Real-time Tracking with Stabilized Frame

Zixuan Wang[1] Zhicheng Zhao[1,2] Fei Su[1,2]
[1]School of Artificial Intelligence
[2]Beijing Key Laboratory of Network System and Network Culture
Beijing University of Posts and Telecommunications, Beijing, China
princexuan@bupt.edu.cn, zhaozc@bupt.edu.cn, sufei@bupt.edu.cn *

Abstract

Deep learning methods have dramatically increased tracking accuracy benefitting from exquisite features extractor. Among these methods, siamese-based tracker performs well. However, in case of camera shaking, the objects are easily to be lost because of no consideration of camera judder, and the position of each pixel changes drastically between frames. In particular, the tracking performance would degrade dramatically in case that the target is small and moving fast, such as UAV tracking. In this paper, the ***S-Siam*** *framework is proposed to deal with this problem and improves the performance of real-time tracking. Through stabilizing each frame by estimating where the object is going to move, the camera is adjusted adaptively to keep the object in its original position. Experimental results on the VOT2018 dataset show that the proposed method obtained an* ***EAO*** *score* ***0.449****, and achieved* ***10%*** *robustness improvement compared with existing three trackers, i.e.,* ***SiamFC****,* ***SiamMask*** *and* ***SiamRPN++****, which demonstrates the effectiveness of the proposed algorithm.*

1. Introduction

Real-time visual object tracking requires learning a robust end-to-end trainable model online during the inference stage [2]. A good tracker needs to exceed the real-time processing level while ensuring accuracy and robustness. Benefit from the booming deep learning technology and increasing maturity of efficient object detection algorithm, real-time object tracking is moving towards to the real application level. For those areas where objects information needs to be inferred, such as automatic driving, unmanned aerial vehicle and intelligent robot, tracking is a fundamental computer vision task and has been receiving rapidly expanding attention lately [32]. However, there are complex and unexpected situations in real application scenarios. Facing with lens dislocation or motion blurring , even the state-of-the-art tracker may not be able to make the correct inference, causing tracking failure. How to deal with this kind of problems is a burning issue for further application.

Many benchmarks have been developed in the field of tracking [22, 23, 5, 12, 14]. Among the most recognized are object tracking benchmark (OTB50 [28] and OTB100 [29]), and visual object tracking challenges (VOT2016 [16], VOT2018 [14], VOT2019 [15]). Both benchmarks contain short sequences with different challenging situations (e.g., motion blur, size change, occlusion), and target-specific information is only available at the first frame during testing. VOT datasets have the following criteria: accuracy, robustness, and expected average overlap (EAO) rate. Accuracy is the average overlap rate between the estimated and ground truth bounding boxes when the target is successfully being tracked of. Robustness measures the ratio between the number of times the tracker loses the target and the number of resumed trackings. Expected average overlap is regarded as the primary measurement in the VOT challenge[16].

Among the available tracking algorithms, template-matching methods are the most popular ones due to their excellent calculation efficiency and accuracy [1, 9, 18, 17]. They utilize a template of the target object and match it with regions of the image in question. Siamese networks[13] are introduced to generate feature space expression for templete matching. The template usually corresponds to a patch in the previous frame, and the goal is to find the best matching area in the current frame.

Though the high accuracy of Siamese-based trackers obtained, most of them ignore the issue of camera pose estimation. When tracking the target object in a new video frame, they usually set a displacement penalty and scaling factor under the template of the previous frame, which may restrict the potential pattern of free movement of objects. In some cases, the position of the object changes rapidly by camera shaking, so that the tracker loses the object easily. Furthermore, the performance would degrade dramatically

*This work is supported by Chinese National Natural Science Foundation (61532018, U1931202, 61471049).

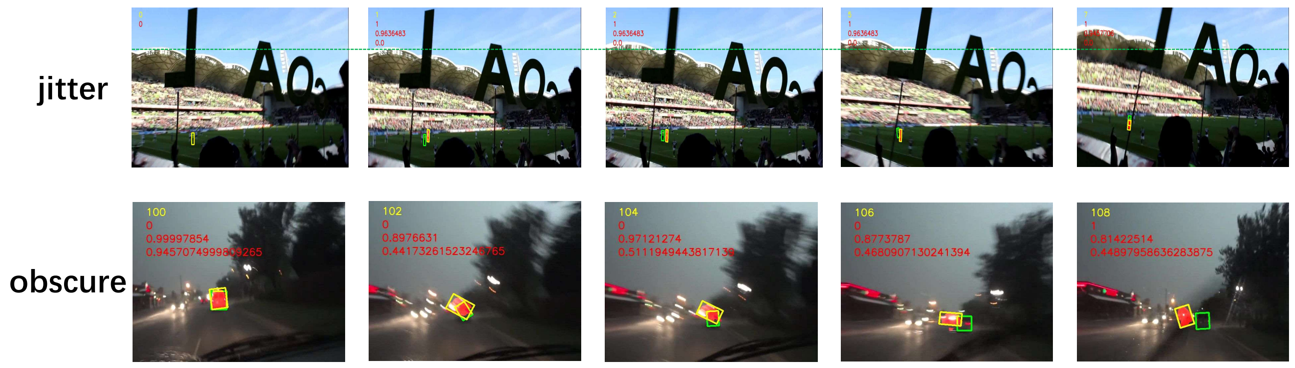

Figure 1. Failure cases on the VOT2018 dataset caused by the camara jitter and obscure. The yellow bounding box and the red mask show the results of state-of-the-art tracker SiamMask. The green bounding box indicates the ground truth. The green imaginary line in the first row points out the same horizontal position in each frame, yet there is a large fluctuation in the background. When tracking a similar scenario in actual use, the performance of robustness maybe not to our satisfaction.

for scenes with a complex background and smaller object. Because the characteristics of the object itself is difficult to capture, and the extra displacement deviation caused by the camera shake will cause serious interference to similar objects around. Some failure cases are shown in Fig. 1. Even the state-of-the-art tracker like SiamMask [27] has not solved this problem very well.

In this paper, we introduce an alternative S-Siam tracking architecture, which does not need to retrain the current trackers but stabilizes the input frame. We take inspiration from a video stabilization algorithm that has been successfully applied to recent video understanding mission. We decouple the motion of the target object from the motion through the transformation of the cross-domain coordinate system. Before tracking, we analyze the inter-frame correlation, infer adverse factors, such as shot jitter, and repair the frame to improve the current input stability. The restored frames simplify the tracking process. Experimental results show that the original tracker could be improved by adding the proposed module. In addition, our method only increases the computation amount during the preprocessing stage. Moreover, the method is relatively timesaving, and it enables the tracker to maintain its speed beyond real time. We perform comprehensive experiments on benchmarks to verify the effectiveness and present state-of-the-art results.

2. Related Work

Several different approaches have been presented to solve the visual tracking problem. Since our main contribution is to bulid shaking removal modules, we breifly review the existing state-of-the-art methods for tracking, in addition to siamese network based methods. Moreover, we review approaches which aim at video stabilization with competitive performance.

2.1. Siamese-based trackers

Siamese-based trackers are one of the most important methods in visual tracking owing to its impressive performance and speed. The first tracker, named SiamFC [1], was introduced by Bertinetto et al. in 2016. The Siamese network is trained offline on a dataset for object detection. The network inputs two images, one is an exemplar image z, and the other one is the search image x. Then, a dense response map is generated from the output of the network. SiamFC learns and predicts the similarity between the regions in x and the exemplar image z. By modifying the original SiamFC with a region proposal network (RPN [7]), Li et al. proposed SiamRPN [18] to estimate the target location with the variable bounding boxes. Inspired by the concept of anchor in object detection field, the output of SiamRPN contains a set of anchor boxes with corresponding scores. Since then, the ability of object tracking to capture semantic information had greatly improved. SiamRPN++ [17], which was inroduced by Li et al., used the deep network ResNet-50 [10] as feature extractor and achieved a state-of-the-art performance last year. Over roughly the same period, SiamMask [27] was proposed which took the advantage of instance object segmentation algorithm. A Siamese net was trained to predict a set of masks and bounding boxes on the target. The bounding boxes are estimated on the basis of the masks using rotated minimum bounding rectangle at a speed of 55 fps. Chen et al. [4] used elliptic fitting strategy to modify the rotated rectangle and improved the performance of SiamMask.

Aiming to obtain a rich multi-template representation, Axel et al. designed a dynamic target expression method

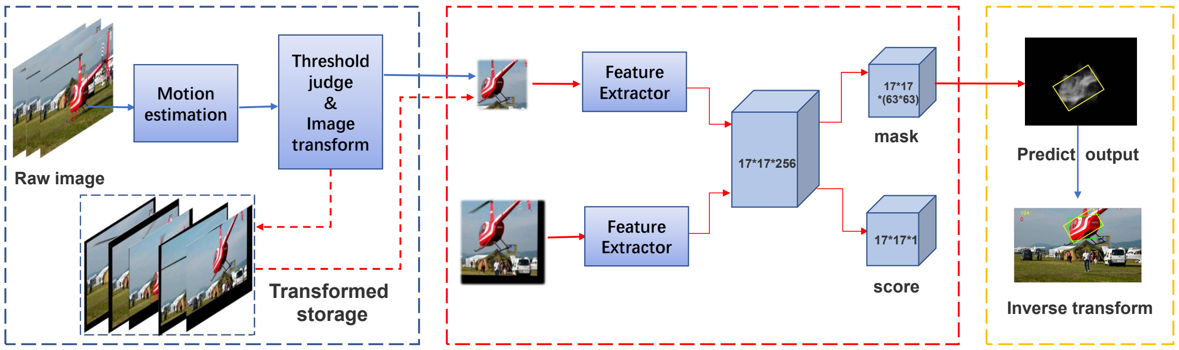

Figure 2. End-to-end schematic of S-Siam including three modules. Frame stabalization module during the warming up stage (blue dotted box), SiamMask module as the main tracker (red dotted box), coordinate restoration module in the end (orange dotted box). In the initial stage, the offset obtained by motion estimation is firstly used to judge whether the steady-state operation is needed. If so, the image is further transformed and sent into transformed storage. The images saved in the storage are coordinate transformed, and we feed them into the network as input. In the output stage, we restore the original coordinate information by the inverse transformation property.

(THOR [24]) to model the change of object template. Wang et al. [25] considered the interference of camera movement, and separated camera motion and object motion to improve the robustness of SiamMask, however, the improvement is not evident. In [6], the architecture of cascade tracker was proposed to solve the problem of background interference and scale variation. [26] added attention module to further boost the understanding of the target object. In addition, a number of recent studies [31, 20, 34] have proposed improvement approaches from different perspectives. Whereas, none of the above works effectively addressed the effect of external factors, such as camera shake, on the tracker effect. In Sec. 3, our proposed method based on video image stabilization technology to improve the robustness of the above algorithms will be described in detail.

2.2. Video stabilization

Video stabilization aims to enhance the quality of an input video by removing undesired camera motion [21]. In the process of panoramic video stitching of UAV, the video stabilization algorithm is widely used. Video stabilization system focuses on motion estimation based on video series, which includes global and local motion estimation. The first step is to find the optimal motion vector by motion etimation. The second step aims to compensate the current frame according to the motion vector and remove the jitter by the motion compensation progress. The most popular solution is to estimate the camera's motion by matching the key point features between adjacent frames. Nghia [11] used the OpenCV module to capture the information of inter frame optical flow and adopted rigid Euclidean transform to achieve efficient real-time online video stabilization. Liu et al. presented a novel video stabilization method, which models camera motion with a bundle of camera paths [19]. Their proposed model is based on a mesh-based, spatially variant motion representation and an adaptive, space-time path optimization. In [8], a grid-based tracking method is designed for an improved robustness, which produces features that are distributed evenly within and across multiple views.

Recently, deep learning method-based algorithms have paved a new path. [33] realized a combining end-to-end training for feature detection, direction assignment and descriptor generation. Xu et al. [30] introduced an adversarial network to determine the stability of a video piece, which was composed of a generative network with spatial transformer networks embedded in different layers and generated stable frame, by computing an appropriate affine transformation. However, the above algorithms require substantial computation and are difficult to migrate to real-time scenarios. Furthermore, it is not easy to perform algorithms fusion in real-time tracking scenarios, because the inference of the tracker depends on prior knowledge of the input frame.

3. Methodology

Trackers based on template matching have added penalty terms of displacement and scaling factors to the inference stage. Thus, when the camera angle has a brief shake or rapid shift, the difference between adjacent frames may lead to inference error. This phenomenon is common in the field of tracking small targets such as UAVs. The essential reason is that the homography matrix changes greatly between adjacent frames. Therefore, by making affine transformation to adjacent frames, we eliminate possible camera jitter

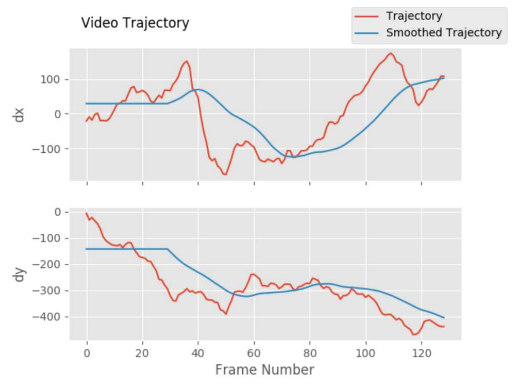

Figure 3. Accumulated trajectories of the pixel deviation on x axis direction and y axis direction (orange curves) and their smoothed result (blue curves). The original trajectories show that the instantaneous deviation of the pixel fluctuates greatly, which correlates with the video being shaky. The smoothed trajectories are obtained by averaging the neighborhood original trajectories. In this paper, the smoothing radiation distance of the previous frames is set to 30.

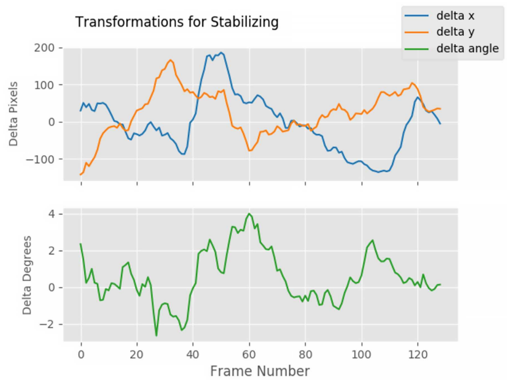

Figure 4. Computed dx (blue curve), dy (orange curve) and da (green curve) for each frame in new_trans. The results are used to form a new transformation matrix and add to the original video frame. For some abrupt points with drastic changes, the optical flow calculation method may be caused by too much interference from the scene background. If the calculated transformation matrix is directly applied to the original video frame, there maybe some mistake. We then smooth the affine matrix through maximum threshold policies.

factors and achieve more robust tracking. In this paper, we propose to combine video image stabilization with Siamese network-based tracker. We refer to the work of Nghia Ho [1] and take SiamFC, SiamRPN++ and SiamMask as examples to introduce the fusion algorithm (shown in Fig. 2).

3.1. S-Siam-stabilized tracking framework

Befor tracking, we evaluate the quality of adjacent frames of the input video. The first step is to compute transformation matrix. After comparing the performance of five different keypoint generators (HARRIS, GFTT, MSER, BRISK and ORB), we use ORB here to generate the set of keypoints. As the frame changes, the new positions of those keypoints are determined via the optical flow using Lucas-Kanade method [3].

When the real-time video stream is fed into the system, we first assign a cache to calculate the changes in the perspective within a certain range. Once the number of video frames is greater than the cache, we simulate a queuing model and update the cache information frame by frame. Assume I_t contains a set of keypoints on frame t, and I_{t+1} is the result corresponding to each keypoint between frame t and $t+1$ by optical flow.

$$\mathrm{I}_t = \{(x_{t,1}, y_{t,1}), (x_{t,2}, y_{t,2}), ..., (x_{t,n}, y_{t,n})\} \tag{1}$$

$$\mathrm{I}_{t+1} = \{(x_{t+1,1}, y_{t+1,1}), (x_{t+1,2}, y_{t+1,2}), ..., (x_{t+1,n}, y_{t+1,n})\} \tag{2}$$

where I_t and I_{t+1} are used to generate frame-to-frame transformation using rigid Euclidean transform. We need

[1]https://adamspannbauer.github.io/python_video_stab/html

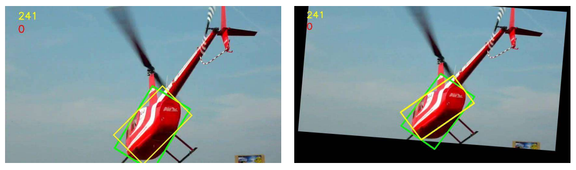

Figure 5. Comparison of the original input frame with the transformed one after affine transformation. The black edges around the right image are the shiftable spaces that we've set aside.

to compute the 2D transform matrix $trans$ as T.

$$T = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \tag{3}$$

where T is computed using the least variance method. Each point (x, y) in I_t and the corresponding (x', y') in I_{t+1} must satisfy the following equation:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{4}$$

We define dx, dy, and da as the deviation of the x-axis direction, y-axis direction, and clockwise offset angle respectively. They can be expressed as $dx = c$, $dy = f$, and $da = \arctan(d/a)$. We accumulate the information of each frame and obtain the original $trajectory$ (orange curve of Fig. 3). Then, we smooth out the video jitter using moving average window and obtain the $smoothed\_trajectory$

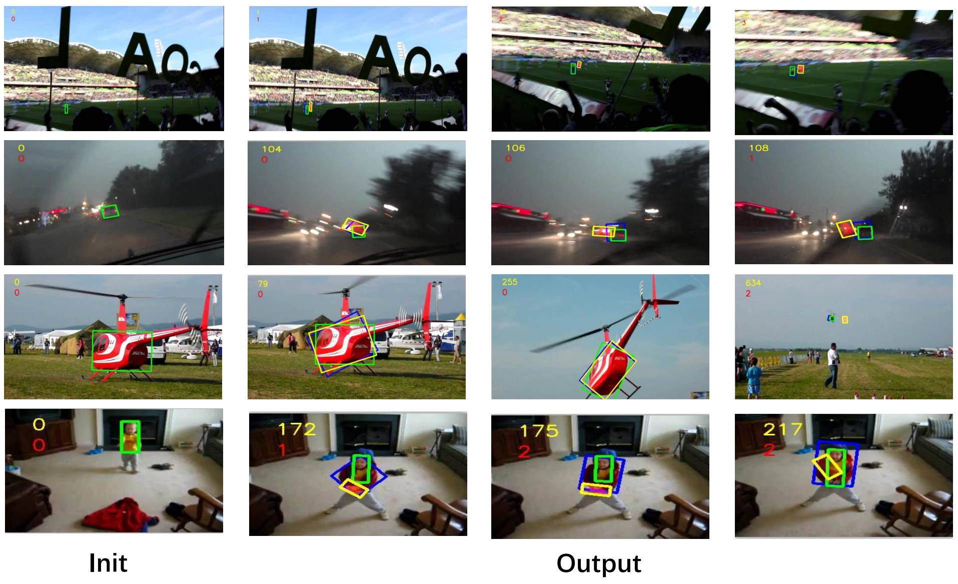

Figure 6. Tracking results of sequences *soccer2*, *wiper*, *helicopter* and *blanket* on the VOT2018. The original version SiamMask (yellow box) loses the target, whereas the prediction of our proposed approach (blue box) exhibits an accurate IoU between the ground truth (green box). The first column represents the initialization frame of the current video sequence. The four sequences shown in the figure all have the situation of camera jitter and image obscure. The first row *soccer2* is even more typical, in which the target object takes only a small part of the image and there are a lot of similar jamming targets around. Debuffeting is especially important when dealing with such situation.

(blue curve of Fig. 3). We calculate the difference between $smoothed\_trajectory$ and $trajectory$, and set a threshold that needs to be debuffeted by comparing the sum of square distance with the x axis and y axis. Finally, we add this difference with orignal $trans$ to obtain the new_trans corresponding to the current frame (shown in Fig. 4).

$$new\_trans = trans + (smoothed\_trajectory - trajectory) \tag{5}$$

For the elements dx, dy, da in new_trans, we place some constraints so that they would not deviate too much. Borders are added to the original image when the image changes. This addition is performed to avoid information loss caused by excessive displacement (shown in Fig. 2). Thus, we assume that the size of the added borders is N. dx_t, dy_t, da_t are the calculated results of the current frame; dx'_{t-1}, dy'_{t-1}, da'_{t-1} are the actual transform from the previous frame; and dx'_t, dy'_t, da'_t are the actual transform in the current frame, which can be obtained by:

$$dx'_t = \begin{cases} (N + TH_{x1}) & if \quad |dx_t| - N > TH_{x1} \\ TH_{x2} & if \quad |dx_t - dx'_{t-1}| > TH_{x2} \\ dx_t & otherwise \end{cases} \tag{6}$$

where TH_{x1} and TH_{x2} are two predefined thresholds. TH_{x1} limits the movement of the camera to keep the target in the image, and TH_{x2} limits the range of the transformation. Similarly, we have TH_{y1} and TH_{y2}. At the same time, TH_a is used to keep the difference between angles in control. We build the transformation matrix M_t of frame t to obtain the stabilized frame s_img_t. Fig. 5 shows the changes for one frame of input.

$$M_t = \begin{bmatrix} \cos(d'a_t) & \sin(d'a_t) & d'x_t \\ -\cos(d'a_t) & \cos(d'a_t) & d'y_t \end{bmatrix} \tag{7}$$

$$s\_img_t = M_t * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \forall (x, y) \in img_t \tag{8}$$

where s_img_t is the input of SiamMask, and its output is Rec_out. Since we have changed img_t into s_img_t, the output Rec_out is in a transformed state. Therefore, it needs to be changed back to the original state, Rec_real, through inverse property (operating by $-M_t$).

$$Rec\_real = -M_t * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \forall (x, y) \in Rec\_out \tag{9}$$

3.2. Implementation details

During tracking, we set the length of cache queue to 30 frames. As we said before, ORB is selected to be the keypoint generator. For hyperparameters, we set $N = 20$, $TH_{x1} = 60$, $TH_{y1} = 30$, $TH_{x2} = 20$, $TH_{y2} = 10$ and $TH_a = 0.3$.

Benchmarks	VOT2019			VOT2018			VOT2016			Speed ↓
	A	R ↑	EAO ↑	A	R ↑	EAO ↑	A	R ↑	EAO ↑	(fps)
SiamFC	0.477	0.687	0.204	0.479	0.492	0.224	0.512	0.331	0.301	121
S_SiamFC (Ours)	0.459	0.577	0.207	0.467	0.450	0.231	0.487	0.261	0.328	58
SiamRPN++	0.595	0.467	0.290	0.600	0.234	0.415	0.643	0.200	0.461	34
S_SiamRPN++ (Ours)	0.594	**0.446**	0.293	0.600	**0.229**	0.425	0.641	**0.186**	**0.475**	23
SiamMask_E	**0.646**	0.497	0.303	**0.649**	0.267	0.432	**0.669**	0.233	0.450	55
S_SiamMask_E (Ours)	0.637	0.477	**0.309**	0.637	0.243	**0.449**	0.655	0.210	0.463	33

Table 1. Comparison with the original Siamese trackers on VOT2019, VOT2018, and VOT2016. Pretrained models are the same in each pair provided by the authors. SiamMask_E [4] represents the advanced SiamMask proposed by Chen et al.. In this paper, more accurate tracking results are obtained by optimizing the fitting process of rotating rectangular frame, and we use it to represent the SiamMask. Compared with the baseline model, all the performance of our trackers have been improved. As can be seen from the table, the major improvement occurred in robustness. For image input with different resolutions, the speed index of our method will fluctuate slightly. The final result is averaged over the entire test sequences.

Generators	A	R	EAO	Speed
GFTT	**0.642**	0.249	0.441	30
ORB	0.637	**0.243**	**0.449**	33
BRISK	0.625	0.283	0.390	38
MSER	0.635	0.245	0.444	34
HARRIS	0.636	0.270	0.412	**39**

Table 2. Comparing five different keypoint generators on VOT2018 using $SiamMask\_E$ (Subsection 3.1). According to the primary measurement (EAO), we choose ORB for final use.

Threshold value		A	R	EAO
TH_{x1} ($2TH_{y1}$)	**60**	0.637	**0.243**	**0.449**
	80	0.640	0.260	0.435
	100	**0.645**	0.277	0.418
TH_{x2} ($2TH_{y2}$)	**20**	0.637	**0.243**	**0.449**
	40	**0.639**	0.251	0.436
	80	0.630	0.272	0.402
TH_a	0.1	**0.641**	0.270	0.413
	0.3	0.637	**0.243**	**0.449**
	0.5	0.634	0.244	0.439
	1.0	0.625	0.264	0.410

Table 3. Comparing the performance with different threshold values in Subsection 3.1 on VOT2018. The aspect ratio of the frame is close to 2 to 1, thus we assume $TH_{x1} = 2TH_{y1}$, $TH_{x2} = 2TH_{y2}$.

4. Experimental Results

To verify the performance of our proposed method, VOT2016, VOT2018, and VOT2019 datasets are used in our experiments. To ensure fairness, we perform all experiments on the same PC with an Intel i7-4790K CPU, 16GB RAM, NVIDIA GTX 1080Ti GPU.

Sequence	Stabilization	Original	S-Siam
soccer2	Yes	3	0
soccer1	Yes	4	3
wiper	Yes	2	1
helicopter	Yes	2	1
blanket	Yes	2	1
rabbit	Yes	2	3
girl	No	4	4
book	No	4	4

Table 4. The comparison results for robustness performance on VOT2018. The first column is the name of the different test sequences. The second column represents whether or not to perform stabilization. The third column and the fourth column represent the times that target object is lost in the sequence. We use SiamMask as the base model and the result is show in the third column. Our S-Siam has a positive response effect to the sequence with the problem of de-buffeting. It is worth noting that the second half of the table shows some sequences that do not work. These sequences have some problems such as background confusion with objects and occlusion with other objects, which need to be further discussed in the future.

4.1. Hyperparameters selection of S-Siam

The comparison results are shown in Table 2 and 3, which are tested on the VOT2018 dataset solely. Table 2 gives the performance comparision of different keypoint generators (GFTT, ORB, BRISK, MSER, HARRIS). ORB is selected to be the keypoint generator in our S-Siam framework. Table 3 shows the performance of different values of TH_{x1}, TH_{y1}, TH_{x2}, TH_{y2}, TH_a. When selecting a variable, other variables are controlled for fixing. The combination of 60, 30, 20, 10, and 0.3 obtains the best result, and we use these parameters as the hyperparameters to test the overall results.

4.2. Overall results

Table 1 presents the comparison results between the state-of-the-art Siamese-based tracking algorithms SiamFC, SiamPRN++ and SiamMask on the VOT2016, VOT2018, and VOT2019 datasets. For each group, the result of original version based on the dataset using the pretrained model exposed by the authors. The prefix "S_" represents trackers that use our S-Siam framework. We verify on three benchmarks that all the original trackers improved their robustness by an average of 10% and achieved a new peak EAO score with the introduction of the S-Siam strategy. Especially, our tracker $S\_SiamMask\_E$ has obtained a robustness of 0.243 and a competitive EAO score of 0.449 on the VOT2018 dataset. Due to the additional stabilization processing of the overall process, the time performance is reduced slightly. Table 4 shows the specific amount of lost frame in different sequences on VOT 2018 dataset. The performance of accuracy indicates that the possible misjudgement of camera pose estimation results in a fluctuation. In general, our algorithm can significantly improve the robustness of the tracker with almost no loss of precision.

5. Conclusion

In this paper, we propose the S-Siam framework and introduce it into the existing Siamese-based trackers. Considering the real-time requirement, we adopt an optical flow matching de-buffeting strategy which directly acted on the input video frame by calculating the affine matrix. After estimating and smoothing the camera pose changing, our method can eliminate the negative effects caused by camera shaking or rapid displacement. We improve the performance of rubustness by 10% on the VOT dataset and operate at over 30 FPS. It is obvious that using a more efficient stabilization method can improve the tracking accuracy for real-time trackers, which can be viewed as a focus of application level considerations.

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6182–6191, 2019.

[3] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International journal of computer vision*, 61(3):211–231, 2005.

[4] Bao Xin Chen and John K Tsotsos. Fast visual object tracking with rotated bounding boxes. *arXiv preprint arXiv:1907.03892*, 2019.

[5] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5374–5383, 2019.

[6] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7952–7961, 2019.

[7] R Faster. Towards real-time object detection with region proposal networks shaoqing ren [j]. *Kaiming He, Ross Girshick, and Jian Sun.*

[8] Heng Guo, Shuaicheng Liu, Tong He, Shuyuan Zhu, Bing Zeng, and Moncef Gabbouj. Joint video stitching and stabilization from moving cameras. *IEEE Transactions on Image Processing*, 25(11):5491–5503, 2016.

[9] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[10] 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.

[11] Nghia Ho. *Simple video stabilization using OpenCV*, 2014.

[12] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018.

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

[14] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[15] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir,

Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2019.

[16] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Gustav Hager, Alan Lukezic, Abdelrahman Eldesokey, et al. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1972, 2017.

[17] 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 Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[18] 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.

[19] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *Acm Transactions on Graphics*, 32(4):1–10, 2013.

[20] Alan Lukežič, Jiří Matas, and Matej Kristan. D3s–a discriminative single shot segmentation tracker. *arXiv preprint arXiv:1911.08862*, 2019.

[21] Sunil Sankol S M, Praveenkumar B S, and Vanishree K Rao. Real time video stabilization using plk tracking algorithm. *International Journal of Science and Research (IJSR)*, 2017.

[22] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[23] 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.

[24] Axel Sauer, Elie Aljalbout, and Sami Haddadin. Tracking holistic object representations, 2019.

[25] Jianren Wang, Yihui He, Xiaobo Wang, Xinjia Yu, and Xia Chen. Prediction-tracking-segmentation. *arXiv preprint arXiv:1904.03280*, 2019.

[26] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen 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*, pages 4854–4863, 2018.

[27] 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 Conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019.

[28] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

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

[30] Sen-Zhe Xu, Jun Hu, Miao Wang, Tai-Jiang Mu, and Shi-Min Hu. Deep video stabilization using adversarial networks. In *Computer Graphics Forum*, volume 37, pages 267–276. Wiley Online Library, 2018.

[31] Yinda Xu, Zeyu Wang, Zuoxin Li, Yuan Ye, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. *arXiv preprint arXiv:1911.06188*, 2019.

[32] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[33] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[34] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4010–4019, 2019.