Comparison of Single Object Tracking Algorithms in OpenCV

Rafael Rodríguez & Alba Martínez

Abstract

Object tracking is a critical component in many computer vision applications, such as surveillance, human-computer interaction, and autonomous vehicles. However, choosing an appropriate tracker for a specific application is not trivial. In this paper we present a comprehensive comparison of single object trackers implemented in OpenCV library. We evaluate these algorithms across multiple performance metrics using an OPE approach over the GOT-10k benchmark in order to report mean IoU values, mean FPS per tracker and the average number of tracking failures per tracker and video as well as confidence intervals for our results.

Keywords – Single Object Tracking, OpenCV, benchmark

1. Introduction

Object tracking is an important subarea of computer vision, it is a crucial step in a wide range of more complex systems such as robotics, autonomous driving or surveillance. It is a widely studied problem where a plethora of algorithms have been proposed over the past decades and years [1], [2].

However, even nowadays, it is not possible to propose a universal single object tracking algorithm that works well in every condition [3]. There are several factors affecting the performance of a tracker: illumination variation, background clutters, low resolution, target bounding box scale variation, occlusions, target movement, fast motion, etc. [4].

Moreover, the selection of an appropriate tracker must be done not only considering the functioning principle of the tracker, but also the target application and additional requirements such as robustness, accuracy or the real time processing needs [3].

OpenCV [5] is a popular C++ and Python computer vision framework that, as of today, provides implementations for 11 single object tracking algorithms, including feature based and estimation based classic trackers and state-of-the-art learning based models [4].

There are several existing surveys that compare OpenCV's single object tracking algorithms (e.g.: [6], [3], [7] or [8]), but to our knowledge the most general one dates from 2020 and fails to evaluate some of the new learning based trackers that are considered to be the state-of-the-art.

In this paper we evaluate all 11 implementations across multiple performance metrics over the GOT-10k [9] tracking benchmark reporting mean IoU values, mean FPS per tracker and the average number of failures per tracker and video, as well confidence intervals for our results.

2. Materials & Methods

2.1. Trackers

A total of 11 different trackers are evaluated in this paper. Table 1 provides a short description of each one of them.

Tracker	Description
Boosting	Real-time object tracking based on a novel on-line version of the AdaBoost algorithm
CSRT	High-performance tracker that uses the correlation filter framework with spatial and channel reliability checks
KCF	Extends the basic correlation filter framework by incorporating kernelized components to improve tracking performance
MedianFlow	Utilizes a sparse optical flow algorithm, tracking points within the object region
MIL	It learns a classifier from a set of labeled bags, where each bag contains multiple instances of an object
MOSSE	Utilizes a fast Fourier transform to learn and update the object's appearance model
TLD	Integrates tracking, learning, and detection to manage tracking failures effectively
GOTURN	Employs deep convolutional neural networks to predict object locations
DaSiamRPN	Integrates a Siamese network with a Region Proposal Network
Nano	Lightweight DNN-based general object tracking algorithm
ViT	Employs vision transformers

Table 1. Evaluated object tracking algorithms.

All trackers are evaluated using the default OpenCV's parameters. For those based on deep-learning algorithms we have downloaded the trained models following OpenCV's official documentation.

2.2. Benchmark

There are several benchmark datasets that can be used for this task such as OTB100, VOT2019, UAB123 or GOT-10k [6].

We choose GOT-10k as reference benchmark dataset to evaluate the trackers. This dataset contains over 10.000 video segments of real world moving objects with more than 560 object classes and more than 80 motion patterns [9].

The dataset is divided into training, validation and test sets. We work with the validation set as it provides a reduced size (180 videos) and ground truth bounding box labels to compute the target metrics.

2.3. Evaluation Strategy

We employ a One Pass Evolution (OPE) approach where the tracker is initialized with the starting frame's ground truth bounding box and it is not reinitialized if there is a tracking failure (i.e.: the tracker does not find the target object in the frame).

As main evaluation metric we use the intersection over union (IoU), computed frame-wise per tracker. After evaluating a video, the mean IoU and the standard deviation are stored. Furthermore, we also target the mean FPS per video and its standard deviation.

Finally, to evaluate the robustness of the tracker we count the number of frames with tracking failures.

The algorithms and scripts used for data analysis in this study are available at [10].

3. Results

We provide several result summaries:

Figure 1 shows a success plot and its corresponding AUCs per tracker as described in [6].

Figure 2 shows a bar plot representing the mean IoU per tracker and the 95% confidence intervals (CI).

Figure 3 combines a violin plot and a strip plot. Each tracker is represented by a set of points (videos) and the mean IoU of each video.

Table 2 represents the mean FPS per tracker, the 95% CIs. Only frames without failure are considered as valid for the mean FPS calculation, hence we also provide the sample size used for each tracker.

Table 3 represents the average number of failures per video and tracker.

Our results show that the best performing model in terms of both IoU and AUC is the ViT tracker, followed by Nano and DaSiamRPN.

Regarding real time processing, the three fastest models are MOSSE, MedianFlow and Nano. MOSSE and MedianFlow yield a similar mean FPS value while there is a significant decrease in speed in the Nano tracker.

In terms of robustness all Boosting, MIL, GOTURN, DaSiamRPN, Nano and ViT yield an average of 0 failures per video while the least robust trackers are MOSSE, MedianFlow and KCF.

4. Discussion & Conclusions

In light of our results the best performing tackers are learning based models that employ deep learning during the tracking, these models constitute the state-of-the-art in single object tracking and have overpassed the classic approaches.

Both ViT and Nano provide good tracking results. DaSiamRPN, although being close in performance, lacks processing speed, making it one of the slowest trackers.

These trackers work well in off-the-shelf scenarios, but performance gets degraded in certain situations, as shown by Figure 3, demonstrating the need to fine-tune the models on specific datasets to improve performance.

Regarding the classic approaches we have found that Boosting, MIL and GOTURN provide robust results, but with under real time speed and deficient performance. In general, classic approaches show a degraded performance compared to state-of-the-art models. This significant difference might be caused by the use of default parameters within these trackers.

It must be noted that the OPE approach, although being the most common, provides a disadvantage as the outcome of the tracking depends on the starting frame, hence the tracking results might vary. Furthermore, most of the classic approach trackers do not include any type reinitializing mechanism.

5. Future Directions

To overcome the limitations described above we propose a set of future directions aiming to provide more robust and generalizable results:

Combine the OPE approach with more complex evaluation strategies that take into account the initialization problem, such as Temporal Robustness Evaluation or Spatial Robustness Evaluation, as described in [4].

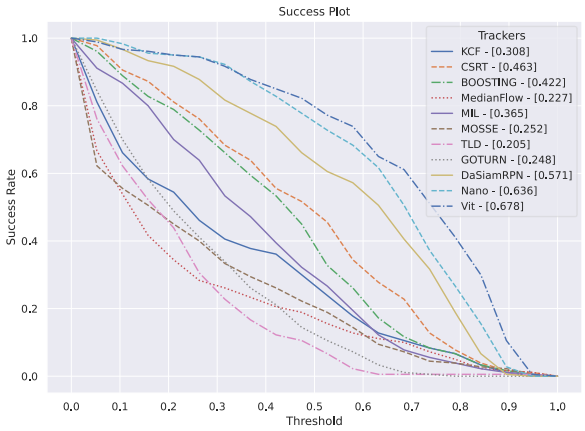

Figure 1. Succes plot and AUC per tracker.

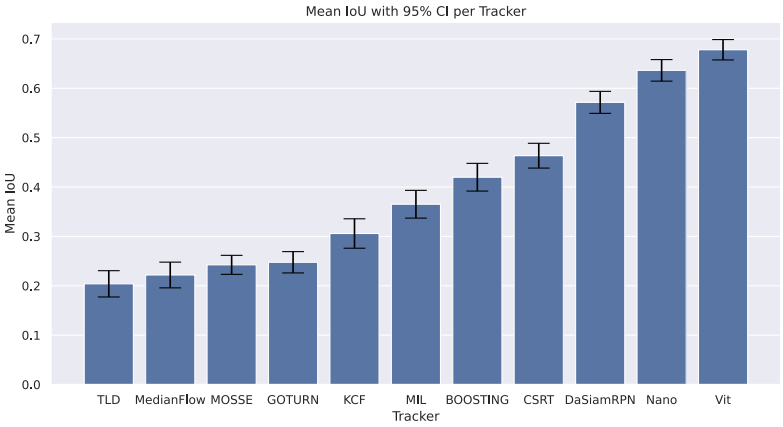

Figure 2. Mean IoU and 95% CI per tracker.

Tracker	Mean FPS	CI	N
TLD	9.222440	0.165231	125
DaSiamRPN	11.991278	0.089782	180
CSRT	14.673335	0.278999	147
MIL	16.332171	0.386475	180
GOTURN	17.762692	0.296051	180
Boosting	20.893065	0.279157	180
KCF	21.572871	0.481824	54
ViT	66.225394	0.808232	180
Nano	80.625924	1.444249	180
MedianFlow	200.324210	6.103787	38
MOSSE	**279.250487**	9.997345	66

Table 2. Mean FPS, confidence intervals and sample size per tracker.

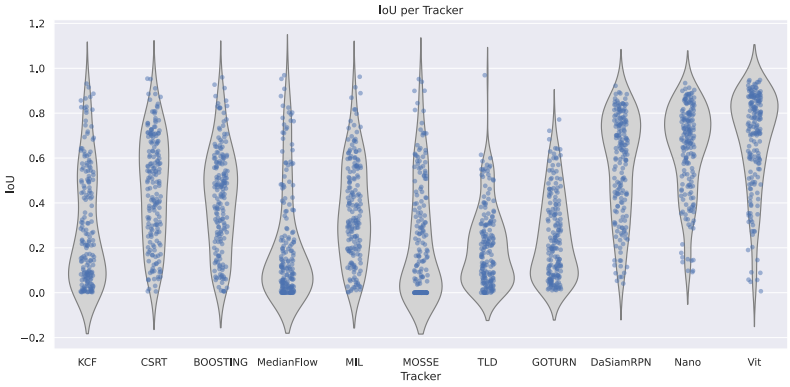

Figure 3. Mean IoU per video and tracker.

Tracker	Failures/video
Boosting	0.0
MIL	0.0
GOTURN	0.0
DaSiamRPN	0.0
Nano	0.0
ViT	0.0
TLD	4.91
CSRT	7.43
MOSSE	44.10
MedianFlow	44.52
KCF	**51.56**

Table 3. Average number of failures per video.

Cross-validate and/or fine-tune the parameters and models on the GOT-10k training set to find the best performing set of hyperparameters for each tracker and then evaluate them.

Include more advanced metrics to extract complex insights about each tracker, such as the tracking length or the failure rate that reinitializes the tracker when fails.

6. References

[1] A. Yilmaz, O. Javed, and M. Shah, 'Object tracking: A survey', *ACM Comput. Surv.*, vol. 38, no. 4, pp. 13-es, Dec. 2006, doi: 10.1145/1177352.1177355.

[2] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, 'Video object tracking using adaptive Kalman filter', *J. Vis. Commun. Image Represent.*, vol. 17, no. 6, pp. 1190–1208, Dec. 2006, doi: 10.1016/j.jvcir.2006.03.004.

[3] P. Janku, K. Koplik, T. Dulik, and I. Szabo, 'Comparison of tracking algorithms implemented in OpenCV', *MATEC Web Conf.*, vol. 76, p. 04031, 2016, doi: 10.1051/matecconf/20167604031.

[4] Z. Soleimanitaleb and M. A. Keyvanrad, 'Single Object Tracking: A Survey of Methods, Datasets, and Evaluation Metrics'.

[5] 'Home', OpenCV. Accessed: May 18, 2024. [Online]. Available: https://opencv.org/

[6] 'Single Object Trackers in OpenCV: A Benchmark | IEEE Conference Publication | IEEE Xplore'. Accessed: May 12, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9194647

[7] K. Ullah, I. Ahmed, M. Ahmad, and I. Khan, 'Comparison of Person Tracking Algorithms Using Overhead View Implemented in OpenCV', in *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, Mar. 2019, pp. 284–289. doi: 10.1109/IEMECONX.2019.8877025.

[8] S. P. Singh and E. Al, 'Comparing Various Tracking Algorithms In OpenCV', *Turk. J. Comput. Math. Educ. TURCOMAT*, vol. 12, no. 6, Art. no. 6, Apr. 2021, doi: 10.17762/turcomat.v12i6.8772.

[9] L. Huang, X. Zhao, and K. Huang, 'GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1562–1577, May 2021, doi: 10.1109/TPAMI.2019.2957464.

[10] 'rafardzp/Comparing-OpenCV-s-Object-Tracking-Algorithms', GitHub. Accessed: May 19, 2024. [Online]. Available: https://github.com/rafardzp/Comparing-OpenCV-s-Object-Tracking-Algorithms