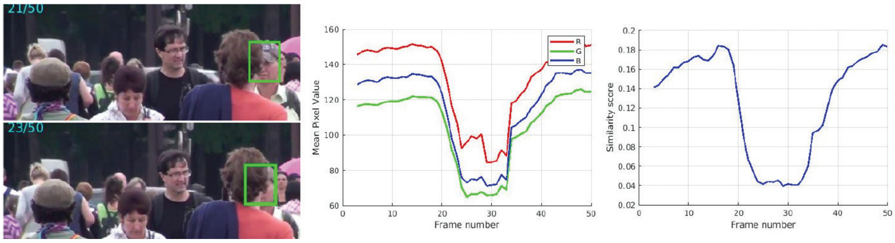

Figure 3: Illustration of occlusion detection, where the green box shows the object location. The color information and similarity score could change rapidly if occlusion occurs.

3.4.3 Rule-based Fusion

Since each of the three proposals has a binary flag, all tracking scenarios can be categorized into 8 cases as shown in Figure 4. We propose a fusion scheme for each case below.

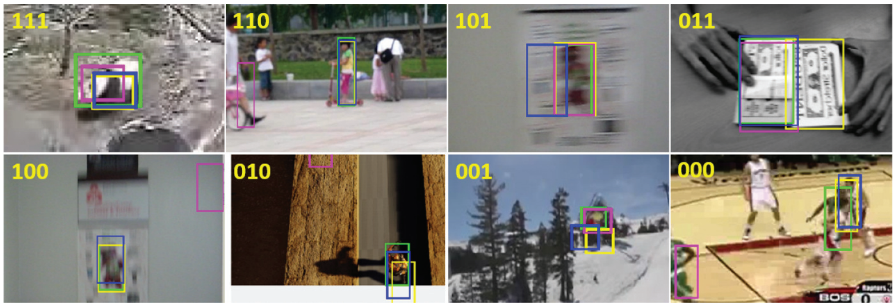

Figure 4: An example of quality assessment of proposals, where the green box is the ground truth, and yellow, blue and magenta boxes are proposals from B_{app}, B_{trj} and B_{bgd}, respectively, and the bright yellow text on the top-left corner denotes the quality of three proposals $(isGood_{app}, isGood_{trj}, isGood_{bgd})$.

- When all three proposals are good, their boxes are merged together as a minimum covering rectangle if they overlap with each other with IoU above a threshold. Otherwise, B_{app} is adopted.
- When two proposals are good, merge them if they overlap with each other with IoU above a threshold. Otherwise, the one with better robustness is adopted.
- When one proposal is good, adopt that one if it is B_{app}. Otherwise, that proposal is compared with B_{app} to verify its superiority by observing a higher similarity score or better robustness.
- When all proposals have poor quality, the occlusion detection process is conducted. The last prediction is adopted in case of occlusion. Otherwise, B_{app} is adopted.
- When other proposals outperform B_{app}, the regularization coefficient, μ, is adjusted accordingly for stronger update. Because this might reveal that the appearance model needs to be updated more to capture the new appearance.

The fusion rule is summarized in Table 1. In most cases, B_{app} is reliable and it will be chosen or merged with other proposals because the change is smooth between adjacent frames in the great majority of frames in a video clip.

Table 1: All tracking scenarios are classified into 8 cases in terms of the overall quality of proposals from three modules. The fusion strategy is set up for each scenario. The update rate is related the regularization coefficient, μ, that controls to which extent the appearance model should be updated.

$isGood_{app}$	$isGood_{trj}$	$isGood_{bgd}$	Proposal to take	Update rate
1	1	1	B_{app} or union of three	Normal
1	1	0	B_{app} or B_{trj} or union of two	Normal
1	0	1	B_{app} or B_{bgd} or union of two	Normal
0	1	1	B_{trj} or B_{bgd} or union of two	Normal or stronger
1	0	0	B_{app}	Normal
0	1	0	B_{app} or B_{trj}	Normal or stronger
0	0	1	B_{app} or B_{bgd}	Normal or stronger
0	0	0	B_{app} or last prediction in case of occlusion	Normal or weaker

4 Experiments

4.1 Experimental Set-up

To show the performance of UHP-SOT++, we compare it with several state-of-the-art unsupervised and supervised trackers on four single object tracking datasets. They are OTB2015 [48], TC128 [27], UAV123 [31] and LaSOT [15]. OTB2015 (also named OTB in short) and TC128, which contain 100 and 128 color or grayscale video sequences, respectively, are two widely used small-scale datasets. UAV123 is a larger one, which has 123 video sequences with more than 110K frames in total. Videos in UAV123 are captured by low-altitude drones. They are useful in the tracking test of small objects with a rapid change of viewpoints. LaSOT is the largest single object tracking dataset that targets at diversified object classes and flexible motion trajectories in longer sequences. It has one training set with dense annotation for supervised trackers to learn and another test set for performance evaluation. The test set contains 280 videos of around 685K frames.

Performance evaluation is conducted using the "One Pass Evaluation (OPE)" protocol. The metrics include the precision plot (i.e., the distance of the predicted and actual box centers) and the success plot (i.e., overlapping ratios at various thresholds). The distance precision (DP) is measured at the 20-pixel threshold to rank different methods. The overlap precision is measured by the area-under-curve (AUC) score. We use the same hyperparameters as those in STRCF except for regularization coefficient, μ. If the appearance box is not chosen, STRCF sets $\mu = 15$ while UHP-SOT++ selects $\mu \in \{15, 10, 5, 0\}$. The smaller μ is, the stronger the update is. The number of previous frames for trajectory prediction is $N = 20$. The cutting threshold along the horizontal or vertical direction is set 0.1. The threshold for good similarity score is 0.08, and a threshold of 0.5 for IoU is adopted. UHP-SOT++ runs at 20 frames per second (FPS) on a PC equipped with an Intel(R) Core(TM) i5-9400F CPU. The speed data of other trackers are either from their original papers or benchmarks. Since no code optimization is conducted, all reported speed data should be viewed as lower bounds for the corresponding trackers.

4.2 Ablation Study

We compare different configurations of UHP-SOT++ on the TC128 dataset to investigate contributions from each module in Figure 5. As compared with UHP-SOT, improvements on both DP and AUC in UHP-SOT++ come from the new fusion strategy. Under this strategy, the background motion modeling plays an more important role and it has comparable performance even without the trajectory prediction. Although the trajectory prediction module is simple, it contributes a lot to higher tracking accuracy and robustness as revealed by the performance improvement over the baseline STRCF.

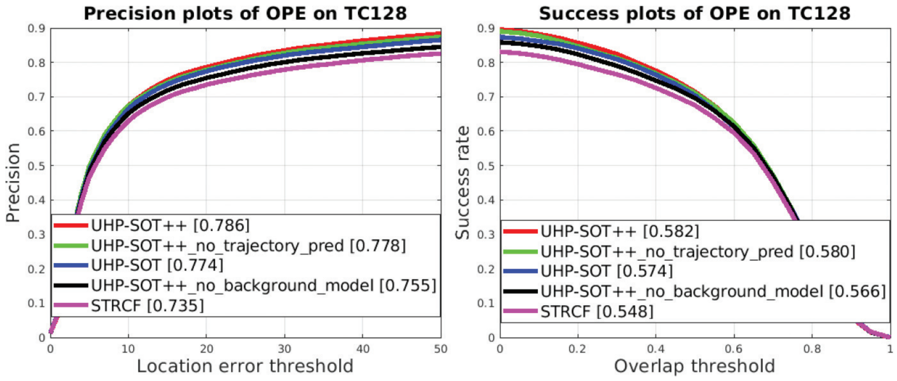

Figure 5: The precision plot and the success plot of our UHP-SOT++ tracker with different configurations on the TC128 dataset, where the numbers inside the parentheses are the DP values and AUC scores, respectively.

More performance comparison between UHP-SOT++, UHP-SOT and STRCF is presented in Table 2. As compared with STRCF, UHP-SOT++ achieves 1.8%, 6.2%, 6.7%, and 6.8% gains in the success rate on OTB, TC128, UAV123 and LaSOT, respectively. As to the mean precision, it has an improvement of 1.2%, 6.9%, 7.2%, and 10.4%, respectively. Except for OTB, UHP-SOT++ outperforms UHP-SOT in both the success rate and the precision. This is especially obvious for large-scale datasets. Generally,

Table 2: Comparison of state-of-the-art supervised and unsupervised trackers on four datasets, where the performance is measured by the distance precision (DP) and the area-under-curve (AUC) score in percentage. The model size is measured in MB by the memory required to store needed data such as the model parameters of pre-trained networks. The best unsupervised performance is highlighted. Also, S, P, G and C indicate **S**upervised, **P**re-trained, **G**PU and **C**PU, respectively.

				OTB2015		TC128		UAV123		LaSOT			Model
Trackers	Year	**S**	**P**	DP	AUC	DP	AUC	DP	AUC	DP	AUC	FPS	size
SiamRPN++[23]	2019	✓	✓	91.0	69.2	-	-	84.0	64.2	49.3	49.5	35 (G)	206
ECO[9]	2017	✓	✓	90.0	68.6	80.0	59.7	74.1	52.5	30.1	32.4	10 (G)	329
UDT+[42]	2019	×	✓	83.1	63.2	71.7	54.1	-	-	-	-	55 (G)	< 1
LUDT[43]	2020	×	✓	76.9	60.2	67.1	51.5	-	-	-	26.2	70 (G)	< 1
ResPUL[47]	2021	×	✓	-	58.4	-	-	-	-	-	-	- (G)	> 6
USOT[53]	2021	×	✓	80.6	58.9	-	-	-	-	32.3	33.7	- (G)	113
ULAST[36]	2022	×	✓	81.1	61.0	-	-	-	-	**40.7**	**43.3**	80 (G)	-
ECO-HC[9]	2017	×	×	85.0	63.8	75.3	55.1	72.5	50.6	27.9	30.4	42 (C)	< 1
STRCF[25]	2018	×	×	86.6	65.8	73.5	54.8	67.8	47.8	29.8	30.8	24 (C)	< 1
UHP-SOT[54]	2021	×	×	**90.9**	**68.9**	77.4	57.4	71.0	50.1	31.1	32.0	23 (C)	< 1
UHP-SOT++	Ours	×	×	87.6	66.9	**78.6**	**58.2**	**72.7**	**51.0**	32.9	32.9	20 (C)	< 1

UHP-SOT++ has better tracking capability than UHP-SOT. Its performance drop in OTB is due to the tracking loss in three sequences; namely, *Bird2*, *Coupon* and *Freeman4*. They have multiple complicated appearance changes such as severe rotation, background clutter and heavy occlusion. As shown in Figure 6, errors at some key frames lead to total loss of the object, and the lost object cannot be easily recovered from motion. The trivial fusion strategy based on appearance similarity in UHP-SOT seems to work well on their key frames while the fusion strategy of UHP-SOT++ does not suppress wrong proposals properly since background clutters have stable motion and trajectories as well.

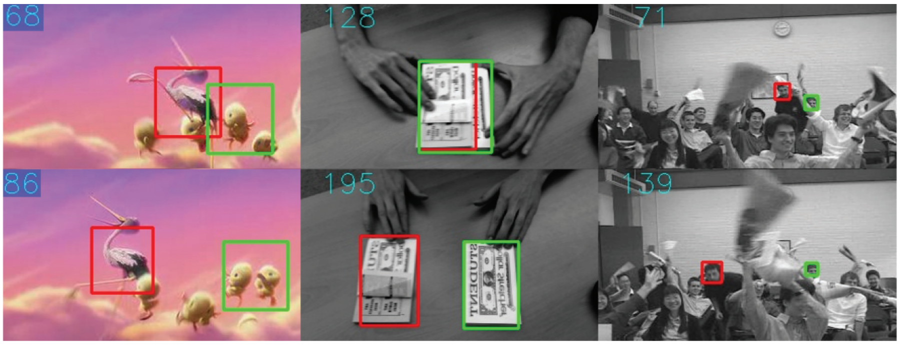

Figure 6: Failure cases of UHP-SOT++ (in green) as compared to UHP-SOT (in red) on OTB2015.

4.3 Comparison with State-of-the-art Trackers

We compare the performance of UHP-SOT++ and several unsupervised trackers for the LaSOT dataset in Figure 7. The list of benchmarking methods includes: USOT [53], ECO-HC [9], STRCF [25], CSR-DCF [2], SRDCF [11], Staple [3], KCF [20], DSST [12]. UHP-SOT++ achieves comparable performance with the state-of-the-art deep unsupervised USOT that has the ResNet-50 [19] backbone network and large-scale offline training. UHP-SOT++ outperforms DCF-based unsupervised methods by a large margin, which is larger than 0.02 in the mean scores of the success rate and the precision. Besides, its running speed is 20 FPS, which is comparable with that of the third runner STRCF (24 FPS) and the fourth runner ECO-HC (42 FPS). With a small increase in computational and memory resources, UHP-SOT++ gains in tracking performance by adding object box trajectory and background motion modeling modules. Object boxes of three leading DCF-based unsupervised trackers are visualized in Figure 8 for qualitative performance comparison. More comparison with other unsupervised methods including LADCF [50],

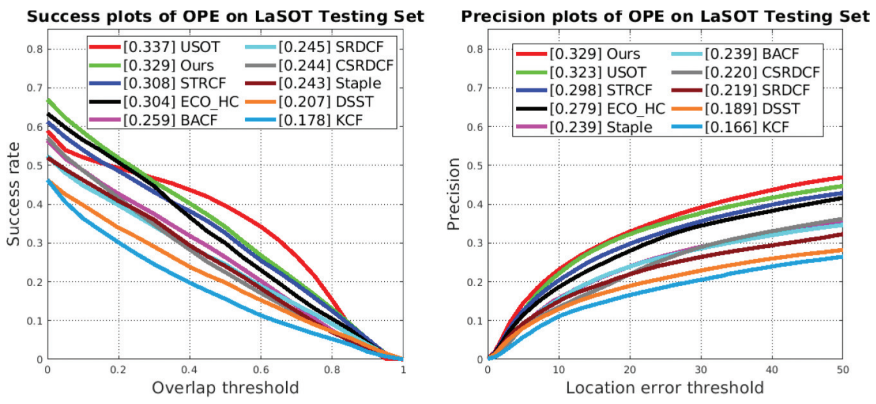

Figure 7: The success plot and the precision plot of ten unsupervised tracking methods for the LaSOT dataset, where the numbers inside the parentheses are the overlap precision and the distance precision values, respectively.

UDT [42], UDT+ [42] over other benchmark datasets is shown in Figure 9. As compared with other methods, UHP-SOT++ offers more robust and flexible box prediction. They follow tightly with the object in both location and shape even under challenging scenarios such as motion blur and rapid shape change.

We compare the success rates of UHP-SOT++ and several supervised and unsupervised trackers against all four datasets in Figure 9. Note that there are more benchmarking methods for OTB but fewer for TC128, UA123 and LaSOT since OTB is an earlier dataset. The supervised deep trackers under consideration include SiamRPN++ [23], ECO [9], C-COT [13], DeepSRDCF [11], HDT [16], SiamFC_3s [4], CFNet [40], and LCT [30]. Other deep trackers that have leading performance but are not likely to be used on resource-limited devices due to their extremely high complexity, such as transformer-based trackers [8, 45], are not included here. Although the performance of a tracker may vary from one dataset to the other due to different video sequences collected by each dataset, UHP-SOT++ is among the top runners in all four datasets. This demonstrates the generalization capability of UHP-SOT++. Its better performance than ECO on LaSOT indicates a robust and effective update of the object model. Otherwise, it would degrade quickly with worse performance because of longer LaSOT sequences. Besides, its tracking speed of 20 FPS on CPU is faster than many deep trackers such as ECO (10 FPS), DeepSRDCF (0.2 FPS), C-COT (0.8 FPS) and HDT (2.7 FPS).

In Table 2, we further compare UHP-SOT++ with state-of-the-art unsupervised deep trackers ULAST [36], USOT [53], UDT+ [42], LUDT [43], and ResPUL [47] in their AUC and DP values, running speeds and model sizes. Two leading supervised trackers SiamRPN++ and ECO are also included in Figure 9. We see that UHP-SOT++ has outstanding overall performance

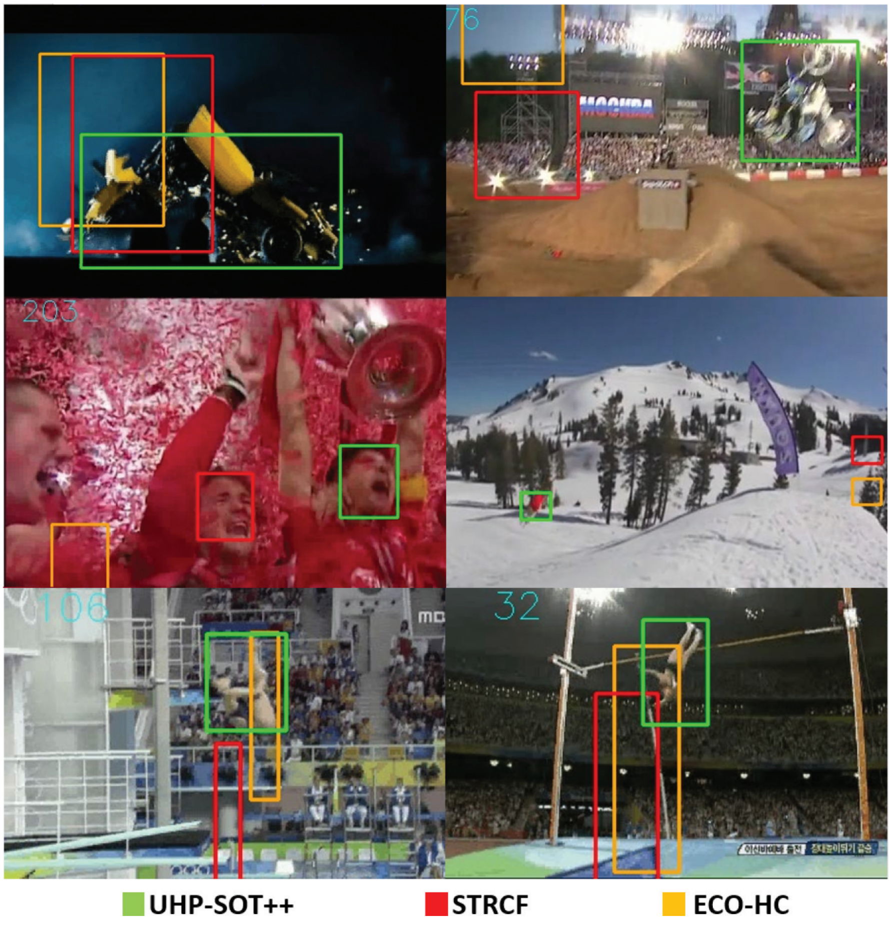

Figure 8: Qualitative evaluation of three leading unsupervised trackers, where UHP-SOT++ offers a robust and flexible box prediction.

against recent unsupervised deep trackers. UHP-SOT++ achieves comparable performance with USOT, which has very deep feature extraction network, on LaSOT and much better accuracy on OTB2015. It also outperforms other unsupervised deep trackers with shallow feature extraction backbones by a large margin. ULAST achieves high performance on LaSOT with a deep region proposal network as well as a carefully designed pre-training strategy. It demands a large amount of data in the pre-training of the large backbone model and the region proposal network.

It is worthwhile to emphasize that deep trackers demand pre-training on offline datasets while UHP-SOT++ does not. In addition, UHP-SOT++

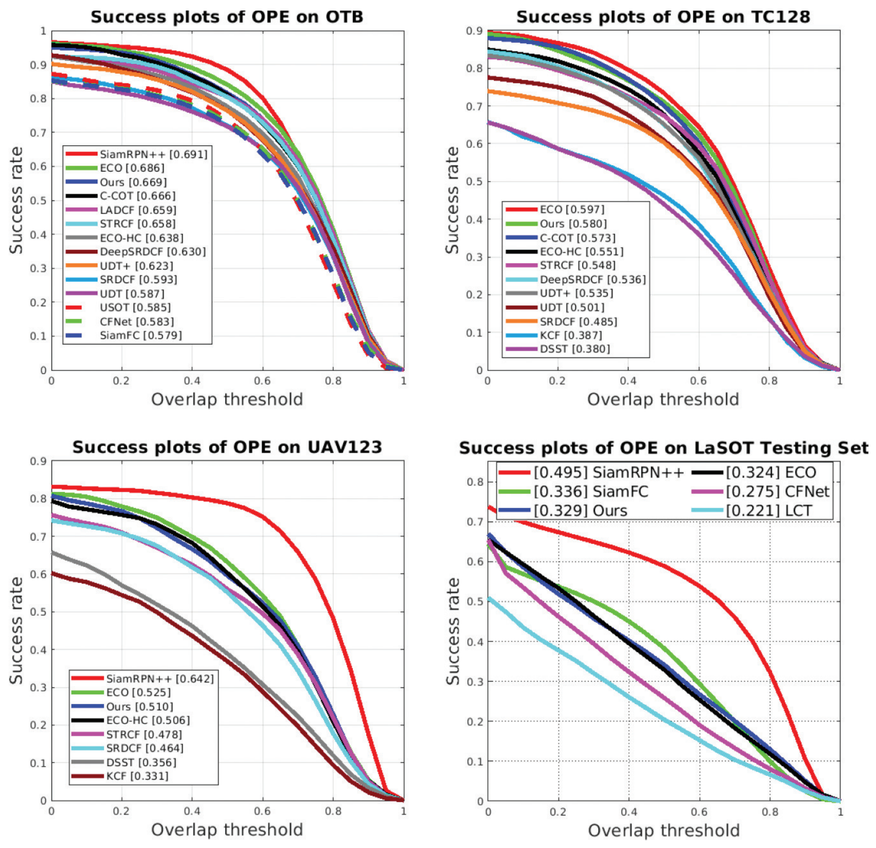

Figure 9: The success plot comparison of UHP-SOT++ with several supervised and unsupervised tracking methods on four datasets, where only trackers with raw results published by authors are listed. For the LaSOT dataset, only supervised trackers are included for performance benchmarking in the plot since the success plot of unsupervised methods is already given in Figure 7.

is attractive because of its lower memory requirement and near real-time running speed on CPUs. Although ECO-HC also provides a lightweight solution, there is a performance gap between UHP-SOT++ and ECO-HC. SiamRPN++ has the best tracking performance among all trackers, due to the merit of end-to-end optimized network with auxiliary modules such as classification head and the region proposal network. Yet, its large model size and GPU hardware requirement limit its applicability in resource-limited devices such as mobile phones or drones. In addition, as an end-to-end optimized deep tracker, SiamRPN++ has the interpretability issue to be discussed later.

4.4 Attribute-based Study

To better understand the capability of different trackers, we analyze the performance variation under various challenging tracking conditions. These conditions can be classified into the following attributes: aspect ratio change (ARC), background clutter (BC), camera motion (CM), deformation (DEF), fast motion (FM), full occlusion (FOC), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV), partial occlusion (POC), scale variation (SV) and viewpoint change (VC). We compare the AUC scores of supervised trackers (e.g., SiamRPN++ and ECO) and unsupervised trackers (e.g., UHP-SOT++, ECO-HC, UDT+, and STRCF) under these attributes in Figure 10.

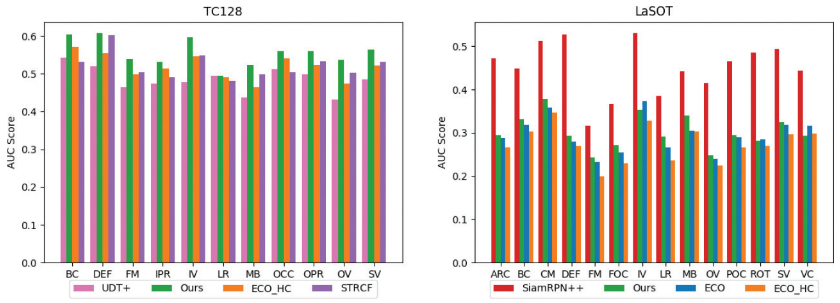

Figure 10: The area-under-curve (AUC) scores for two datasets, TC128 and LaSOT, under the attribute-based evaluation, where attributes of concern include the aspect ratio change (ARC), background clutter (BC), camera motion (CM), deformation (DEF), fast motion (FM), full occlusion (FOC), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV), partial occlusion (POC), scale variation (SV) and viewpoint change (VC), respectively.

We have the following observations. First, among unsupervised trackers, UHP-SOT++ has leading performance in all attributes, which reveals improved robustness from its basic modules and fusion strategy. Second, although ECO utilizes deep features, it is weak in flexible box regression and, as a result, it is outperformed by UHP-SOT++ in handling such deformation and shape changes against LaSOT. In contrast, SiamRPN++ is better than other trackers especially in DEF (deformation), ROT (rotation)and VC (viewpoint change). The superior performance of SiamRPN++ demonstrates the power of its region proposal network (RPN) in generating tight boxes. The RPN inside SiamRPN++ not only improves IoU score but also has the long-term benefit by excluding noisy information. Fourth, supervised trackers perform better in IV (illumination variation) and LR (low resolution) than unsupervised trackers in general. The gap between ECO and its handcrafted version, ECO-HC, is more obvious under these attributes. This can be explained by the fact that unsupervised trackers adopt HOG, CN features or other shallow features which do not work well under these attributes. They focus on local structures of the appearance and tend to fail to capture the object when the local gradient or color information is not stable. Finally, even with the feature limitations, UHP-SOT++ still runs second in many attributes against LaSOT because of the stability offered by trajectory prediction and its capability to recover from tracking loss via background motion modeling.

5 Exemplary Sequences and Qualitative Analysis

After providing quantitative results in Section 4, we conduct error analysis on a couple of representative sequences to gain more insights in this section. Several exemplary sequences from LaSOT are shown in Figure 11, in which SiamRPN++ performs either very well or quite poorly. In the first two sequences, we see the power of accurate box regression contributed by the RPN. In this type of sequence, good trackers can follow the object well. Yet, their poor bounding boxes lead to a low success score. Furthermore, the appearance model would be contaminated by the background information as shown in the second cat example. The appearance model of DCF-based methods learns background texture (rather than follows the cat) gradually. When the box only covers part of the object, it might also miss some object

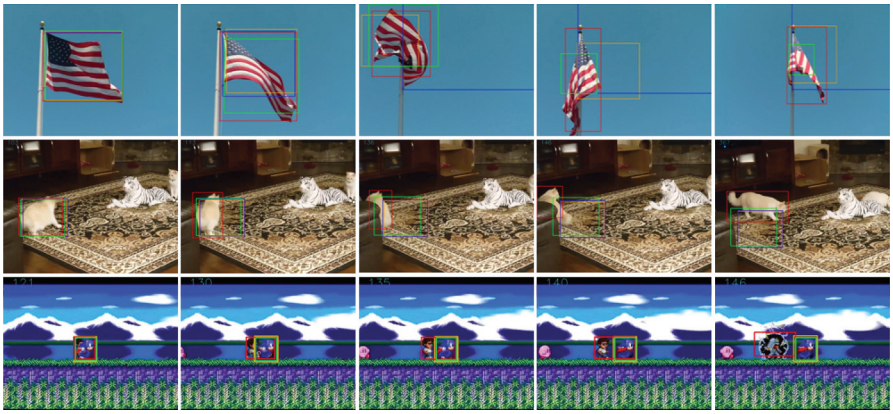
Figure 11: Qualitative comparison of top runners against the LaSOT dataset, where tracking boxes of SiamRPN++, UHP-SOT++, ECO and ECO-HC are shown in red, green, blue and yellow, respectively. The first two rows show sequences in which SiamRPN++ outperforms others significantly while the last row offers the sequence in which SiamRPN++ performs poorly.

features, resulting in a degraded appearance model. In both scenarios, the long-term performance will drop rapidly. Although UHP-SOT++ allows the aspect ratio change to some extent as seen in the first flag example, its residual map obtained by background motion modeling is still not as effective as the RPN due to lack of semantic meaning. Generally speaking, the performance of UHP-SOT++ relies on the quality of the appearance model and the residual map.

On the other hand, SiamRPN++ is not robust enough to handle a wide range of sequences well. The third example sequence is from video games. SiamRPN++ somehow includes background objects in its box proposals and drifts away from its targets in the presented frames. Actually, these background objects are different from their corresponding target objects in either semantic meaning or local information such as color or texture. The performance of the other three trackers is not affected. We see that they follow the ground truth without any problem. One explanation is that these video game sequences could be few in the training set and, as a result, SiamRPN++ cannot offer a reliable tracking result for them.

Finally, several sequences in which UHP-SOT++ has the top performance are shown in Figure 12. In the first cup sequence, all other benchmarking methods lose the target while UHP-SOT++ could go back to the object once the object has obvious motion in the scene. In the second bottle sequence, UHP-SOT++ successfully detects occlusion without making random guesses and the object box trajectory avoids the box to drift away. In contrast, other trackers make ambitious moves without considering the inertia of motion. The third bus sequence is a complicated one that involves several challenges such as full occlusion, scale change and aspect ratio change. UHP-SOT++ is the only one that can recover from tracking loss and provide flexible box predictions. These examples demonstrate the potential of UHP-SOT++ that exploits object and background motion clues across frames effectively.

6 Conclusion and Future Work

An unsupervised high-performance tracker, UHP-SOT++, was proposed in this paper. It incorporated two new modules in the STRCF tracker module. They were the background motion modeling module and the object box trajectory modeling module. Furthermore, a novel fusion strategy was adopted to combine proposals from all three modules systematically. It was shown by extensive experimental results on large-scale datasets that UHP-SOT++ can generate robust and flexible object bounding boxes and offer a real-time high-performance tracking solution on resource-limited platforms.

The pros and cons of supervised and unsupervised trackers were discussed. Unsupervised trackers such as UHP-SOT and UHP-SOT++ have the potential

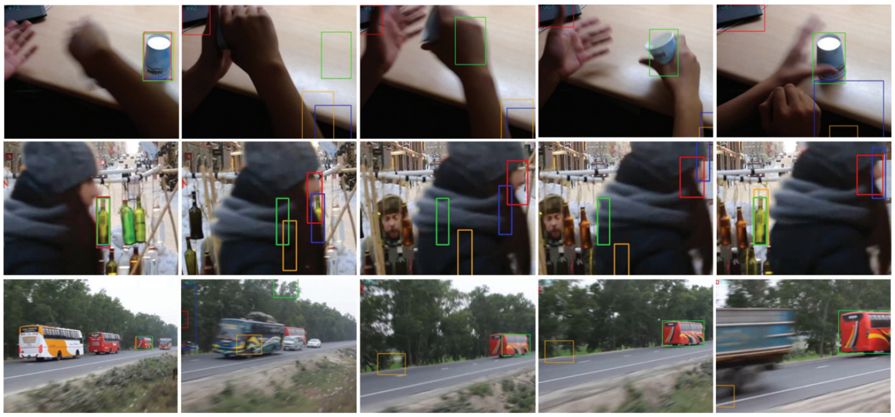
Figure 12: Illustration of three sequences in which UHP-SOT++ performs the best. The tracking boxes of SiamRPN++, UHP-SOT++, ECO and ECO-HC are shown in red, green, blue and yellow, respectively.

in delivering an explainable lightweight tracking solution while maintaining good performance in accuracy. Supervised trackers such as SiamRPN++ benefit from offline end-to-end learning and perform well in general. However, they need to run on GPUs, which is too costly for mobile and edge devices. They may encounter problems in rare samples. Extensive supervision with annotated object boxes is costly. Lack of interpretability could be a barrier for further performance boosting.

Although UHP-SOT++ offers a state-of-the-art unsupervised tracking solution, there is still a performance gap between UHP-SOT++ and SiamRPN++. It is worthwhile to find innovative ways to narrow down the performance gap while keeping its attractive features such as interpretability, unsupervised real-time tracking capability on small devices, etc. as future extension. One possible direction is to investigate how to exploit offline unlabeled data and learn efficiently from few annotated frames [35]. One main challenge in object tracking is the design of a robust tracker that can generalize well to various situations. This is an open problem still not solved satisfactorily by current unsupervised deep or lightweight trackers.

One of our research goals is to provide a "white-box" tracker. To achieve it, we attempt to understand the underlying tracking mechanisms of traditional trackers, identify the failure cases, and find solutions to overcome them. Furthermore, to illustrate the generalizability of our proposed solution, we have conducted extensive experiments from small-scale datasets in early days to recent large-scale datasets that cover various object classes and diverse motion trajectories and seen performance improvement. Hope that this endeavor will lead to interpretable, robust, and high-performance tracking solutions in the long run.

References

[1] A. Aggarwal, S. Biswas, S. Singh, S. Sural, and A. K. Majumdar, "Object Tracking Using Background Subtraction and Motion Estimation in MPEG Videos," in *Asian Conference on Computer Vision*, Springer, 2006, 121–30.

[2] L. Alan, T. Vojíř, L. Čehovin, J. Matas, and M. Kristan, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability," *International Journal of Computer Vision*, 126(7), 2018, 671–88.

[3] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary Learners for Real-time Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 1401–9.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional Siamese Networks for Object Tracking," in *European Conference on Computer Vision*, Springer, 2016, 850–65.

[5] G. Bhat, J. Johnander, M. Danelljan, F. S. Khan, and M. Felsberg, "Unveiling the Power of Deep Tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, 483–98.

[6] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual Object Tracking Using Adaptive Correlation Filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, 2544–50.

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *arXiv preprint arXiv:1405.3531*, 2014.

[8] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 8126–35.

[9] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient Convolution Operators for Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 6638–46.

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional Features for Correlation Filter Based Visual Tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, 58–66.

[11] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning Spatially Regularized Correlation Filters for Visual Tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, 4310–8.

[12] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative Scale Space Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8), 2016, 1561–75.

[13] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking," in *European Conference on Computer Vision*, Springer, 2016, 472–88.

[14] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive Color Attributes for Real-time Visual Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, 1090–7.

[15] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A High-quality Benchmark for Large-scale Single Object Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 5374–83.

[16] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and Deep Trackers: Recent Visual Object Tracking Approaches and Trends," *ACM Computing Surveys (CSUR)*, 52(2), 2019, 1–44.

[17] K. Hariharakrishnan and D. Schonfeld, "Fast Object Tracking Using Adaptive Block Matching," *IEEE Transactions on Multimedia*, 7(5), 2005, 853–9.

[18] A. He, C. Luo, X. Tian, and W. Zeng, "A Twofold Siamese Network for Real-time Object Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 4834–43.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 770–8.

[20] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3), 2014, 583–96.

[21] J. Janai, F. Güney, A. Behl, A. Geiger, *et al.*, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art," *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3), 2020, 1–308.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 25, 2012, 1097–105.

[23] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of Siamese Visual Tracking with Very Deep Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 4282–91.

[24] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High Performance Visual Tracking with Siamese Region Proposal Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 8971–80.

[25] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning Spatial-temporal Regularized Correlation Filters for Visual Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 4904–13.

[26] Y. Li, C. Fu, F. Ding, Z. Huang, and G. Lu, "AutoTrack: Towards High-performance Visual Tracking for UAV with Automatic Spatio-temporal regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, 11923–32.

[27] P. Liang, E. Blasch, and H. Ling, "Encoding Color Information for Visual Tracking: Algorithms and Benchmark," *IEEE Transactions on Image Processing*, 24(12), 2015, 5630–44.

[28] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang, "Deep Regression Tracking with Shrinkage Loss," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, 353–69.

[29] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical Convolutional Features for Visual Tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, 3074–82.

[30] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term Correlation Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, 5388–96.

[31] M. Mueller, N. Smith, and B. Ghanem, "A Benchmark and Simulator for UAV Tracking," in *European Conference on Computer Vision*, Springer, 2016, 445–61.

[32] H. Nam and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 4293–302.

[33] S. Pu, Y. Song, C. Ma, H. Zhang, and M.-H. Yang, "Deep Attentive Tracking via Reciprocative Learning," *arXiv preprint arXiv:1810.03851*, 2018.

[34] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged Deep Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 4303–11.

[35] I. Ruiz, L. Porzi, S. R. Bulo, P. Kontschieder, and J. Serrat, "Weakly Supervised Multi-object Tracking and Segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, 125–33.

[36] Q. Shen, L. Qiao, J. Guo, P. Li, X. Li, B. Li, W. Feng, W. Gan, W. Wu, and W. Ouyang, "Unsupervised Learning of Accurate Siamese Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, 8101–10.

[37] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, and M.-H. Yang, "Crest: Convolutional Residual Learning for Visual Tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, 2555–64.

[38] Y. Sun, C. Sun, D. Wang, Y. He, and H. Lu, "Roi Pooled Correlation Filters for Visual Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 5783–91.

[39] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese Instance Search for Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 1420–9.

[40] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end Representation Learning for Correlation Filter Based Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 2805–13.

[41] G. Wang, Y. Zhou, C. Luo, W. Xie, W. Zeng, and Z. Xiong, "Unsupervised Visual Representation Learning by Tracking Patches in Video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 2563–72.

[42] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Unsupervised Deep Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 1308–17.

[43] N. Wang, W. Zhou, Y. Song, C. Ma, W. Liu, and H. Li, "Unsupervised Deep Representation Learning for Real-time Tracking," *International Journal of Computer Vision*, 129(2), 2021, 400–18.

[44] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li, "Multi-cue Correlation Filters for Robust Visual Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 4844–53.

[45] N. Wang, W. Zhou, J. Wang, and H. Li, "Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 1571–80.

[46] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 4854–63.

[47] Q. Wu, J. Wan, and A. B. Chan, "Progressive Unsupervised Learning for Visual Object Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 2993–3002.

[48] Y. Wu, J. Lim, and M.-H. Yang, "Object Tracking Benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 2015, 1834–48, DOI: 10.1109/TPAMI.2014.2388226.

[49] J. Xing, H. Ai, and S. Lao, "Multiple Human Tracking Based on Multi-view Upper-body Detection and Discriminative Learning," in *2010 20th International Conference on Pattern Recognition*, IEEE, 2010, 1698–701.

[50] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler, "Learning Adaptive Discriminative Correlation Filters via Temporal Consistency Preserving Spatial Feature Selection for Robust Visual Object Tracking," *IEEE Transactions on Image Processing*, 28(11), 2019, 5596–609.

[51] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *Acm Computing Surveys (CSUR)*, 38(4), 2006, 13–es.

[52] G. Zhang and P. A. Vela, "Good Features to Track for Visual Slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, 1373–82.

[53] J. Zheng, C. Ma, H. Peng, and X. Yang, "Learning to Track Objects from Unlabeled Videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, 13546–55.

[54] Z. Zhou, H. Fu, S. You, C. C. Borel-Donohue, and C.-C. J. Kuo, "UHP-SOT: An Unsupervised High-Performance Single Object Tracker," in *2021 International Conference on Visual Communications and Image Processing (VCIP)*, IEEE, 2021, 1–5.

[55] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware Siamese Networks for Visual Object Tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, 101–17.