Single Object Trackers in OpenCV: A Benchmark

Adnan Brdjanin, Nadja Dardagan,
Department of Informatics
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67, 9020 Klagenfurt, Austria
Email: {adnanbr, n1dardagan}@edu.aau.at

Dzemil Dzigal, Amila Akagic
Department of Computer Science and Informatics
Faculty of Electrical Engineering, University of Sarajevo
Sarajevo, Bosnia and Herzegovina
Email: {ddzigal1, aakagic}@etf.unsa.ba

***Abstract*—Object tracking is one of the fundamental tasks in computer vision. It is used almost everywhere: human-computer interaction, video surveillance, medical treatments, robotics, smart cars, etc. Many object tracking methods have been published in recent scientific publications. However, many questions still remain unanswered, such as, which object tracking method to choose for a particular application considering some specific characteristics of video content or which method will perform the best (quality-wise) and which one will have the best performance? In this paper, we provide some insights into how to choose an object tracking method from the widespread OpenCV library. We provide benchmarking results on the OTB-100 dataset by evaluating the eight trackers from the OpenCV library. We use two evaluation methods to evaluate the robustness of each algorithm: OPE and SPE combined with Precision and Success Plot.**

***Index Terms*—Single Object Tracking, Benchmark, OpenCV, Computer Vision, Evaluation**

I. Introduction

Computer vision is a rapidly growing interdisciplinary scientific field devoted to analyzing, modifying, and high-level understanding of images. It has been a subject of increasing interest for the last two decades. Its popularity stems from the fact that it provides the means to *see* as humans do, and in some applications it can even outperform humans [1] [2]. It is intensively used to automate tasks traditionally performed by humans. The computer vision system usually consists of multiple stages of processing, such as application of some simple filters, extractions of objects, analysis of data from extracted objects, data communication, and comparison with an existing pattern. The analysis of multi-dimensional data is usually required, and as such, it is much more complex than the analysis of other forms of binary information.

Computer vision is divided into many sub-domains, including object recognition, 3D pose estimation, learning, indexing, and motion estimation just to name a few. One of the fundamental tasks in computer vision is visual object tracking. Tracking is used when application requires some degree of reasoning about the object of interest [3]. It has a long history dating back several decades [4] [5] [6]. Today, it is used in many computer vision systems, such as human-computer interaction, autonomous vehicles, robotics, dynamic behavioral recognition, medical treatments, unmanned aerial vehicle video analysis, military navigation, video indexing, surveillance, and security, etc.

In the last decade, the amount of video data created by people, devices and businesses has increased dramatically. Just in 2019, Youtube usage more than tripled when compared to a period between 2014-2016. In each minute users upload 500h of new video content, while users watch more than 4M videos at the same time. We have also seen a world-wide rise in raw video data from various types of surveillance cameras. Usually, the raw data needs to be processed to extract useful information, such as movements, gestures, identities and even emotions. In this process, visual object tracking is used to estimate the object (target) state over time. Apart from surveillance, analysis of video content is used for security of public spaces, monitoring and security in the residential apartments and industries, military and defense applications, commercial and enterprise facilities and government infrastructures.

In this paper, we provide an overview and benchmark of a single object tracking algorithms available in Open Source Computer Vision (OpenCV) library. The OpenCV is chosen because of its versatility and simplicity of use. Even though it is widespread, the proper benchmarking of object tracking algorithms has not been available up to this day. We focus on the single object tracking, in which an object is being tracked even if the environment consists of multiple objects. We overview and evaluate the eight algorithms available in OpenCV, and provide some recommendations for choosing the object tracker. The original source and binary codes of all algorithms are publicly available, and we provide enough technical details and parameter settings to re-evaluate our findings.

The paper is organized as follows: In Section II, we review state-of-the-art benchmarks. Section III provides a high-level overview of OpenCV and Object tracking. We also provide some details about the object tracking algorithms available in OpenCV. Section IV further details the evaluation methodology used to benchmark object tracking algorithms. Details about the dataset used to evaluate the algorithms are provided in Section V. Section VI summarizes the results of experiments performed on the benchmarked algorithms. Concluding remarks and recommendations are given in Section VII.

II. Related Work

Due to their importance in everyday applications, many object trackers have been proposed in recent literature. There are also many benchmarks which try to provide valuable answers to common questions. Unfortunately, in many Computer Vision sub-domains there is no standardization on how to evaluate the algorithm and define the dataset (and which attributes should a dataset have?) in order to provide a fair comparison of trackers.

There are a number of object tracking algorithms surveys, such as [5] [7] [8] [9]. In [10], benchmarks are classified into two types based on the annotations density: 1) dense and 2) other benchmarks. In dense benchmarks, object in each frame is manually labeled with careful inspection, which is important when either training or assessing trackers. In other benchmarks, sequences are sparsely and/or semi-automatically annotated (i.e. each frame might not be annotated). Representatives of dense benchmarks include (sorted by the publication year) OTB [11] [12], TC-128 [13], VOT [14], NUS-PRO [15], UAV [16], NfS [17], GOT-10k [18] and LaSOT [10], while representatives of other benchmarks include ALOV [8], TrackingNet [19] and OxUvA [20].

Dense benchmarks are more valuable because they provide much more details about the object being tracked, hence better assessment of existing trackers can be made. Existing benchmarks vary in the number of attributes and the range of its values. The benchmarks are usually represented by a set of parameters. An example of parameters include (minimum to maximum value values from referenced dense benchmarks are provided in parentheses): a) number of videos (from 20 to 10,000 videos), b) number of frames in a video (from 41 to 20,665 frames per video), c) total number of frames in a dataset (from 10K to 3.52M), d) total duration of videos (from 5.7 to 62.5 min), e) frame rate, f) object classes, g) class balance, h) number of attributes and others.

III. Overview of OpenCV and Object tracking

A. OpenCV

The Open Source Computer Vision (OpenCV) library is an open source cross-platform computer vision and machine learning software library. It was originally developed by Intel to advance CPU-intensive applications in 2000. Some of its initial goals still hold today, such as providing optimized code for basic computer vision infrastructure, disseminating knowledge to build applications faster, and providing portable, performance-optimized code. The library is licensed with open-source BSD license, and can be used for academic and commercial applications. Today it has around 2,500 optimized algorithms [21] used to detect and recognize human faces, identify various objects, classify human actions in video, track moving objects, extract 3D models of objects, etc. The latest stable release was published in April 2020 as 4.3.0 version.

B. Object tracking

The goal of object tracking is to estimate the state of the selected object in the subsequent frames [12]. The object being tracked is usually marked using a rectangle to indicate its location in the starting[1] frame. When there are no changes in the environment, object tracking is not overly complex, but this is rarely the case. Various disturbances are a normal occurrence in the real world. These disturbances might include occlusion, variations in illumination, change of viewpoint, rotation, blurring due to motion, etc. The task of designing a robust and efficient tracker is known to be a very challenging task.

[1]Please note that the starting frame does not have to be the first frame in a video sequence.

C. Object tracking algorithms in OpenCV

The OpenCV library includes eight algorithms for object tracking, which are available through OpenCV tracking API. Table I provides some information about the available algorithms in the OpenCV library with their publication years and reference to research papers detailing their implementation.

In general, tracking an object in the video involves steps such as: a) choosing the tracker, b) selecting the object (target) from the starting frame with the bounding box, c) initializing the tracker with information about the frame and bounding box, and d) reading the remaining frames and finding the new bounding box of the object. The last step is usually implemented in the loop.

An OpenCV tracker consists of three main components, which also coincide with the components in a typical tracking algorithm [22] (indicated in parentheses):

1) TrackerFeatureSet (the model of the target object's visual appearance): used to represent objects of interests. In OpenCV, possible features can be extracted with HAAR, HOG, LBP, Feature2D, etc. Many other global and local features exist in literature.
2) TrackerSamplerAlgorithm (the mechanism for matching model parts to image regions at each frame): computes the patches over the frame based on the last target location.
3) TrackerModel (the mechanism for continuously relearning or updating models of targets which change their appearance over time): internal representation of the target. It stores all state candidates and computes the trajectory.

TrackerFeatureSet and TrackerSamplerAlgorithm are the visual representation of the target, while the TrackerModel represents the statistical model.

IV. Evaluation Methodology

Evaluation metric ensures the valuable feedback about the algorithm being evaluated, hence choosing the right metric is crucial in the evaluation process. In this section, we provide a rationale behind choosing the evaluation method.

As discussed in [11] [12], there are two **evaluation metrics** to measure the performance of a tracking algorithm: A) Precision Plot and B) Success Plot. The Precision plot shows *how well* the tracker found the object, while the Success Plot shows *whether the tracker found a required object* in the frame. The average precision or success rate is obtained by executing a tracker on a test sequence, where the starting frame is selected from the ground truth position. Evaluation of algorithm's robustness under different conditions can be measured with the **evaluation method**. There are several evaluation methods, as discussed in C) Robustness evaluation.

A. Precision Plot

The precision plot evaluation metric is based on an average Euclidean distance between center locations of the tracked object and the manually labeled ground truth of all the frames in the test sequence. It represents how far the tracker drifts away from the actual target. One major issue with this evaluation metric is that it does not consider the difference in size between the ground truth and the tracked bounding

No	Tracker Full Name (Abbreviation)	Publication Title and Reference	Publication Year (Google Scholar Citations)
1.	**Boosting**	Real-time tracking via on-line boosting [23]	2006 (1432)
2.	Multiple Instance Learning (**MIL**)	Visual tracking with online multiple instance learning [24]	2009 (2095)
3.	**MedianFlow**	Forward-backward error: Automatic detection of tracking failures [25]	2010 (802)
4.	Minimum Output Sum of Squared Error (**MOSSE**)	Visual object tracking using adaptive correlation filters [26]	2010 (1839)
5.	Tracking Learning Detection (**TLD**)	Tracking-learning-detection [27]	2011 (3275)
6.	Kernelized Correlation Filter (**KCF**)	High-speed tracking with kernelized correlation filters [28]	2014 (3131)
7.	**GOTURN** (Generic Object Tracking Using Regression Networks)	Learning to track at 100 fps with deep regression networks [29]	2016 (648)
8.	**CSRT** (Channel and Spatial Reliability Tracker)	Discriminative Correlation Filter with Channel and Spatial Reliability [30]	2017 (444)

TABLE I: OpenCV single object rackers sorted by the year of their publication. Google Scholar Citations are accessed on April 21th 2020.

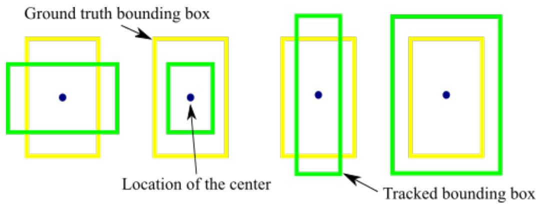


Fig. 1: Precision Plot does not consider the difference in size between the ground truth and the tracked bounding box. Many other variations are possible which are not shown in the illustrated graph.

box (some examples are illustrated in Fig. 1, but many other possibilities also exist). This is why we also use the success plot metric. It should be noted that when the tracker fails, the output location is not shown, as it does not represent the actual state of the tracker (object in the frame). This is why we introduced thresholding in our measurements, as mentioned in [11]. The score for the threshold is up to 50px.

B. Success Plot

The success plot evaluation metric is based on the bounding box overlap. Given the ground truth bounding box bb_{gt} and the tracked bounding box bb_{tr}, the overlap is defined as:

$$S = \frac{|bb_{gt} \cap bb_{tr}|}{|bb_{gt} \cup bb_{tr}|},$$

where $|\cdot|$ represents number of pixels, and $\cap$ and $\cup$ represent intersection and union operators, respectively.

The overlap score of S is calculated for each frame, and it measures the algorithm's success while tracking an object. If S is within a certain threshold of t, it means that the algorithm successfully tracked the target object (i.e. $t = 0.5$).

There are two ways to measure the performance of an algorithm. One way is to calculate the **average overlap score** (AOS), which measures whether bounding boxes in all frames are within a certain threshold (0 to 100 % scaling of the ground truth bounding box). And the other way is to find the **area under curve** (AUC) of each success plot, which shows the tracker's overall performance. AUC is calculated as the average success rate corresponding to sampled overlap thresholds. In practice, success plots are preferred evaluation metric, because they do not ignore the size of the bounding box and the overlap.

C. Robustness evaluation

Robustness evaluation is an evaluation method which shows how *stiff* the tracker is. It tries to answer the following question: can the algorithm track a certain object better if the initial bounding box is smaller or bigger than required? The trackers are evaluated by running them on a test sequence. The tracking process starts from one initial bounding box from the starting frame. The evaluation method reports the average precision and success rate of each tracker.

There are three methods to perform algorithm's robustness evaluation:

1) **One Pass Evaluation** (OPE): tracks the object with a standardized initial bounding box (as big as possible but no extra space around sides of the object) from the ground truth position in the starting frame. In this method, earlier part of a sequence affects OPE score more than the later part. This is the simplest method, because initialization of the frame is required only in the starting frame.
2) **Spatial Robustness Evaluation** (SRE): tracks the object with 8 different kinds of scaling (spatially shifted and scaled initializations). The initial bounding box is scaled in the following directions: left, right, up, down and combinations of all of them.
3) **Temporal Robustness Evaluation** (TRE): tracks the object from a different starting frame in a test sequence. The object in the frame is initialized from the corresponding ground-truth object until the end of an image sequence.

The OPE evaluation does not show tracker's sensitivity to different types of initialization in the starting frame, and different initial states of the frame. One other aspect is also very important to analyze: what happens when tracking fails (object is not found in the frame anymore) and how to re-initialize the bounding box? Initially, SRE and TRE methods were proposed in [12] to tackle these issues.

In this paper, we evaluate the following algorithms from the OpenCV library: 1) Boosting, 2) MIL, 3) MedianFlow, 4) MOSSE, 5) TLD, 6) KCF, 7) GOTURN and 8) CSRT. We use two evaluation methods to evaluate robustness of each algorithm: OPE and SPE combined with Precision and Success Plot.

V. Evaluation Dataset

This section reviews the dataset used to evaluate OpenCV single object tracking algorithms. The dataset used for evaluation is publicly available **OTB-100**[2] (Object Tracking Benchmark) [11]. It consists of 98 videos divided into 11 classes (attributes) which are shown in Table II. In the dataset, the minimum number of frames in a sequence is 71 frames, maximum is 3,872 frames, and the average number of frames is 590. The total number of frames is 59K, and the total duration is 32.8 minutes. Frame rate is 30fps. Videos in the dataset vary in resolution, with lowest resolution being 128×96 and highest 800×336.

Many datasets for object tracking are available today. Detailed review is provided in [10]. We decided to use OTB-100 because it is properly labeled, documented and contains class decomposition we want to test OpenCV trackers on. Each tracker was evaluated through all videos of each class with respect to success and precision rate. Success rate is measured with threshold values from 0 to 1, which represents strictness criteria for some trackers. Precision rate is measured through pixel threshold value from 0 to 50 pixels that represents center distance (CD) from ground truth center to tracked bounding box center. Looking at certain threshold value might not be representative, so we use the AUC score in order to rank algorithms. It is important to note that we did not exclude any tracker failures from our results.

IV	Illumination Variation - the illumination in the target region is significantly changed.
SV	Scale Variation - the ratio of the bounding boxes of the first frame and the current frame is out of the range $[1/t_s, t_s], t_s > 1 (t_s = 2)$
OCC	Occlusion - the target is partially or fully occluded.
DEF	Deformation - non-rigid object deformation.
MB	Motion Blur - the target region is blurred due to the motion of target or camera.
FM	Fast Motion - the motion of the ground truth is larger than t_m pixels $(t_m = 20)$.
IPR	In-Plane Rotation - the target rotates in the image plane.
OPR	Out-of-Plane Rotation - the target rotates out of the image plane.
OV	Out-of-View - some portion of the target leaves the view.
BC	Background Clutters - the background near the target has the similar color or texture as the target.
LR	Low Resolution - the number of pixels inside the ground-truth bounding box is less than t_r $(t_r = 400)$.

TABLE II: Classes of videos in the TB-100 Dataset. Each sequence in the dataset is categorized with 11 attributes shown in this table. Threshold value for some attributes is also shown in the table. One video can belong to multiple classes.

VI. Evaluation Results

For each evaluated tracker, we ran a series of tests on a Linux based machine with i7-8565U CPU @ 1.80GHz with four cores. All tests were successfully executed on all videos except GOTURN. This tracker could not be executed on 27 videos out of 98 and exhibited very unstable behavior in various tests. An example is shown in Fig. 2. At the beginning of the test, the tracker runs normally with high FPS comparable with other trackers. However, the tracker loses the object due to various reasons related to the object's behavior. In such a case, the search region increases, and the complexity of the network for offline training also increases. The results is extensive usage of memory (RAM, then VM) which causes the process to be killed by the Out Of Memory (OOM) manager.

The majority of videos on which tracker could not be tested belong to classes such as IV, SV, OCC, DEF, OPR, and BC (see Table II). GOTURN tracker was also very sensitive when running SPE evaluation metric, where the bounding box of the object was scaled eight times (i.e. test could be executed in one scaling configuration, but it would fail in another). Due to these reasons, it was not possible to measure GOTURN performance results on all videos, hence we exclude the GOTURN tracker from our results. However, GOTURN might be used with slow-moving objects outside before mentioned classes. In [29], GOTURN authors suggest that their algorithm could be combined with other approaches to track fast-moving objects, which might be a solution to this problem.

Evaluation results for remaining 7 trackers are presented as average value for OPE and SRE evaluation. We first analyze the OPE plots because the evaluation method is simple and shows *raw* results (Fig. 3). The OPE precision plot in Fig. 3a. shows that KCF tracker drifts the least pixel distance on average from ground truth bounding box with average value of 0.735. It shows the best results on all video classes except for classes *Low Resolution* and *Motion Blur*, where CSRT tracker gets the best results. KCF stands out the most in classes *In Plane Rotation*, *Out of Plane rotation* and *Occlusion* with the best results among all trackers.

The OPE success plot in Fig. 3b. shows that CSRT has the largest average bounding box overlap with the value of 0.292. The second best algorithm in regards to success evaluation is Boosting with the value of 0.242. CSRT has the best evaluation looking class-by-class with exceptionally good results in classes *Background Clutter*, *Fast Motion*, *Low resolution* and *Scale Variation*[3].

The SRE evaluation method results are shown in Fig. 4. In SRE precision plot in Fig. 4a, the MedianFlow tracker has the best results with an average of 0.632, which is closely followed by KCF with 0.617. However, in the SRE success plot in Fig. 4b, all algorithms showed similar results with minor evaluation differences. The KCF is the best tracker with evaluation grade of 0.37, which is closely followed by MedianFlow with 0.365. It is worth mentioning that if it is required to have a bounding box as small as possible, but covering the whole object, then the OPE evaluation is more important. However, that should not diminish the importance of the SRE evaluation as it represents realistic values when the bounding box is not ideally set to have minimum area whilst covering the whole object.

The average success rate based on the bounding box overlap per class and average success rate for the entire OTB-100 dataset are shown in Fig. 5a and Fig. 5b, while the average FPS per class and average FPS for the entire OTB-100 dataset

[2]More information about the dataset is available at http://cvlab.hanyang.ac.kr/tracker_benchmark/benchmark_v10.html

[3]For detailed results and graphs visit https://github.com/adnanb97/OpenCV-Research-Benchmarking

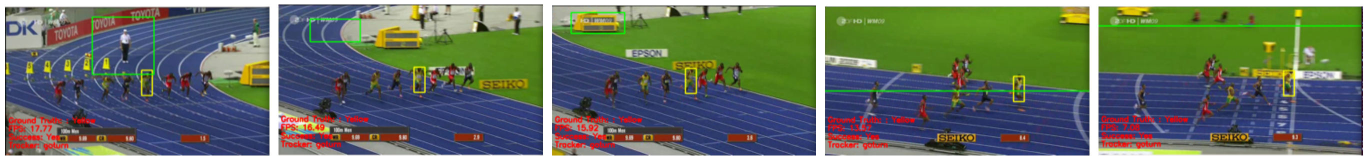
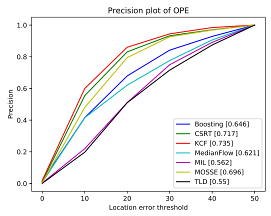
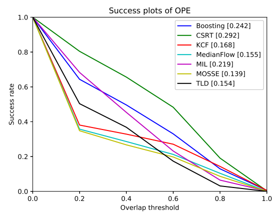


Fig. 2: Test sequence from “Bolt.avi” (OTB-100) with GOTURN tracker. The object ground truth is displayed in yellow, and object tracked by the tracker is in green. Notice the increase of green bounding box size and simultaneous decrease of FPS. **Best viewed in color.**

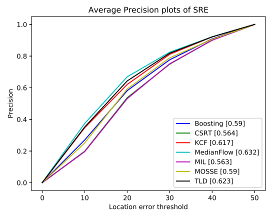


(a) OPE Success Plot

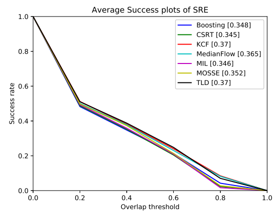


(b) OPE Success Plot

Fig. 3: AUC OPE algorithm visualization



(a) SRE Precision Plot Average



(b) SRE Success Plot Average

Fig. 4: AUC SRE algorithm visualization

are shown in Fig. 5c and Fig. 5d. From (a) and (c), it is easy to choose the tracker based on the performance of the presented category. In terms of success rate, the overall best results are achieved with Boosting and MIL, which are closely followed by TLD and CSRT. In terms of speed (FPS), the MOSSE is the best overall tracker, but it should be noted that its success rate is very low. The second best is MedianFlow with relatively high success rate, so this would be a better choice. All trackers show significantly better results in the *Low Resolution* category, which is expected. In category *Motion Blur*, all trackers performed poorer and this resulted in poorer FPS. Except for MIL, all trackers are capable of real-time or near real-time processing (please note that these are the average FPS values, so for a high resolution video, this might not be the case).

The average pixel center distance from the ground truth and the tracker bounding boxes are shown in Fig. 6a (lower is better). The overall best tracker is KCF, while the second best is CSRT. The average IoU rate for each tracker is shown in Fig. 5c (higher is better). The best tracker in this group is CSRT, followed by MIL.

VII. Conclusion

In this paper, we evaluate single object trackers available in the OpenCV library. We perform series of tests on each tracker and provide some recommendations on how to choose the object tracker based on general characteristics of a dataset which should be helpful when choosing the tracker for a practical application. The evaluation was performed on OTB-100 dataset with two evaluation methods, OPE and SPE.
In summary, we can sort results in three categories: success rate, speed and precision. In terms of success rate, Boosting and MIL trackers performed the best, while MOSSE and MedianFlow were the fastest. In terms of precision (pixel distance between centers) and IoU rate, CSRT tracker performed the best overall.

References

[1] J. J. Titano, M. Badgeley, J. Schefflein, M. Pain, A. Su, M. Cai, N. Swinburne, J. Zech, J. Kim, J. Bederson *et al.*, “Automated deep-neural-network surveillance of cranial images for acute neurologic events,” *Nature medicine*, vol. 24, no. 9, pp. 1337–1341, 2018. 1

[2] S. Stabinger, A. Rodríguez-Sánchez, and J. Piater, “25 years of cnns: Can we compare to human abstraction capabilities?” in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 380–387. 1

[3] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, “Fast online object tracking and segmentation: A unifying approach,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 1328–1338. 1

[4] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, “Video object tracking using adaptive kalman filter,” *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006. 1

[5] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *Acm computing surveys (CSUR)*, vol. 38, no. 4, pp. 13–es, 2006. 1

[6] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003. 1

[7] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, “A survey of appearance models in visual object tracking,” *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, pp. 1–48, 2013. 1

[8] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2013. 1, 2

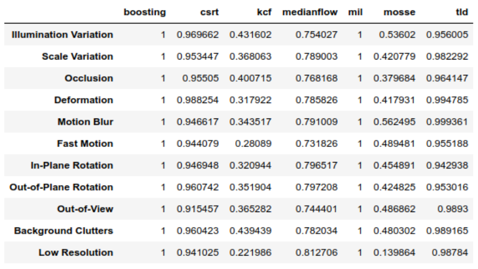
[9] P. Li, D. Wang, L. Wang, and H. Lu, “Deep visual tracking: Review and experimental comparison,” *Pattern Recognition*, vol. 76, pp. 323–338, 2018. 1

[10] 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 Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5374–5383. 1, 2, 4

[11] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418. 2, 3, 4

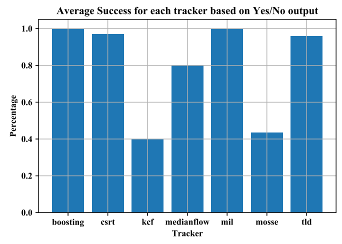
[12] Y. Wu, L. Jongwoo, and M.-H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015. 2, 3

[13] P. Liang, E. Blasch, and H. Ling, “Encoding color information for visual tracking: Algorithms and benchmark,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5630–5644, 2015. 2

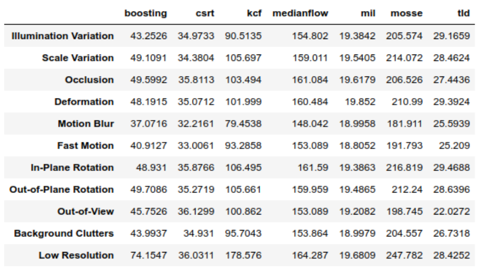


	boosting	csrt	kcf	medianflow	mil	mosse	tld
Illumination Variation	1	0.969662	0.431602	0.754027	1	0.53602	0.956005
Scale Variation	1	0.953447	0.368063	0.789003	1	0.420779	0.982292
Occlusion	1	0.95505	0.400715	0.768168	1	0.379684	0.964147
Deformation	1	0.988254	0.317922	0.785826	1	0.417931	0.994785
Motion Blur	1	0.946617	0.343517	0.791009	1	0.562495	0.999361
Fast Motion	1	0.944079	0.28089	0.731826	1	0.489481	0.955188
In-Plane Rotation	1	0.946948	0.320944	0.796517	1	0.454891	0.942938
Out-of-Plane Rotation	1	0.960742	0.351904	0.797208	1	0.424825	0.953016
Out-of-View	1	0.915457	0.365282	0.744401	1	0.486862	0.9893
Background Clutters	1	0.960423	0.439439	0.782034	1	0.480302	0.989165
Low Resolution	1	0.941025	0.221986	0.812706	1	0.139864	0.98784

(a) Average Success Rate per Class

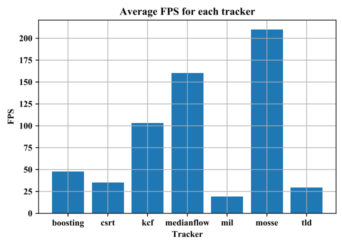


(b) Average Success Rate



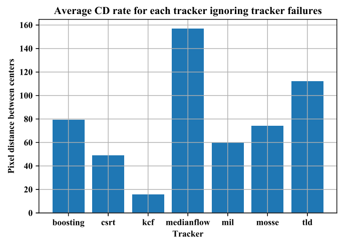
	boosting	csrt	kcf	medianflow	mil	mosse	tld
Illumination Variation	43.2526	34.9733	90.5135	154.802	19.3842	205.574	29.1659
Scale Variation	49.1091	34.3804	105.697	159.011	19.5405	214.072	28.4624
Occlusion	49.5992	35.8113	103.494	161.084	19.6179	206.526	27.4436
Deformation	48.1915	35.0712	101.999	160.484	19.852	210.99	29.3924
Motion Blur	37.0716	32.2161	79.4538	148.042	18.9958	181.911	25.5939
Fast Motion	40.9127	33.0061	93.2858	153.089	18.8052	191.793	25.209
In-Plane Rotation	48.931	35.8766	106.495	161.59	19.3863	216.819	29.4688
Out-of-Plane Rotation	49.7086	35.2719	105.661	159.959	19.4865	212.24	28.6396
Out-of-View	45.7526	36.1299	100.862	153.089	19.2082	198.745	22.0272
Background Clutters	43.9937	34.931	95.7043	153.864	18.9979	204.557	26.7318
Low Resolution	74.1547	36.0311	178.576	164.287	19.6809	247.782	28.4252

(c) Average FPS per Class

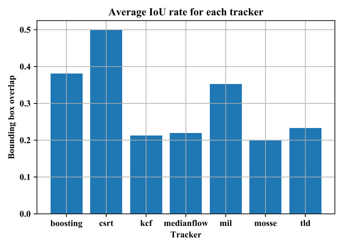


(d) Average FPS

Fig. 5: Average Success Rate and FPS per Class and in total for OpenCV trackers.



(a) Average Center Distrance



(b) Average IoU rate

Fig. 6: Average Center Distance (CD) nad Intersection over Unit (IoU)

[14] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2137–2155, 2016. 2

[15] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan, "Nus-pro: A new visual tracking challenge," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 335–349, 2015. 2

[16] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European conference on computer vision*. Springer, 2016, pp. 445–461. 2

[17] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1125–1134. 2

[18] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[19] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 300–317. 2

[20] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. Smeulders, P. H. Torr, and E. Gavves, "Long-term tracking in the wild: A benchmark," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685. 2

[21] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to opencv," in *2012 proceedings of the 35th international convention MIPRO*. IEEE, 2012, pp. 1725–1730. 2

[22] J. Xiao, R. Stolkin, and A. Leonardis, "Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4978–4987. 2

[23] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Bmvc*, vol. 1, no. 5, 2006, p. 6. 3

[24] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *2009 IEEE Conference on computer vision and Pattern Recognition*. IEEE, 2009, pp. 983–990. 3

[25] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 2756–2759. 3

[26] 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, pp. 2544–2550. 3

[27] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2011. 3

[28] 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*, vol. 37, no. 3, pp. 583–596, 2014. 3

[29] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 749–765. 3, 4

[30] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and 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. 3