Software Impacts 12 (2022) 100266

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

A Siamese Network for real-time object tracking on CPU

Daitao Xing [a,*], Nikolaos Evangeliou [b], Athanasios Tsoukalas [b], Anthony Tzes [c]

[a] *New York University, USA*
[b] *New York University Abu Dhabi, United Arab Emirates*
[c] *Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, United Arab Emirates*

ARTICLE INFO

Keywords:
Object tracking
Reduced complexity
Mobile devices
Siamese Network

ABSTRACT

Visual object tracking methods depend upon deep networks that can hardly meet real-time processing requirements on mobile platforms with limited computing resources. In this work, we propose a real-time object tracking framework by enhancing a lightweight feature pyramid network with Transformer architecture to construct a robust target-specific appearance model efficiently. We further introduce the pooling attention module to avoid the computation and memory intensity while fusing pyramid features with the Transformer. The optimized tracker operates over 45 Hz on a single CPU, allowing researchers to deploy it on any mobile device with limited power resources.

Code metadata

Current code version	*v1.0*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-196
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/6988064/tree/v1
Legal Code License	*MIT License.*
Code versioning system used	*git*
Software code languages, tools, and services used	*python*
Compilation requirements, operating environments & dependencies	*Pytorch 1.7.0, Python 3.7+, OpenVINO 2021.4.1 LTS*
If available Link to developer documentation/manual	
Support email for questions	daitao.xing@nyu.edu

1. Introduction

Visual Object Tracking (VOT) has attracted increasing attention in recent years, given its applications in several fields, including path planning [1], visual surveillance [2] and border security [3]. While extensive achievements have been made towards powerful object tracking methods, most of those trackers employ deep networks, complex structures, or online update mechanisms and require GPU acceleration to achieve real-time processing. Designing an efficient tracker for mobile devices with lower number crunching capabilities and limited computing resources remains a challenging topic.

In this work, we propose a novel object tracker optimized for mobile devices, which achieves high-performance and real-time speed using only a CPU-component. Firstly, we consider a lightweight backbone, i.e., ShuffleNetV2 [4], for efficient feature extraction. Multi-scale features from Feature Pyramid Network (FPN) [5] enhance the model representative ability by exploiting low/high-level semantics contexts. Inspired by the recent development of Transformer [6], we integrate the self-attention mechanism into a pyramid network to model global dependencies, which is vital for tracking performance, in complicated real-world scenarios. Furthermore, a pooling attention layer is designed to control the model complexity, yielding robust target-specific appearance representation efficiently.

We implement the tracking framework, Siamese Transformer Pyramid Network (SiamTPN) [7] in Pytorch. To demonstrate the effectiveness of SiamTPN, we conduct comprehensive experiments on both

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: daitao.xing@nyu.edu (D. Xing), nikolaos.evangeliou@nyu.edu (N. Evangeliou), athanasios.tsoukalas@nyu.edu (A. Tsoukalas), anthony.tzes@nyu.edu (A. Tzes).

https://doi.org/10.1016/j.simpa.2022.100266
Received 23 December 2021; Received in revised form 9 February 2022; Accepted 22 February 2022

Table 1
Comparison between Siamese based trackers. All speed tested on GPU. The backbones contain AlexNet [12], Resnet-50 (R50) [16] and ShuffleNet (Shuffle) [4].

BackBone	AlexNet	AlexNet	R50	R50	R50	AlexNet	Shuffle
#Params (M)	3.1	3.1	25.5	25.5	25.5	3.1	0.8
GFLOPS	4.33	4.33	4.12	4.12	4.12	4.33	0.16
	Siam FC [10]	Siam RPN [11]	Siam RPN++ [13]	Siam CAR [14]	Siam Attn [15]	SiamTPN	SiamTPN
#Params (M)	3.1	22.63	53.95	51.38	140.24	6.47	4.24
GFLOPS	5.05	9.23	59.56	59.31	59.93	6.06	1.31
FPS (GPU)	27	160	35	52	45	105	85
FPS (CPU)	6	10	3	6	4	14	32

prevalent tracking benchmarks and real-world field tests. Our tracker operates at over 30 FPS on an i7-CPU Intel NUC. We also utilize ONNX [8] and OpenVINO [9] to boost the inference speed further, The optimized version runs at over 45 FPS on a single CPU with state-of-the-art performance.

2. Framework complexity analysis

We employ the pretrained lightweight network ShuffleNetV2 [4] as the backbone for feature extraction. Given the input images, we extract the feature maps of spatial ratio equal to $\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ respectively. Due to the generalization ability of the proposed tracker, the model can be used to track generic objects without further training or modification.

The parameter count and MACs of models are important indicators that directly affect inference speed. We compare the FLOPS and Prams with 5 famous trackers, including two trackers (SiamFC [10], SiamRPN [11]) based on AlexNet [12], three trackers (SiamRPN++ [13], SiamCAR [14] and SiamAttn [15]) based on Resnet-50 [16]. The comparison is shown in Table 1. AlexNet is very fast on GPU, considering its simple architecture, while Resnet-50 is too heavy for CPU computing. ShuffleNet is very lightweight and friendly for CPU computing. All experiments are conducted with a 2060 Nvidia for GPU test and an Intel I7-9750H for the CPU test.

3. Description and usage

We implemented a modular API for the proposed tracker to allow easy usage for testing on benchmark or deployment on mobile devices. With PyTorch, we implemented wrappers to handle various data processing and usage scenarios. An example of using SiamTPN to track object from video stream is shown in Fig. 1. The model is initialized with the first frame from a local video file or camera stream. The user is required to select a bounding box from the first frame as **object_state**, which includes four numbers (x, y, w, h) representing the upper-left corner coordinates, width and height. The selected object is cropped from the first frame and resized into 80×80 to initialize the tracker. The object state is updated after running tracking algorithm on each frame. The output has the same format with the **object_state**, representing the bounding box in new frame.

In addition to object tracking from video sources, the PyTorch based wrappers API supports evaluation on multiple benchmarks including UAV123 [17], UAV20L [17], Got10K [18], etc. An example of testing performance on UAV123 is shown in Fig. 2. In this example, the tracker is applied to the whole dataset or partial sequences specified by users.

4. Impact overview

Real-time visual object tracking with CPU remains challenging due to the limited computing resources and scenarios complexity, limiting its deployment on mobile devices. SiamTPN is designed to achieve real-time visual tracking on CPU and maintain the competitive performance compared to the state-of-the-art GPU-based trackers. To verify the reliability of SiamTPN, we evaluate the tracking performance on prevalent

```python
import cv2
import torch
from lib.test.evaluation.tracker import Tracker

# Initilize Input stream
cap = cv2.VideoCapture(video_file_path)

# Get first frame as template
_, frame = cap.read()
object_state = cv2.selectROI(frame)

# Initialize Tracker
tracker = Tracker(tracker_params)
tracker.initialize(frame, object_state)

# Tracking object in each frame
while True:
    ret, frame = cap.read()

    if frame is None:
        break

    object_new_state = tracker.track(frame)
```

Fig. 1. Example of using SiamTPN to track object from video stream. In this example, the input source is either a prerecord video file or live camera stream.

```python
import cv2
import torch
from lib.test.evaluation import get_dataset
from lib.test.evaluation.tracker import Tracker

# Initialize dataset
dataset = get_dataset(dataset_name="UAV123")

# Select a specific sequence in dataset
if sequence is not None:
    dataset = [dataset[sequence]]

# Running tracker on each sequence
for seq_ in dataset:

    # Get first frame and object state
    frame = seq_.frames[0]
    object_state = seq_.init_info()

    # Initilize Tracker
    tracker = Tracker(tracker_params)
    tracker.initialize(frame, object_state)

    # Tracking object in each frame
    for frame in seq_.frames:
        object_new_state = tracker.track(frame)
```

Fig. 2. Example of using SiamTPN to evaluate performance on benchmarks.

Table 2
Comparison results on UAV123 dataset [17] in terms of precision (Prec.), success (Succ.) and speed (FPS). HF refers to handcraft features, R50 (18) is Resnet-50 (18) [16], Alex, Shuffle, VGG represents AlexNet [12], ShuffleNet [4], VGGNet [19] respectively. GPU speeds are mark with *, Our SiamTPN based on AlexNet and ShuffleNet exhibit promising results. The top three trackers are shown in red, green and blue fonts.

Trackers	KCF [20]	BACF [21]	CSRDCF [22]	ARCF [23]	Auto Track [24]	ECO [25]	Siam RPN++ [13]	DaSiam [26]	HiFT [27]	Siam BAN [28]	Siam CAR [14]	Siam Attn [15]	DiMP [29]	ATOM [30]	Siam TPN	Siam TPN
Feat	HF	HF	HF	HF	HF	VGG	R50	Alex	R50	R50	R50	R50	R50	R18	Alex	Shuffle
Prec.	52.3	66.2	67.6	67.1	68.9	75.2	76.9	60.8	78.7	83.3	76	84.5	84.9	83.7	79	85.83
Succ.	33.1	46.1	48.1	46.8	47.2	52.2	57.9	40	58.9	63.1	61.4	65	65.4	65	59.3	66.04
FPS	95	14.4	58	15.3	65.4	45	35*	134*	130*	40*	52*	45*	45*	46*	105*	32.1

Fig. 3. Visualization results on UAV123 benchmark.

Fig. 4. Visualization results on fieldtest.

tracking benchmarks as well as real-world field tests. Figs. 3 and 4 shows the tracking results on UAV123 [17] benchmark and field tests respectively. Table 2 compares the tracking performance with State-Of-The-Art (SOTA) methods on UAV123. We include both Discriminative Correlation Filter (DCF) based trackers and Deep Siamese Network based trackers for a thorough comparison. Our tracker achieves SOTA results and maintains real-time processing speed on CPU. The experiments demonstrate both robustness and effectiveness of SiamTPN, which is useful for engineers to integrate the tracker into real-world applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project was supported by New York University Abu Dhabi Global Fellowship, UAE

References

[1] A. Shahzad, X. Gao, A. Yasin, K. Javed, S.M. Anwar, A vision-based path planning and object tracking framework for 6-DOF robotic manipulator, IEEE Access 8 (2020) 203158–203167.

[2] D. Xing, A. Tsoukalas, N. Giakoumidis, A. Tzes, Computationally efficient RGB-t UAV detection and tracking system, in: 2021 International Conference on Unmanned Aircraft Systems, ICUAS, IEEE, 2021, pp. 1410–1415.

[3] A. Tsoukalas, D. Xing, N. Evangeliou, N. Giakoumidis, A. Tzes, Deep learning assisted visual tracking of evader-UAV, in: 2021 International Conference on Unmanned Aircraft Systems, ICUAS, IEEE, 2021, pp. 252–257.

[4] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient CNN architecture design, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 116–131.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: Proceedings of the European Conference on Computer Vision, 2020, pp. 213–229.

[7] D. Xing, N. Evangeliou, A. Tsoukalas, A. Tzes, Siamese transformer pyramid networks for real-time UAV tracking, 2021, arXiv preprint arXiv:2110.08822.

[8] J. Bai, F. Lu, K. Zhang, et al., Onnx: Open neural network exchange, 2019, GitHub Repository.

[9] Intel Corporation, OpenVINO Toolkit, URL https://software.intel.com/en-us/openvino-toolkit.

[10] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, P.H. Torr, Fully-convolutional Siamese networks for object tracking, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 850–865.

[11] B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8971–8980.

[12] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.

[13] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, J. Yan, Siamrpn++: Evolution of Siamese visual tracking with very deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4282–4291.

[14] D. Guo, J. Wang, Y. Cui, Z. Wang, S. Chen, Siamcar: Siamese fully convolutional classification and regression for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 6269–6277.

[15] Y. Yu, Y. Xiong, W. Huang, M.R. Scott, Deformable Siamese attention networks for visual object tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 6728–6737.

[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[17] M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for UAV tracking, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 445–461.

[18] L. Huang, X. Zhao, K. Huang, Got-10k: A large high-diversity benchmark for generic object tracking in the wild, IEEE Trans. Pattern Anal. Mach. Intell. (2019).

[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[20] J. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Trans. Pattern Anal. Mach. Intell. 37 (3) (2014) 583–596.

[21] H. Kiani Galoogahi, A. Fagg, S. Lucey, Learning background-aware correlation filters for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1135–1143.

[22] A. Lukezic, T. Vojir, L. Čehovin Zajc, J. Matas, M. Kristan, Discriminative correlation filter with channel and spatial reliability, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6309–6318.

[23] Z. Huang, C. Fu, Y. Li, F. Lin, P. Lu, Learning aberrance repressed correlation filters for real-time UAV tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2891–2900.

[24] Y. Li, C. Fu, F. Ding, Z. Huang, G. Lu, AutoTrack: TOwards high-performance visual tracking for UAV with automatic spatio-temporal regularization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 11920–11929.

[25] M. Danelljan, G. Bhat, F. Shahbaz Khan, M. Felsberg, Eco: Efficient convolution operators for tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6638–6646.

[26] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 101–117.

[27] Z. Cao, C. Fu, J. Ye, B. Li, Y. Li, HiFT: Hierarchical feature transformer for aerial tracking, 2021, arXiv preprint arXiv:2108.00202.

[28] Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 6668–6677.

[29] G. Bhat, M. Danelljan, L.V. Gool, R. Timofte, Learning discriminative model prediction for tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6182–6191.

[30] M. Danelljan, G. Bhat, F.S. Khan, M. Felsberg, Atom: Accurate tracking by overlap maximization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4660–4669.