METHODS TO ROBUST RANKING OF OBJECT TRACKERS AND TO TRACKER DRIFT CORRECTION

Julien Valognes

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science in Electrical and Computer Engineering

Concordia University

Montréal, Québec, Canada

August 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Julien Valognes**

Entitled: **Methods to Robust Ranking of Object Trackers and to Tracker Drift Correction**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Wei-Ping Zhu _____ Chair

Dr. Thomas Fevens _____ External Examiner

Dr. Wei-Ping Zhu _____ Examiner

Dr. Maria A. Amer _____ Supervisor

Approved by _____

Dr. Wei-Ping Zhu, Graduate Program Director

_____ 2020 _____

Dr. Mourad Debbabi, Interim Dean

Faculty of Engineering and Computer Science

Abstract

Methods to Robust Ranking of Object Trackers and to Tracker Drift Correction

Julien Valognes

This thesis explores two topics in video object tracking: (1) performance evaluation of tracking techniques, and (2) tracker drift detection and correction. Tracking performance evaluation consists into comparing a set of trackers' performance measures and ranking these trackers based on those measures. This is often done by computing performance averages over a video sequence and then over the entire test video dataset, consequently resulting in an important loss of statistical information of performance between frames of a video sequence and between the video sequences themselves. This work proposes two methods to evaluate trackers with respect to each other. The first method applies the median absolute deviation (MAD) to effectively analyze the similarities between trackers and iteratively ranks them into groups of similar performances. The second method gains inspiration from the use of robust error norms in anisotropic diffusion for image denoising to perform grouping and ranking of trackers. A total of 20 trackers are scored and ranked across four different benchmarks, and experimental results show that using our scoring evaluation is more robust than using the average over averages.

In the second topic, we explore methods to the detection and correction of tracker drift. Drift detection refers to methods that detect if a tracker is about to drift or has drifted away while following a target object. Drift detection triggers a drift correction mechanism which updates the tracker's rectangular output bounding box. Most drift

detection and correction algorithms are called while the target model is updating and are, thus, tracker-dependent. This work proposes a tracker-independent drift detection and correction method. For drift detection, we use a combination of saliency and objectness features to evaluate the likelihood an object exists inside a tracker's output. Once drift is detected, we run a region proposal network to reinitialize the bounding box output around the target object. Our implementation applied on two state-of-the-art trackers show that our method improves overall tracker performance measures when tested on three benchmarks.

Acknowledgments

I wish to express my most sincere gratitude to my supervisor, Dr. Maria A. Amer, for her invaluable help in my research, her guidance, and her unwavering patience. This thesis would not have existed in this form had she not believed in me and in my ability to succeed. I have learned and grown significantly from working under her supervision.

I would like to extend my thanks to the members of the VidPro research group for their friendly cooperation and for making the research environment enjoyable.

I also wish to acknowledge my mother, my partner, my in-laws, and my closest friends for their love and their unparalleled support throughout the course of my studies.

Contents

List of Figures

List of Tables

Glossary of Acronyms

AOR	Average Overlap Ratio
BB	Bounding Box
CF	Correlation Filter
CNN	Convolutional Neural Network
DC	Drift Correction
DD	Drift Detection
EAO	Expected Average Overlap
EB	Edge Box
FPS	Frames Per Second
FR	Failure Rate
HVS	Human Visual System
IoU	Intersection over Union
KDE	Kernel Density Estimation
LSM	Longest Subsequence Measure
MAD	Median Absolute Deviation
Pr	Precision
RPN	Region Proposal Network
RP	Region Proposal
Re	Recall
SRE	Spatial Robustness Evaluation

SR Success Rate

TNR True Negative Rate

TPR True Positive Rate

TRE Temporal Robustness Evaluation

Chapter 1

Introduction

1.1 Motivation

Visual object tracking has been a growing field of research as it plays a fundamental role in many applications such as activity recognition (e.g., elderly health care or athletes performance measurement), video surveillance, human-computer interactions, augmented reality, and robot navigation. Ideally, object tracking techniques aim to follow objects similarly to how the human visual system (HVS) would [14]. Given the initial location of a target object (in the form of a rectangular bounding box) in a video sequence's first frame, an object tracker aims to estimate the position of the object in the next frames. While the HVS is well equipped to memorize shapes and anticipate movements in unconstrained environments, developing a robust tracking method which can behave similarly to the HVS remains a challenge due to numerous object-related (scale variation, deformation, motion blur, and fast motion) and environment-related (illumination variation, partial and full occlusion, and background clutter) attributes [48, 91, 55]. Such attributes can lead trackers to show inaccuracies, drift, and potentially fail [17, 18]. Numerous object tracking techniques are being continuously presented in the literature to address these challenges

and one of the central questions is how to evaluate their performance with respect to the state-of-the-art.

In this thesis, we have two main objectives: first, to investigate methods that score and rank trackers and account for how trackers perform with respect to each other; secondly, to investigate methods that detect tracker drift independently of how a baseline tracker is designed. For our first objective, the motivation comes from a recurring problem in object tracking evaluation protocols, which is the evaluation's reliance on the average as a measure of central tendency for estimating a tracker's performances with respect to other trackers [43, 44, 45, 46, 47, 48, 91, 90, 30, 64, 82, 77]. For our second objective, our motivation comes from the increasing research interest in long-term video tracking challenges, such as drift, failure, or object disappearance [48, 63, 64, 82, 52, 100, 50, 81, 73].

1.2 Problem Statement

Research in video object tracking progresses at such a high rate that many designs are proposed on a yearly basis to compete with the state-of-the-art [48, 63]. One issue that comes with this rapid growth is the difficulty to compare and evaluate as objectively as possible the differences in performances between trackers. Since most rankings are based on averaging averages of performance measures, it has become increasingly difficult to argue which is best between a tracker that performs well in certain sequences but poorly in other sequences, a tracker that fluctuates a lot in-between sequences, and a tracker that performs consistently but slightly less well on average. While ranking according to an average performance measure allows to numerically order trackers with respect to each other, minor differences in performance (for example an average difference of 0.03) should not justify alone such ranking. Therefore, it is useful to present a method for scoring and ranking of trackers using

a robust estimator against outliers.

To this day, developing a tracking algorithm that is robust remains a challenge. Due to appearance changes caused by illumination variation, object deformation, and dynamic motion, it is a complex task for an object tracker to consistently estimate the position of its target object without drifting away. To correct errors caused by tracker drift and prevent a tracker from failing, integrating methods that aim to propose regions in an image (called region proposal networks) may help improve a tracker's robustness.

1.3 Summary of Contributions

The two main objectives of this thesis are: (1) proposing a scoring and ranking method for evaluation of video object trackers taking into account variations of performance across video frames and across test video sequences, and (2) handling detection and correction of tracker drift by integrating modern region proposal networks .

For our first objective, we divide our contribution into two separate works. Our first work introduces a strategy to effectively determine similarly performing trackers and iteratively rank them by using the median absolute deviation (MAD). Our second work borrows the use of robust error norms in image denoising to propose a robust method for scoring, grouping, and ranking trackers. We show that our scores are more robust to noise and more representative of a tracker's performance than the widely used average of averages. We consider this robust method as our main contribution in this thesis.

For our second objective, we integrate a tracker-independent drift detection method using both saliency and objectness measures and a drift correction strategy that improves the overall robustness of tracking algorithms using region proposal networks.

1.4 Object Tracking: State-of-the-art

Due to numerous and unpredictable challenges present in video sequences (illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter, and low resolution), it is demanding for a tracker to follow any kind of arbitrary target. Over the last decade, object tracking techniques have received lots of attention and have made considerable progress to overcome those challenges.

CNN (Convolutional Neural Network) and CF (Correlation Filter)-based trackers have significantly advanced the field of visual object tracking and are amidst the state-of-the-art [47, 48, 63]. CF-based visual tracking approaches have attracted considerable attention due to being computationally efficient in the Fourier domain and not requiring multiple target appearances. These methods circularly shift versions of the input and regress them to soft class labels. The target object is tracked in the next frame by matching the filter to the search window which yields the highest correlation with the initial object. There exist multiple CF optimization methods to model the input, including sum-of-squared error [8], kernelized correlation filters [40], multiple dimensional features [93], spatio-temporal regularization [53], short-term and long-term memory storage [41], multi-scale estimation [26], CNN-features [54], and patch reliability [57]. CNN-based tracking is another widely applied tracking technique for modeling target appearances on-line. When pre-trained on a large-scale comprehensive dataset, those architectures have shown to carry out significant performance improvements. Discriminative models [65, 38] are off-line pre-trained frameworks which aim to learn a classifier that discriminates a target from its background. Deep regression models [79, 87, 39] more typically integrate CNN features with the discriminative correlation filter framework to predict a set of interdependent values: the bounding box (BB) coordinates in the case of visual tracking. In addition to those two deep-tracking techniques, the Siamese architecture [52, 6, 100, 24, 97]

has started to gain more attention due to its balance between accuracy and speed. It consists of a CNN applied on two streams, processing the input image and an image patch containing the object of interest separately, and cross-correlates them to search for the test image in the next frame. To this day, most research attention goes to Siamese architectures.

1.5 Datasets and Performance Measures

In this section, we introduce performance measures and datasets which are widely used in video object tracking.

1.5.1 Datasets

Widely used publicly available benchmarks are OTB-100 [91] (also denoted OTB), VOT2018-ST [48] (also denoted VOT-ST), NfS-30 [42] (also denoted NfS), TC128 [58], and VOT2018-LT [48] (also denoted VOT-LT). These benchmarks are summarized in Table 2.

Each benchmark is compiled for a different purpose. OTB-100 is the first large benchmark introduced to cover all challenging aspects in visual tracking. VOT2018-ST and VOT2018-LT both come from the 2018 VOT challenge [48] but are differentiated to tackle short-term and long-term video challenges, respectively. Sequences from the NfS dataset are captured at different frame rates, one at 30 frames per second (NfS-30) and the other at 240 frames per second (NfS-240), to test the impact of frame rate on different tracking architectures. TC128 only contains color sequences in order to understand the role color information has on visual tracking.

Often, trackers perform differently for different datasets. To test the generalization ability of a method, datasets can be combined into one dataset, e.g., one can combine the short-term datasets into a single one called the OTB-100+VOT2018-ST+NfS-30

Table 2: Sequence length information for benchmarks used: number of sequences, minimum, maximum, average, and total number of frames.

Benchmark	Sequences	Min	Max	Average	Total
OTB-100	100	71	3872	590	59040
VOT2018-ST	60	41	1500	356	21356
NfS-30	100	22	2584	484	48399
TC128	129	71	3872	429	55346
VOT2018-LT	35	1389	29700	4196	146847

dataset.

1.5.2 Performance Measures

To measure performance of object tracking algorithms, we assume there are T trackers $\{t_i, i = 1, \cdots, T\}$ and L test video sequences $v_l, l = 1, \cdots, L\}$ each having N_l frames $\{F_t, t = 1, \cdots, N_l\}$. The objective measures widely used for accuracy and robustness are Average Overlap Ratio (AOR) and Failure Rate (FR), respectively. OTB-100 [91] and VOT2018 [48] benchmarks make use of said measures, but other benchmarks present variations of them, such as Expected Average Overlap (EAO), Longest Subsequence Measure (LSM), and F-measure (F).

1.5.2.1 Average Overlap Ratio

The AOR measures how accurately a tracker BB is placed with respect to the ground truth over a sequence l. Given v_l of N_l frames, it is defined as

$$\text{AOR}_l = \frac{\sum_{t=1}^{N_l} \text{IoU}_{tl}}{N_l}; \quad \text{AOR} = \frac{\sum_{l=1}^{L} \text{AOR}_l}{L}, \tag{1}$$

where AOR is calculated over all L test videos of a dataset, and IoU_{tl} is the ratio of the number of overlapping pixels (intersection) between the output BB and the ground truth BB and the total number of pixels in the ground truth box of frame F_t in v_l.

1.5.2.2 Failure Rate

The FR evaluates the rate at which a tracker completely fails in a video sequence l, i.e., how often $\text{IoU}_{tl} = 0$. Given the Heaviside step function $H(.)$, FR over sequence l can be written as

$$\text{FR}_l = \frac{\sum_{t=1}^{N_l} H(\text{IoU}_{tl})}{N_l}; \quad \text{FR} = \frac{\sum_{l=1}^{L} \text{FR}_l}{L}. \tag{2}$$

This measure may also be used alternatively as the Success Rate SR, where $SR = 1 - FR$, to generate success plots as in [91].

1.5.2.3 Expected Average Overlap

The EAO is an objective measure used in the VOT2018 challenge [49] which combines accuracy values of fragmented sequences to yield an expected AOR value. The VOT protocol re-initializes a tracker after each failure. Any time a tracker fails at frame F_t of a video sequence l, it is reinitialized to the ground truth BB at the subsequent frame F_{t+5}. A sequence v_l is thus turned into $m + 1$ video fragments if it fails m times, $m \geq 1$. For a tracker i, the VOT challenge calculates EAO as follows. AOR is first measured over the entire video fragments on all frame ranges $[1, n]$, where $n = 2, \cdots, N_{max}$ and N_{max} is the maximum length of all tested fragments, meaning evaluated ranges are $[1, 2], [1, 3], \cdots, [1, N_{max}]$. Only fragments that end with a failure are picked; the rest is discarded. If a fragment is too short on $[1, n]$, it is padded with zeros. Otherwise, it is trimmed to fit said range. A EAO curve is then obtained by averaging the AOR over all the fragments for each range. Obtaining the desired EAO output consists in averaging the curve in a specific interval $H = [H_{low}, H_{high}]$. To find H, the authors apply a data smoothing probabilistic Kernel Density Estimation (KDE) model on the dataset $\{v_l\}$. H_{low} and H_{high} correspond to the lengths where the area under the probability density function is 0.5 and where $p(H_{low}) \approx p(H_{high})$ starting from the mode. Finally, given EAO_n the EAO at frame n, the desired EAO

over H is calculated as

$$\text{EAO} = \frac{1}{N_{high} - N_{low} + 1} \sum_{n=N_{low}}^{N_{high}} \text{EAO}_n. \tag{3}$$

In VOT2018-ST, the authors calculate EAO assuming that once a tracker fails, it will keep failing and will not recover. This is not accurate since a tracker may still recover once it has drifted or failed; therefore, we do not use the EAO for evaluation in this chapter.

1.5.2.4 Longest Subsequence Measure

LSM is proposed in [64] to appropriately quantify BB overlap evaluation on long-term tracking rather than short-term. Given the longest successfully tracked subsequence v_λ of a video sequence v_l, it is defined as the ratio of the length N_λ of v_λ to the full length N_l of v_l. For a tracker i, v_λ is a successfully tracked subsequence if $x\%$ of frames F_t in subsequence λ yield $\text{IoU}_{tl} \geq 0.5$. Therefore,

$$\text{LSM}_l = \frac{N_\lambda}{N_l}. \tag{4}$$

The parameter x is a tolerance threshold which the authors set to 95% to allow some tracking failure. This means LSM permits temporary drift or failure to happen but still penalizes cases in which a tracker accidentally recovers a lost target.

1.5.2.5 Precision, Recall, F-measure

These three measures are used in the VOT2018-LT [48] challenge for long-term tracking evaluation. Trackers in that challenge are required to perform re-detection after a target object is lost. In fact, Precision Pr measures the percentage of all the re-detection predictions (e.g., the tracker BBs) that agree with the ground truth, while Recall Re is the percentage of all the ground truths that agree with the predictions. Widely-used in binary classification, the F-measure provides a Pr/Re trade-off and is

calculated as follows

$$F = \frac{2\mathrm{Pr.Re}}{\mathrm{Pr} + \mathrm{Re}}, . \tag{5}$$

1.6 Thesis Outline

In Chapter 2, we propose a similarity-based strategy to robustly evaluate and iteratively rank tested trackers. We employ the median absolute deviation (MAD) to effectively analyze the similarities amidst trackers, place them in groups of similar performance, and rank them to determine which are the top performing trackers.

In Chapter 3, we gain inspiration from image denoising and use the notion of robust error norms in robust statistics to provide a scoring and ranking method that is scientifically solid.

In Chapter 4, we present a drift detection strategy which computes edge-based objectness and superpixel saliency measures on the output bounding box of a tracker. Then, if drift is assumed, a drift correction algorithm relocates the bounding box on the target object using a region proposal network.

In Chapter 5, we summarize our contributions and present future work to conclude the thesis.

Chapter 2

Similarity-Based Scoring and Iterative Ranking of Object Trackers

2.1 Introduction

Each year, numerous tracking techniques are introduced and compete with the state-of-the-art. It is, therefore, important to have a systematic robust ranking strategy for a fair evaluation and comparison of trackers. In the past 20 years, several benchmarks such as CAVIAR [28], CDC [34], FERET [71], iLIDS [3], PETS [2], and MOT [78] became publicly available to encourage research initiatives in video object tracking. Tested trackers are initialized at a single frame of a video sequence with a rectangular-shaped BB and are required to follow an object given that single example. More recent benchmarks [43, 44, 45, 46, 47, 48, 63, 90, 91, 77] also tackle the problem of formulating tracking performances in a manner that is easily quantifiable and as unbiased as possible. The most commonly used indicators for evaluation are tracking accuracy and robustness. However, quantifying accuracy and robustness often requires to measure

averages over a sequence and then over the entire dataset, consequently resulting in an important loss of statistical information. The final result is, thus, not truly indicative of a tracker's performance through time, which makes it difficult to evaluate it relatively to that of its counterparts.

This chapter presents our strategy to robustly evaluate and iteratively rank tested trackers; we use the median absolute deviation (MAD) to effectively analyze the similarities amidst trackers while quantifying tracking performances over four benchmarks.

In the rest of this chapter, section 2.2 presents related tracker ranking work; section 2.3 proposes our ranking method; section 2.4 gives simulation results; and section 2.5 concludes the chapter.

2.2 Related Work

Prior related work can be divided into evaluation and ranking of tracking algorithms. Evaluation consists into comparing a set of trackers' average performance measures [43, 44, 45, 46, 47, 48, 91, 90, 30, 64, 82, 77], while ranking categorizes them based on said evaluation [43, 44, 45, 46, 47, 48, 30, 77, 83, 68].

2.2.1 Evaluation

A variety of datasets are available with their own proposed evaluation methods. Among the most widely-used benchmarks for tracking, OTB-100 [91] consists of 100 short sequences and evaluates trackers using precision and success plots as well as accuracy and robustness measures. Accuracy is measured by the commonly used Average Overlap Ratio (AOR) between a tracker's BB and the ground truth BB over all test videos, whereas robustness is measured using either Temporal Robustness Evaluation (TRE) or Spatial Robustness Evaluation (SRE). For a given sequence,

TRE averages all AORs obtained from running a tracker at different frames of the sequence; the tracker is first initialized at the starting frame, and it is then reset to all the consecutive subsequent frames. To measure SRE, the target BB is shifted by 10% of the target size and scaled from 80% to 120% of the ground truth BB. SRE is thus the AOR average of 12 configurations: four center shifts, four corner shifts, and four scale variations. The more recent VOT2018 competition [48] uses a dataset of 60 short sequences and 35 long sequences and is divided into three challenges: short-term, real-time, and long-term. In the short-term VOT2018-ST challenge, three primary measures are applied for tracking evaluation: accuracy, robustness, and Expected Average Overlap (EAO), and trackers which fail are reset to the ground truth five frames later. Under that protocol, robustness simply consists in the number of failures in a video sequence, accuracy is the widely-used AOR, and EAO, which is introduced to account for variable sequence lengths, estimates the AOR a tracker is expected to yield on a dataset of equally sized short sequences. The real-time challenge works similarly to the short-term one but adds a constraint on tracking speed; tested trackers require to run at a speed equal or greater to 20 frames per second (FPS), and trackers that do not respond in time are penalized by assigning their last reported BB for evaluation. For long-term tracking, however, the VOT2018-LT challenge uses precision Pr, recall Re, and a standard tracking F-score as per [62] to combine Pr with Re and optimize their tradeoff. [77] covers various video tracking challenges in 315 sequence fragments and also combines precision Pr and recall Re using an overlap-based F-score with a 50% Intersection over Union (IoU) criterion to distinguish false positives from true positives.

Most available datasets for tracking evaluation have been tailored for short-term scenarios, which are not representative of real life applications. To address such disparity, researchers have proposed more largely scaled tracking benchmarks. The VOT challenge [48], for instance, had only started introducing long-term tracking

evaluation in 2018. [64] introduces its own TLP dataset of 50 lengthy videos of approximately 13500 frames each on average. The authors also divide this benchmark into two other separate datasets: TinyTLP, which consists of each sequence's first 600 frames to match the OTB-100 dataset average length, and TLPattr, which comprises of 90 short sequences categorized on a challenge basis (fast motion, illumination variation, scale variation, partial occlusion, out-of-view or full occlusion, background clutter). [64] proposes the Longest Subsequence Measure (LSM) objective measure to address a tracker's ability to continuously track a target in lengthy videos and with a certain tolerance for failure. [82] focuses on the problem of target re-detection due to objects not always being present throughout segments of video sequences. The authors evaluate object localization by calculating the geometric mean of the True Positive Rate (TPR) and True Negative Rate (TNR) of the estimated object presence at each frame. This measure differs from the more commonly applied ones for robustness as it evaluates more a tracker's ability to re-detect an object in long sequences rather than its ability to recover from occurring drift.

2.2.2 Ranking

The authors in [43, 44, 45, 46, 47, 48] evaluate all tested trackers based on accuracy and robustness measures and numerically rank them from best to worst. VOT2013 [43] includes three experiments: firstly with initialization on the ground truth BB, secondly with initialization on a noisy ground truth BB, and lastly with gray-scaled image sequences. Trackers are then ordered accordingly to a joint ranking of all three experiments. VOT2014 [44] does a joint ranking of only the two first experiments, leaving the gray-scaled one behind. Compared to the first one, the second experiment adds an element of randomness to the initialization of all trackers by perturbing the size and shape of the ground truth BB up to 10%. [45] introduces the first VOT-TIR in 2015 to rank trackers in infra-red and thermal imagery. The authors also add the

EAO performance measure to which the ranking is based, and the following VOT2016 challenge [46] follows that protocol as well. The 2017 VOT challenge [47] evaluates and ranks similarly to the previous one but adds a real-time challenge which accounts for tracking speed in the ranking process. The 2018 VOT challenge [48] tackles three experiments separately, short-term, real-time, and long-term tracking, to which each have their own ranking. Finally, the VOT 2019 challenge [63] adds two challenges: VOT-RGBT, which focuses on short-term tracking in RGB and thermal imagery, and VOT-RGBD, which focuses on long-term tracking in RGB and depth imagery. [4] revisits the VOT 2013 challenge alongside OTB-100 and creates mirror-transformed versions of these datasets to evaluate and rank the robustness of trackers subject to mirroring.

In [83], the authors conduct a comparative experiment on several objective measures to determine which are equivalent information-wise and then rank a set of test trackers using the least correlated measures to reflect on different aspects of tracking. Similarly, [77] searches for multiple measures to best describe accuracy and robustness but its final decision on which measures to use for ranking is biased from the start as it favors detection-based trackers. [42] compares accuracy and real-time performances on a dataset at two different frame rates (30 FPS and 240 FPS) and shows that changing the frame rate affects the ranking of trackers depending on their tracking principle. [68] applies four different ranking methods and averages them into a mean rank; the first two model datasets as graphs and assign ranks using both an aggregation algorithm and a PageRank-based solution [67], whilst the two last rankings are derived from the Elo [22] and Glicko [33] sports rating systems. Similarly to our work, the authors of that paper identify trackers as either best or second best; however, any tracker which is not qualified as best is automatically assumed to be second best, whereas our algorithm objectively justifies both score assignments.

2.3 Proposed Ranking Method

Our proposed method consists of three stages: scoring, grouping, and ranking. The inputs to the system are (a) the output BB of all tested trackers $\{t_i; i = 1, \cdots, T\}$ over all the test sequences $\{v_l; l = 1, \cdots, L\}$, each sequence has N_l frames $\{F_t; t = 1, \cdots, N_l\}$ (b) the ground truth BBs over said sequences, (c) the performance measure, and (d) the data dispersion estimator, MAD (Median Absolute Deviation) in our case. Quality data q_{il} is first calculated according to the selected measure (for example, AOR or FR) over each tracker i and sequence l. The scoring stage applies the data dispersion estimator on each T-sized row of $\{q_{il}\}$ and outputs a score s_i to each t_i; $\frac{1}{L} \leq s_i \leq 1$. A higher s_i for a tracker i indicates that t_i is more consistent in yielding good outputs. The output vector $\{s_i\}$ serves as the input quality data for the grouping and ranking stages where, again, the MAD dispersion estimator is iteratively applied on $\{s_i\}$ to group all trackers i into groups $\{g_i\}$ and ranks $\{r_i\}$.

2.3.1 Similarity-Based Scoring

The input to scoring is a T by L matrix of quality data $\{q_{il}\}$. A tracker t_i scores best or second best when it achieves best or second best average performance among the entirety of T trackers for a sequence l. However, due to possible outliers or similarities in the quality data for a v_l, simply appointing the best and second best scores to the two top performance values does not make up for a fair assignment, as doing so does not account for the dispersion in the quality data. We, therefore, define for each test sequence l a deviation threshold d_q based on the MAD estimator as in (6),

$$d_q = Median(|\{q_{il}\} - Median(\{q_{il}\})|). \tag{6}$$

For each tracker i, the MAD evaluates its quality data's affiliation to either a best or a second best score. Our scoring methodology is described in Algorithm 1 and is ran twice to first assign best scores, then second best scores. Scores are appointed to

trackers $\{t_i\}$ on a per-sequence basis until all sequences $\{v_l\}$ are processed sequentially. For each t_i and an objective measure, the final score s_i over all $\{v_l\}$ is the normalized summation of all its obtained scores for that measure; therefore, the output is a set of T final scores. In the first scoring round, all trackers i are scored, whereas in the second round, only trackers which have not scored best are considered for evaluation. In Algorithm 1, $Best(\{q_{il}\})$ represents the best value (for instance the maximum AOR or the minimum FR) in a set $\{q_{il}\}$.

Algorithm 1: Scoring of trackers over all $\{v_l\}$ for an objective measure.

Data: Quality data $\{q_{il}\}$ of all trackers $\{t_i; i = 1, \cdots, T\}$ on all test sequences $\{v_l; l = 1, \cdots, L\}$.
Result: Score s_i for each t_i.

```
for each tracker t_i do
    s_i = 0;
end
for each test sequence v_l do
    d_q = MAD({q_il});
    q_o = Best({q_il});
    for each tracker t_i do
        if |q_il − q_o| < d_q then
            s_i = s_i + 1;
        end
    end
end
{s_i} = {s_i} / L;
```

Since a t_i can be given either a best or a second best score, more weight is attributed to the best ones. Therefore, we set the score weighting system {best, second best} to fixed values $\{0.8, 0.2\}$. We also altered those weights from 0.6 to 0.9, with increments of 0.1, and did not notice important variations in our final scoring results when compared to our chosen distribution. We also thought about increasing the number of best sets of scores to evaluate; however, increasing the number of best sets does not improve our system and is even more likely to assign a score to trackers that are poorly performing. In the scenario where we would have added a third best

score, the weight would be so low that the final output would not be too different to our current scoring system. For this reason, we only score best and second best performing trackers.

2.3.2 Similarity-Based Grouping

Grouping divides the T test trackers into a set of groups $\{g_i, i = 1, \cdots, G \leq T\}$ of similarly performing trackers based on the T scores $\{s_i\}$ of an objective measure over all test sequences $\{v_l\}$. We name *group* the t_i which share similar g_i. There can be T groups if all trackers have fully different scores; otherwise, the number of groups $G < T$.

Algorithm 2 shows how grouping is performed. First, it calculates the MAD threshold d_s over the T scores s_i of the unassigned (not yet grouped) trackers i in $\{t_i\}$. A score s_i which is within the range defined by d_s and $\max(\{s_i\})$ has its corresponding tracker assigned a counter *count* as its group g_i; the counter starts at 1 and increments after each round. A new round of grouping starts if there still exists an unassigned tracker in $\{t_i\}$. The grouping algorithm ends whenever each t_i is assigned a g_i.

Algorithm 2: Grouping of all trackers $\{t_i\}$.

Data: Scoring data $\{s_i\}$ of all trackers $\{t_i; i = 1, \cdots, T\}$.
Result: Group g_i for each t_i.

```
count = 1;
while ∃ unranked t_i in {t_i} do
    d_s = MAD({s_i});
    s_o = max({s_i});
    for each unassigned t_i in {t_i} do
        if |s_i − s_o| ≤ d_s then
            g_i = count;
        end
    end
    count++;
end
```

2.3.3 Iterative Ranking

The T tested trackers are iteratively scored and ranked based on the quality data $\{q_{il}\}$ over all test sequences $\{v_l\}$. At the start of each iteration k, each tracker is placed in a set j of trackers with the same assigned rank $\{t_i\}_j$, $j = 1, \cdots, J \leq T$. Initially, none of the t_i are ranked and they all belong in the same set j of unranked trackers (i.e., $r_i^0 = 0$ and $J = 1$). The first iteration is similar to grouping; therefore, $r_i^1 = g_i$. At iteration k, a rank is assigned at each tracker and multiple trackers may have the same rank. The ranking algorithm ends whenever all ranks in the final set $\{r_i^{1:k}\}$ are all distinct from each other or when $k = T$. For more simplicity, the final rank obtained at the last iteration is named *rank* and is noted r_i. As an example, iterations operate as follows. In the first iteration, the ranking algorithm ranks and partitions the T trackers into J^1 sets j^1 based on the trackers' scores $\{s_i\}$ computed from the quality values $\{q_{il}\}$; best trackers are in set 1, second best trackers are in set 2, third best trackers are in set 3, etc. In the second iteration, the J^1 sets j^1 are partitioned into $J^2 > J^1$ subsets j^2 based on the scores re-computed from the quality values $\{q_{il}\}_j$ of the t_i in each set j^1. Scores are re-computed in each set to better determine how trackers within the same set perform with respect to each other. In the following iterations, the same process is repeated: subsets are partitioned into smaller subsets, and all trackers in the same newly partitioned subset are scored and ranked against each other. In the last iteration, there can be up to $J = T$ sets, that is, when all trackers are ranked differently.

Algorithm 3 shows more precisely how the entire ranking process is achieved. First, we use the quality data $Q_j = \{q_{il}\}_j$, in other words, the quality values of trackers i in the set of trackers $\{t_i\}_j$ over all test sequences $\{v_l\}$. Q_j is a matrix of size $T_j \times L$, where $T_j = |\{t_i\}_j|$ is the number of trackers in $\{t_i\}_j$ at iteration k; a row in Q_j designates a tracker i from $\{t_i\}_j$ and a column designates a sequence l. Then, we *score()* best and second best performances of each tracker i in $\{t_i\}_j$ as per section

2.3.1 to produce the T_j scores of set j, namely $\{s_i\}_j$. We calculate the MAD threshold d_s over $\{s_i\}_j$, and a score which is within the range defined by d_s and $\max(\{s_i\}_j)$ has its corresponding tracker assigned a counter *count* as its rank r_i^k; the counter starts at 1 and increments after each round of ranking ("while" loop in Algorithm 3), that is, after at least one tracker in $\{t_i\}_j$ has been ranked. A new round starts if there still exists an unranked t_i in $\{t_i\}_j$. Once all trackers are ranked in iteration k, we take the new sets $\{t_i\}_j$ of similarly ranked trackers for iteration $k+1$. The sets j for iteration $k+1$ depend on the tracker ranks at iteration k, and j varies from 1 and J. The algorithm iterates if $k \leq T$ and if at least two trackers are of the same rank. The algorithm stops either when $k = T+1$ or when all trackers are ranked differently. In fact, T is the minimal amount of iterations required to fully achieve ranking, without exception, over all $\{t_i\}$. Hence, a final rank $1 \leq r_i \leq T$ and only trackers that have the same score end with the same rank.

Essentially, algorithm 3 in its first iteration $k = 1$ divides all trackers $\{t_i\}$ into groups (same as algorithm 2); then, in each of the following iterations, each group is divided into sub-groups, and each tracker in each sub-group is given a rank. Ranking is based on the scores which are recomputed at each iteration. This is repeated until all trackers are ranked.

At each new iteration, each trackers within a same set j have its score recomputed. The score at $k = 1$ is the most relevant to display for analysis since it is used for grouping and shows how all T trackers compare with respect to the top performing trackers. At the last iteration, the final score of a tracker i is simply the score of the tracker in the last set it was assigned in. This means the final score is not meaningful to show or interpret on its own. However, the final rank r_i depends on all scores, from the first iteration to the last iteration, hence the importance of computing scores at each iteration.

Algorithm 3: Iterative ranking of all trackers $\{t_i\}$.

Data: Quality data $\{q_{il}\}$ of all trackers $\{t_i; i = 1, \cdots, T\}$ on all test sequences $\{v_l; l = 1, \cdots, L\}$.
Result: Rank r_i for each t_i.

$k = 1$;
$j = 1$;
$\{t_i\}_j = \{t_i\}$;
do
  for *each* $\{t_i\}_j$ **do**
    $Q_j = \{q_{il}\}_j$;
    $\{s_i\}_j = score(Q_j)$;
    $count = 1$;
    while $\exists$ *unranked* t_i *in* $\{t_i\}_j$ **do**
      $d_s = MAD(\{s_i\}_j)$;
      $s_o = \max(\{s_i\}_j)$;
      for *each unranked* t_i *in* $\{t_i\}_j$ **do**
        if $|s_i - s_o| \leq d_s$ **then**
          $r_i^k = count$;
        end
      end
      $count++$;
    end
  end
  $k++$;
  $\{t_i\}_j = update(\{t_i\}_j)$;
while $k \leq T$ *and* $\exists\ t_{i_1} \wedge t_{i_2}$ *such that* $r_{i_1} = r_{i_2}$;

2.3.4 How Robust is our Method?

Our scoring and ranking methods are based on a robust dispersion estimator: the MAD. To demonstrate their robustness, we run a total of M experiments where the original AOR quality data of each frame of a test video signal and for all trackers are subjected to variations of signal independent noise; then, we evaluate our algorithm's response to impulse and non-zero mean Gaussian noise.

In the context of this experiment, impulse noise can be interpreted as the result of random sharp and sudden image changes such as occlusion or fast motion. We can also interpret the additivity of Gaussian noise to $\{q_{il}\}$ as random gradual changes through frames such as illumination variation or motion blur.

To show the robustness of our scoring method, given T tested trackers $\{t_i; i = 1, \cdots, T\}$, let their scores be $\{s_i\}$ and mean measures be $\{p_i\}$ generated from the original quality data $\{q_{il}\}$ of an objective measure (such as AOR) over all sequences $\{v_l\}$, and let their noisy scores be $\{s_i^n\}$ and noisy mean measures be $\{p_i^n\}$ generated from the noisy quality data $\{q_{il}^n\}$ of that same objective measure. Our scoring is more robust than averaging (mean) if, after M experiments,

$$\forall i, \quad \Psi(s_i, \mu_{s_i^n}) > \Psi(p_i, \mu_{p_i^n}), \tag{7}$$

where $\Psi(\cdot) = \frac{min(\cdot)}{max(\cdot)} \in [0, \cdots, 1]$ is a min-max ratio applied for normalization,

$$\mu_{s_i^n} = \frac{1}{K} \sum_{k=1}^{K} s_i^k, \tag{8}$$

$$\mu_{p_i^n} = \frac{1}{K} \sum_{k=1}^{K} p_i^k. \tag{9}$$

$\mu_{s_i^n}$ is the average score of tracker i over K different levels of a noise distribution under test, and $\mu_{p_i^n}$ is the average performance mean of tracker i over the same K noise levels. In other words, our scoring is more robust than the mean if, overall, each t_i yields a ratio of scores $\Psi(s_i, \mu_{s_i^n})$ closer to 1 than the ratio of means $\Psi(p_i, \mu_{p_i^n})$.

To test if our ranking method is robust, let the ranks of all the T trackers be $\{r_i\}$ generated from the original quality data $\{q_{il}\}$ of an objective measure over each sequence l, and let their noisy ranks be $\{r_i^n\}$ generated from the noisy quality data $\{q_{il}^n\}$ of that same objective measure. Our ranking is robust if, after M experiments,

$$\forall i, \quad |r_i - \mu_{r_i^n}| \leq 1, \tag{10}$$

where $\mu_{r_i^n}$ is the average rank of each tracker i over K different levels of a noise distribution,

$$\mu_{r_i^n} = \frac{1}{K} \sum_{k=1}^{K} r_i^k. \tag{11}$$

In other words, our ranking is robust if, overall, each t_i yields at most a difference of 1 between its original r_i and its corresponding average of noisy ranks $\mu_{r_i^n}$.

Given the original overlap ratio quality data $\{q_{il}\}$, that is, AOR_l values at each test video sequence v_l for each tracker t_i, we add noise to AOR_l to get $\{q_{il}^n\}$. For impulse noise, the noisy quality data $\{q_{il}^n\}$ is the original quality data to which we apply a binary-state sequence, with a noise density varying from 0.05 to 0.5, specifically, 0.05, 0.2, 0.35, and 0.5. For Gaussian noise, we pick three distributions with different standard deviations *std*, namely 0.01, 0.05, and 0.1. To each Gaussian distribution, we vary the mean from 0.1 to 0.4, specifically, 0.1, 0.2, 0.3, and 0.4. We run the experiments $M = 50$ times across the combined dataset OTB-100+VOT2018-ST+NfS-30.

Tables 3 and 4 display $\Psi(p_i, \mu_{p_i^n})$ (noted 'Mean ratio'), $\Psi(s_i, \mu_{s_i^n})$ (noted 'Score ratio') and $|r_i - \mu_{r_i^n}|$ (noted 'Rank diff.') results under impulse noise and Gaussian noise, respectively. Both tables show that our proposed ranking responds moderately to even strong variations in the input data not just on average, but for every tracker i. Indeed, the rank difference is less or equal to 1, affirming the robustness of our ranking methodology. We also see that, overall, the score ratio does not goes below 0.85, which is comparatively much more robust than the mean ratio where the ratio can reach 0.55. We notice that in the case of impulse noise, the average score ratio

and the average mean ratio are comparable and remain above 0.9, but in all cases of Gaussian disturbances, the score performs much better than the mean.

In Table 5, we show the percentage of trackers with mean ratio and score ratio above a certain threshold th_s over all 16 cases of noise levels: 4 levels of impulse noise as per Table 3 and 12 levels of Gaussian noise as per Table 4. We can observe that the score is more robust than the mean.

We applied our robustness experiment only to the AOR values AOR_l at each sequence but not to the FR values FR_l because the latter are too small. In fact, the average of all our AOR data is 0.46331, whereas that of our FR data is 0.2535, and larger levels of noise negatively affects the experiment if our measure is, overall, closer to 0 or to 1 (here the FR). To keep our noise distribution as unaltered as possible, it is best to operate on a measure that is, overall, closer to 0.5 (here the AOR).

2.4 Results and Discussion

2.4.1 Experimental Setup

For testing our methods, we selected 20 trackers of different performance, speed, and with their code publicly available: ATOM [24], CFWCR [38], CREST [79], CSRDCF [61], DASIAMRPN [100], DAT [72], DIMP [97], DLST [94], DSST [26], ECO [25], IBCCF [54], KCF [40], LADCF [92], MCCT [86], MDNET [65], SAMF [93], SIAMFC [6], SIAMRPN++ [51], STAPLE [5], and STRCF [53]. Table 6 briefly describes each of their tracking principles. As can be seen, the majority of them are correlation filter-based, CNN-based, Siamese, or a combination of architectures, as those methods have proven to compete most with the state-of-the-art [47, 48, 63]. DAT was chosen to compete among the older trackers and is the only one to solely rely on a color tracking principle. LADCF and MCCT trackers used in our evaluation are correlation filter-based but also exist with a CNN architecture.

Table 3: AOR mean ratio, score ratio, and rank difference results under impulse noise and averaged over $M = 50$ runs for each tracker t_i in the combined dataset OTB-100+VOT2018-ST+NfS-30.

(a) Impulse noise of density 0.05 and 0.2

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.981078654	0.984974282	0
CFWCR	0.996780421	0.932929316	0
CSRDCF	0.988732015	0.965226743	0
CREST	0.983288468	0.892499209	1
DASIAMRPN	0.991781021	0.879023526	0.5
DAT	0.917651148	0.925658363	0
DIMP	0.97464534	0.995363048	0
DLST	0.983835936	0.970115304	0.5
DSST	0.957685338	0.878554701	0
ECO	0.992843112	0.929840395	0
IBCCF	0.996848717	0.92409775	1
KCF	0.936910052	0.969487197	0
LADCF	0.996321597	0.863093764	0
MCCT	0.99156707	0.90154167	1
MDNET	0.994658004	0.938606526	0
SAMF	0.969248802	0.807540922	0
SIAMFC	0.975777956	0.825912014	0.5
SIAMRPN++	0.995545584	0.988086439	0.5
STAPLE	0.975109287	0.997254617	0
STRCF	0.999360849	0.850102995	0
Mean	0.979983469	0.920995439	0.25

(b) Impulse noise of density 0.35 and 0.5

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.934882845	0.974849135	0
CFWCR	0.990490375	0.939047841	0
CSRDCF	0.962662934	0.906076358	1
CREST	0.945035584	0.8724549	0
DASIAMRPN	0.97336923	0.896741805	0.5
DAT	0.766090635	0.994684512	0
DIMP	0.91390958	0.982298687	0
DLST	0.946237705	0.972153361	1.5
DSST	0.868971616	0.824355972	0
ECO	0.976099981	0.926515295	0
IBCCF	0.990599838	0.949182716	0
KCF	0.812712296	0.949324551	0
LADCF	0.987156695	0.854178018	0
MCCT	0.972919019	0.901169286	0
MDNET	0.982325837	0.951952933	0
SAMF	0.903102689	0.7305631	0
SIAMFC	0.921908232	0.801948305	1
SIAMRPN++	0.984727234	0.974533898	1
STAPLE	0.9194122	0.93859258	0
STRCF	0.997719301	0.843236677	0
Mean	0.937516691	0.909192997	0.25

(c) Impulse noise of density 0.05 to 0.5

Impulse noise	Mean ratio	Score ratio	Rank diff.
ATOM	0.95798075	0.979911709	0
CFWCR	0.993635398	0.935988579	0
CSRDCF	0.975523343	0.93565155	0.5
CREST	0.963782608	0.882477055	0.5
DASIAMRPN	0.982488874	0.887882666	0.5
DAT	0.835049599	0.958930876	0
DIMP	0.94427746	0.988787716	0
DLST	0.96467061	0.971133263	1
DSST	0.911174234	0.851455336	0
ECO	0.984471546	0.928177845	0
IBCCF	0.993714454	0.936640233	0.5
KCF	0.870403057	0.959405874	0
LADCF	0.991739146	0.858635891	0
MCCT	0.982154536	0.901355478	0.5
MDNET	0.988491921	0.94527973	0
SAMF	0.935007346	0.769052011	0
SIAMFC	0.948078492	0.81393016	0.75
SIAMRPN++	0.990106859	0.992941176	0.25
STAPLE	0.946442023	0.967923599	0
STRCF	0.998540075	0.846669836	0
Mean	0.957886617	0.915611529	0.225

Table 4: AOR mean ratio, score ratio, and rank difference results under Gaussian noise and averaged over $M = 50$ runs for each tracker t_i in the combined dataset OTB-100+VOT2018-ST+NfS-30.

(a) Gaussian noise with $std = 0.01$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.748214507	0.955626512	0
CFWCR	0.709223798	0.963154202	0
CSRDCF	0.676218643	0.814155418	0.5
CREST	0.665404088	0.884441334	0.25
DASIAMRPN	0.676245202	0.851372303	0.25
DAT	0.551333114	0.722026371	0
DIMP	0.763261544	0.947672827	0
DLST	0.667779477	0.796805854	1
DSST	0.626906324	0.822962555	0.75
ECO	0.719521345	0.975193978	0
IBCCF	0.697200361	0.862988827	0.25
KCF	0.587551475	0.759617103	0
LADCF	0.715058153	0.987171232	0.75
MCCT	0.687020924	0.898427568	0.25
MDNET	0.712601568	0.922114781	0.75
SAMF	0.643980694	0.917707431	0.5
SIAMFC	0.656223841	0.909007092	1
SIAMRPN++	0.679283962	0.747479179	0
STAPLE	0.653835926	0.816417625	0.25
STRCF	0.708117988	0.968546981	0
Average	0.677249147	0.876144459	0.325

(b) Gaussian noise with $std = 0.05$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.750081733	0.956552058	0
CFWCR	0.71079911	0.975472502	0
CSRDCF	0.677649947	0.84722855	0.25
CREST	0.666787461	0.914520019	0.75
DASIAMRPN	0.677679033	0.865885361	0.75
DAT	0.552113129	0.724036186	0
DIMP	0.765255539	0.952071515	0
DLST	0.669141625	0.812724479	0.75
DSST	0.628073436	0.816666667	0
ECO	0.721142927	0.98339367	0
IBCCF	0.698786194	0.872740113	0.25
KCF	0.588503664	0.788813784	0
LADCF	0.716701809	0.967853782	0.75
MCCT	0.688532417	0.921234991	0.75
MDNET	0.714264459	0.932655913	0.75
SAMF	0.64521491	0.920160374	0
SIAMFC	0.657564815	0.934572917	0.75
SIAMRPN++	0.680738166	0.750681355	0.75
STAPLE	0.655113483	0.830133918	0
STRCF	0.709728753	0.959964084	0
Average	0.678693631	0.886368112	0.325

(c) Gaussian noise with $std = 0.1$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.755364431	0.963833317	0
CFWCR	0.715102504	0.977277904	0
CSRDCF	0.681508377	0.889260809	0.25
CREST	0.670359174	0.925909646	0.75
DASIAMRPN	0.681550049	0.876720604	0.75
DAT	0.553648704	0.756683636	0
DIMP	0.770917229	0.956428633	0
DLST	0.672796428	0.834714769	1
DSST	0.630710363	0.858168246	0
ECO	0.725544081	0.98316404	0
IBCCF	0.702987379	0.884135217	0.5
KCF	0.590429639	0.82034459	0
LADCF	0.721047971	0.957699737	0.75
MCCT	0.692516513	0.946730125	0.5
MDNET	0.718964236	0.934912147	0.75
SAMF	0.648278495	0.955600278	0
SIAMFC	0.660866444	0.924223539	1
SIAMRPN++	0.684797439	0.760297521	0.25
STAPLE	0.658331094	0.873746797	0
STRCF	0.713915243	0.945967359	0
Average	0.68248179	0.901290946	0.325

Table 5: Percentage of trackers with mean ratio or score ratio above th_s.

Threshold th_s	Mean ratio % trackers	Score ratio % trackers
0.85	38	72.5
0.9	37	56
0.95	28	28

Table 6: Tracking principle and FPS over each benchmark for each tested tracker.

Tracking Principle	Siamese Network	Discriminative Correlation Filter	CNN Matching	Color Template	Code	Frames per second OTB	VOT-ST	VOT-LT	NfS	Average FPS
1. ATOM [24]	✓	-	✓	-	CC / GPU / P	21.324	25.761	30.836	24.468	25.597
2. CFWCR [38]	-	✓	✓	-	PC / GPU / M	3.231	4.675	2.890	2.828	3.578
3. CSRDCF [61]	-	✓	-	-	PC / CPU / M	15.008	12.543	10.455	9.993	12.515
4. CREST [79]	-	✓	✓	-	PC / GPU / M	1.463	1.592	1.316	1.674	1.576
5. DASIAMRPN [100]	✓	-	✓	-	CC / GPU / P	68.313	175.55	71.724	72.341	96.982
6. DAT [72]	-	-	-	✓	PC / CPU / M	90.975	55.100	44.948	43.908	63.328
7. DIMP [97]	✓	-	✓	-	CC / GPU / P	25.792	26.520	30.438	26.606	27.339
8. DLST [94]	-	-	✓	-	PC / GPU / M	0.888	0.899	-	0.774	0.854
9. DSST [26]	-	✓	-	-	PC / CPU / M	53.054	54.516	39.683	42.193	49.921
10. ECO [25]	-	✓	✓	-	PC / GPU / M	2.082	2.298	1.934	1.394	1.925
11. IBCCF [54]	-	✓	✓	-	PC / GPU / M	0.660	0.670	0.559	0.572	0.634
12. KCF [40]	-	✓	-	-	PC / CPU / M	121.960	79.050	51.262	37.436	79.482
13. LADCF [92]	-	✓	-	-	PC / CPU / M	23.775	21.853	18.755	19.291	21.640
14. MCCT [86]	-	✓	-	-	PC / CPU / M	39.094	7.476	27.665	32.696	26.422
15. MDNET [65]	-	-	✓	-	PC / GPU / M	0.708	0.717	0.641	0.659	0.695
16. SAMF [93]	-	✓	-	-	PC / CPU / M	26.185	15.031	8.636	6.502	15.906
17. SIAMFC [6]	✓	-	✓	-	PC / GPU / M	14.117	13.93	-	8.919	12.322
18. SIAMRPN++ [51]	✓	-	✓	-	CC / GPU / P	47.747	44.494	30.973	39.464	40.670
19. STAPLE [5]	-	✓	-	-	PC / CPU / M	77.335	71.463	49.510	61.56	70.119
20. STRCF [53]	-	✓	-	-	PC / CPU / M	25.985	20.568	18.001	18.935	21.829

We collect the mean, score, group, and rank of each tracker for AOR and FR measures over three short-term datasets: OTB-100 [91], VOT2018-ST [48], and NfS-30 [42] (Chapter 1 section 1.5.1 briefly introduces these datasets). The scores and groups displayed in all following tables is that calculated at iteration $k = 1$ since that is the only time all trackers compete against each other.

2.4.2 Experimentation on OTB-100 Dataset

Mean, score, group, and rank results over the 100 sequences of OTB-100 are displayed in Table 7. We first notice that the scoring does not replicate outputs similar to the mean. For example, there is much more variance in the AOR scores of DIMP and ECO trackers than in their respective mean AOR. We also see that a tracker can score higher even if it has a lower mean than another tracker. For example, DASIAMRPN has a higher mean AOR than CSRDCF but its score is slightly lower. Our results show that the score does not directly correlate with the objective measure itself; it is instead an indication of the frequency at which a tracker performs best relatively to its counterparts using that objective measure. Hence, ranking would be different if it were done according to the mean. Table 7 also shows that one tracker can be overall more accurate (i.e., higher AOR score) than another one, while being simultaneously less robust (i.e., lower FR score). For instance, SIAMRPN++ is in groups 5 and 2 for AOR and FR, respectively, whereas STAPLE is assigned groups 3 and 3 for those measures. This means that SIAMRPN++ overall fails less than STAPLE on OTB-100, but that STAPLE more consistently yields better accuracy. Moreover, we notice less differences between FR scores and means than between AOR scores and means. This makes sense since many of our tested trackers rarely fail and yield consistent FR results across the widely studied OTB-100 dataset. Consequently, FR data variability is much lower and the ranking based on score looks more closely to that which would be based on the mean.

Table 7: AOR and FR mean, score, group, and rank for all trackers t_i over the OTB-100 dataset. The 5 top-ranked trackers in terms of AOR are: ECO, LADCF, STRCF, DIMP, CFWCR; those in terms of FR are: STRCF, LADCF, ECO, DIMP, MDNET.

AOR	Mean	Score	Group	Rank
ATOM	0.66047	0.25287	1	6
CFWCR	0.65476	0.24667	1	5
CSRDCF	0.58432	0.13129	4	14
CREST	0.58197	0.12317	4	13
DASIAMRPN	0.60037	0.12148	4	12
DAT	0.33449	0.029813	7	20
DIMP	0.67811	0.25325	1	4
DLST	0.54874	0.096911	5	17
DSST	0.52344	0.1312	4	16
ECO	0.67839	0.29586	1	1
IBCCF	0.63671	0.16511	3	9
KCF	0.48142	0.053513	6	19
LADCF	0.67518	0.30778	1	2
MCCT	0.63866	0.19006	3	8
MDNET	0.66146	0.2309	2	7
SAMF	0.56328	0.12273	4	15
SIAMFC	0.58167	0.14678	4	11
SIAMRPN++	0.57366	0.093188	5	18
STAPLE	0.59016	0.16178	3	10
STRCF	0.668	0.28563	1	3

FR	Mean	Score	Group	Rank
ATOM	0.061709	0.40584	1	7
CFWCR	0.067064	0.42427	1	6
CSRDCF	0.11426	0.37435	2	12
CREST	0.12212	0.37312	2	13
DASIAMRPN	0.073469	0.3682	2	11
DAT	0.34213	0.17894	5	20
DIMP	0.039562	0.43615	1	4
DLST	0.14764	0.35189	2	14
DSST	0.21229	0.28721	4	18
ECO	0.056994	0.42827	1	3
IBCCF	0.080992	0.38043	2	9
KCF	0.19949	0.27897	4	19
LADCF	0.070324	0.44906	1	2
MCCT	0.088518	0.38669	2	10
MDNET	0.061759	0.42391	1	5
SAMF	0.14799	0.34819	3	16
SIAMFC	0.14506	0.34646	3	15
SIAMRPN++	0.063241	0.39234	2	8
STAPLE	0.15002	0.33433	3	17
STRCF	0.074952	0.44069	1	1

AOR results split the trackers into 7 groups of performance. The first AOR group includes ATOM, CFWCR, DIMP, ECO, LADCF, and STRCF. FR results split the trackers into 5 groups which is a smaller amount of groups than with AOR. The first FR group includes ATOM, CFWCR, DIMP, ECO, LADCF, MDNET, and STRCF trackers, which is one more tracker than the first AOR group. In fact, the lower data variability in FR explains the reduced number of groups, hence the addition of MDNET in group 1.

2.4.3 Experimentation on VOT2018-ST Dataset

Mean, score, group, and rank results over the 60 sequences of VOT2018-ST are displayed in Table 8. For matters of consistency with OTB-100, we use a rectangular annotated ground truth at each frame instead of the rotated one provided by the benchmark. We first notice that the number of AOR and FR groups are larger than in OTB-100, underlying the higher dispersion of data. In addition to that, the mean

Table 8: AOR and FR mean, score, group, and rank for all trackers t_i over the VOT2018-ST dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, MDNET, SIAMRPN++, IBCCF; those in terms of FR are: DIMP, SIAMRPN++, MDNET, ATOM, CSRDCF.

AOR	Mean	Score	Group	Rank
ATOM	0.48283	0.23037	2	2
CFWCR	0.38706	0.070731	5	14
CSRDCF	0.37496	0.10206	4	11
CREST	0.35584	0.086165	4	12
DASIAMRPN	0.40275	0.12019	3	7
DAT	0.27111	0.030051	7	18
DIMP	0.54821	0.33489	1	1
DLST	0.35568	0.076726	5	13
DSST	0.26008	0.014031	8	20
ECO	0.38471	0.12401	3	6
IBCCF	0.37951	0.15224	3	5
KCF	0.25528	0.027177	7	19
LADCF	0.36317	0.12133	3	8
MCCT	0.37466	0.13808	3	9
MDNET	0.47667	0.26904	2	3
SAMF	0.3151	0.065374	5	15
SIAMFC	0.28669	0.040221	6	17
SIAMRPN++	0.43424	0.15648	3	4
STAPLE	0.3294	0.062432	5	16
STRCF	0.35269	0.095349	4	10

FR	Mean	Score	Group	Rank
ATOM	0.22094	0.32067	2	4
CFWCR	0.24885	0.30765	2	8
CSRDCF	0.28918	0.28864	2	5
CREST	0.34068	0.22049	3	11
DASIAMRPN	0.23565	0.31615	2	7
DAT	0.41683	0.13179	6	20
DIMP	0.15352	0.40835	1	1
DLST	0.34039	0.22679	3	12
DSST	0.4761	0.13185	5	19
ECO	0.29352	0.29904	2	6
IBCCF	0.33421	0.28192	2	9
KCF	0.46715	0.19885	4	17
LADCF	0.35591	0.23929	3	13
MCCT	0.35584	0.27759	2	10
MDNET	0.21411	0.3632	1	3
SAMF	0.4068	0.22189	3	14
SIAMFC	0.42409	0.17775	4	18
SIAMRPN++	0.17486	0.36259	1	2
STAPLE	0.36835	0.23093	3	16
STRCF	0.37352	0.21414	3	15

performance measures are lower overall than in OTB-100. Despite having shorter sequences on average, VOT2018-ST has more challenging videos than OTB-100, and such attributes increase a tracker's likeliness to drift earlier in a sequence. Indeed, datasets from the VOT challenge are updated to keep up with the state-of-the-art, whereas OTB-100 has remained unchanged since its release.

AOR and FR performance measures divide the candidate trackers into 8 and 6 groups, respectively. AOR group 1 only consists of DIMP whereas FR group 1 includes DIMP, MDNET, and SIAMRPN++. Trackers such as ATOM, ECO, and LADCF that were ranked first in Table 7 over OTB-100 are now ranked in a lower group, hence emphasizing the difference in performance between those trackers and DIMP.

Table 9: AOR and FR mean, score, group, and rank for all trackers t_i over the NfS-30 dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, STRCF, ECO, CFWR; those in terms of FR are: DIMP, ATOM, CFWCR, ECO, LADCF.

AOR	Mean	Score	Group	Rank
ATOM	0.58561	0.25773	1	2
CFWCR	0.4453	0.14445	2	5
CSRDCF	0.38228	0.10166	3	8
CREST	0.34775	0.073993	4	14
DASIAMRPN	0.37921	0.077612	3	11
DAT	0.2597	0.0213	6	17
DIMP	0.62307	0.28944	1	1
DLST	0.38551	0.10303	3	9
DSST	0.28051	0.041648	5	16
ECO	0.46645	0.1693	2	4
IBCCF	0.40722	0.069012	4	12
KCF	0.20883	0.0066866	8	20
LADCF	0.44727	0.13601	2	6
MCCT	0.35691	0.078146	3	10
MDNET	0.4105	0.10301	3	7
SAMF	0.2864	0.016411	7	18
SIAMFC	0.32932	0.053794	4	15
SIAMRPN++	0.42076	0.070547	4	13
STAPLE	0.29171	0.016421	7	19
STRCF	0.42849	0.13571	2	3

FR	Mean	Score	Group	Rank
ATOM	0.14553	0.33238	1	2
CFWCR	0.25451	0.26407	2	3
CSRDCF	0.34074	0.21811	3	12
CREST	0.39732	0.19383	3	10
DASIAMRPN	0.31826	0.20895	3	11
DAT	0.44931	0.14328	5	18
DIMP	0.14074	0.33886	1	1
DLST	0.34155	0.19114	3	9
DSST	0.52114	0.12089	6	19
ECO	0.26452	0.26101	2	4
IBCCF	0.34482	0.18068	4	13
KCF	0.56598	0.082056	7	20
LADCF	0.28651	0.24439	2	5
MCCT	0.39756	0.18059	4	14
MDNET	0.32881	0.2013	3	8
SAMF	0.48344	0.14888	5	17
SIAMFC	0.42674	0.14394	5	15
SIAMRPN++	0.22451	0.26255	2	6
STAPLE	0.43588	0.13178	5	16
STRCF	0.29873	0.22221	2	7

2.4.4 Experimentation on NfS-30 Dataset

The NfS benchmark is not as well known or as widely used as OTB-100 and VOT2018-ST. It is divided into two datasets of 100 video sequences captured at different frame rates: 240 FPS (NfS-240) and 30 FPS (NfS-30). All frames are annotated with a rectangular ground truth. We test the trackers on NfS-30 as it presents a variety of examples which have not been explored as much as in OTB-100 and VOT2018-ST datasets, thus allowing to test their robustness to different target objects and environments. Table 9 displays AOR and FR performance results over the 100 sequences.

AOR and FR results split the trackers into 8 and 7 groups of performance, respectively. The FR-based number of groups is the highest amount among the three studied datasets. This means there is more FR variance in the NfS-30 than in the other datasets. Trackers ranked in the first group for both AOR and FR measures are ATOM and DIMP. Trackers such as ECO, LADCF, and MDNET which have ranked high in either OTB-100 or VOT2018-ST are placed in groups 2 or 3.

Table 10: AOR and FR mean, score, group, and rank for all trackers t_i over the combined short-term dataset OTB-100, VOT2018-ST, and NfS-30. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, ECO, MDNET, LADCF; those in terms of FR are: DIMP, ATOM, CFWCR, ECO, SIAMRPN++.

AOR	Mean	Score	Group	Rank
ATOM	0.59067	0.24691	1	2
CFWCR	0.51241	0.17161	2	7
CSRDCF	0.45829	0.11158	3	10
CREST	0.43969	0.09479	4	13
DASIAMRPN	0.46969	0.10159	4	12
DAT	0.29109	0.026011	7	20
DIMP	0.62696	0.27691	1	1
DLST	0.4414	0.092549	4	15
DSST	0.36923	0.074725	5	18
ECO	0.5291	0.21052	2	3
IBCCF	0.48908	0.12358	3	9
KCF	0.32438	0.029758	6	19
LADCF	0.51552	0.20189	2	5
MCCT	0.46937	0.13427	3	8
MDNET	0.52229	0.18357	2	4
SAMF	0.39951	0.068851	5	17
SIAMFC	0.41653	0.089719	4	14
SIAMRPN++	0.48267	0.091996	4	11
STAPLE	0.41519	0.085234	4	16
STRCF	0.50311	0.18933	2	6

FR	Mean	Score	Group	Rank
ATOM	0.13069	0.3551	1	2
CFWCR	0.18111	0.3324	2	3
CSRDCF	0.24173	0.29217	2	9
CREST	0.27841	0.26893	3	13
DASIAMRPN	0.20505	0.28836	3	10
DAT	0.40059	0.15217	6	20
DIMP	0.10478	0.38404	1	1
DLST	0.2667	0.2599	3	14
DSST	0.39196	0.1889	5	18
ECO	0.19139	0.33171	2	4
IBCCF	0.2409	0.27914	3	11
KCF	0.40222	0.18374	5	19
LADCF	0.21938	0.32353	2	6
MCCT	0.26907	0.28109	3	12
MDNET	0.19963	0.31691	2	7
SAMF	0.33673	0.2412	4	17
SIAMFC	0.31779	0.23027	4	16
SIAMRPN++	0.15103	0.32787	2	5
STAPLE	0.31035	0.23233	4	15
STRCF	0.22992	0.30748	2	8

2.4.5 Experimentation on Cross Datasets (Short-Term)

To obtain an overall evaluation of each tracker's accuracy and robustness performance across all short-term datasets (OTB-100, VOT218-ST, and NfS-30), one method would be to collect the rank of each tracker in each dataset and then assign it the mean or median rank as its final rank. Doing so, however, poses certain issues: (1) two different trackers may be assigned a similar final rank, hence defeating the purpose of ordering them, (2) smaller datasets are treated as equally as larger datasets, and (3) some trackers are trained to perform better in a specific benchmark, therefore, affecting the ranking when performed on that benchmark. To reduce possible bias towards certain benchmarks, test the generalization ability of the trackers, and make the ranking more fair, we run the scoring and ranking across all 260 video sequences of OTB-100, VOT2018-ST, and NfS-30 combined. In doing so, the output of our scoring and ranking method is a set of ranks and groups assigned at each tracker.

Table 10 displays AOR and FR means, scores, ranks, and groups across the combined short-term dataset. There is a total of 7 groups for AOR and 6 groups for FR. We see that ATOM and DIMP trackers are in group 1 for both AOR and FR performance measures. In addition, CFWCR, ECO, LADCF, MDNET, and STRCF trackers are in group 2 for both AOR and FR measures are. Even though CFWCR, ECO, LADCF, and STRCF were top trackers in OTB-100, we can clearly see that ATOM and DIMP are the best performing ones on the cross datasets.

2.4.6 Experimentation on Long-Term Dataset VOT2018-LT

We run the scoring and ranking on the VOT2018-LT dataset, consisting of 35 long-term videos. The aim of this experiment is to observe how much tracking performances can be affected by video length and differ from those on short-term videos. AOR and FR means, scores, ranks, and groups are displayed in Table 11.

We notice that DIMP is the only tracker to be assigned group 1 on both AOR and FR measures since it performs, overall, much better than its counterparts. Nonetheless, we can observe that ATOM, DASIAMRPN, LADCF, and SIAMRPN++ are in group 2 for both measures; among those four trackers, three of them (ATOM, DASIAMRPN, and SIAMRPN++) have an implemented target re-detection algorithm which allows them to reduce error loss when facing long-term challenges. While trackers such as ECO, MDNET, and STRCF performed well on short-term sequences, their incapability to re-detect lost objects puts them at a disadvantage on long-term sequences.

Table 11: AOR and FR mean, score, group, and rank for all trackers t_i over the VOT2018-LT dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, DASIAMRPN, SIAMRPN++, LADCF; those in terms of FR are: DIMP, SIAMRPN++, ATOM, DASIAMRPN, CFWCR.

AOR	Mean	Score	Group	Rank
ATOM	0.43031	0.21762	2	2
CFWCR	0.32468	0.10194	3	9
CSRDCF	0.25068	0.017302	6	14
CREST	0.30297	0.04794	5	11
DASIAMRPN	0.44682	0.20443	2	3
DAT	0.20798	0	8	17
DIMP	0.51174	0.37425	1	1
DSST	0.22894	0.0069117	7	16
ECO	0.3344	0.11799	3	7
IBCCF	0.31265	0.07494	4	10
KCF	0.15045	0	8	17
LADCF	0.35478	0.1575	2	5
MCCT	0.28742	0.025097	6	12
MDNET	0.33627	0.1176	3	6
SAMF	0.23821	0.022857	6	13
SIAMRPN++	0.42198	0.17557	2	4
STAPLE	0.22645	0.0093118	7	15
STRCF	0.32383	0.097085	3	8

FR	Mean	Score	Group	Rank
ATOM	0.35738	0.31182	2	3
CFWCR	0.36858	0.2678	2	5
CSRDCF	0.51136	0.14113	4	12
CREST	0.3983	0.20551	3	9
DASIAMRPN	0.34046	0.25717	2	4
DAT	0.53705	0.095122	5	15
DIMP	0.26387	0.4228	1	1
DSST	0.61271	0.10525	5	16
ECO	0.38545	0.25506	2	7
IBCCF	0.49876	0.19836	3	10
KCF	0.69388	0.060618	7	18
LADCF	0.42805	0.26665	2	8
MCCT	0.51724	0.13249	4	14
MDNET	0.41263	0.25184	2	6
SAMF	0.53949	0.14862	4	13
SIAMRPN++	0.32137	0.32195	2	2
STAPLE	0.62446	0.082567	6	17
STRCF	0.43885	0.23371	3	11

Table 12: Summarizing results: trackers grouped 1 and trackers ranked 1 (underlined and in bold) in all benchmark experiments.

Dataset	AOR	FR
OTB	**ECO**, LADCF, STRCF, DIMP, CFWCR, ATOM	**STRCF**, LADCF, ECO, DIMP, MDNET, CFWCR, ATOM
VOT-ST	**DIMP**	**DIMP**, SIAMRPN++, MDNET
NfS	**DIMP**, ATOM	**DIMP**, ATOM
VOT-LT	**DIMP**	**DIMP**
OTB+VOT-ST+NfS	**DIMP**, ATOM	**DIMP**, ATOM

2.4.7 Analysis and Discussion

2.4.7.1 Which are the Best Performing Trackers?

We summarize in Table 12 which trackers achieve rank 1 (i.e., $r_i = 1$) at the end of our iterative algorithm and which ones achieve group 1 (i.e., $r_i^1 = 1$) after the first iteration. We see that ECO, DIMP, MDNET, and STRCF achieve rank 1 in either accuracy or robustness, and that ATOM, CFWCR, ECO, DIMP, LADCF, and MDNET achieve group 1 for both AOR and FR, simultaneously, across at least one dataset. When all three short-term datasets are merged, ATOM and DIMP are placed in group 1, and DIMP is ranked 1 for both AOR and FR.

ATOM and DIMP ranks are also preserved when looking into the four individual experiments (OTB-100, VOT2018-ST, NfS-30, and VOT2018-LT). By counting how often a tracker achieves group 1 in AOR and FR measures across all four benchmarks, we see that DIMP scores 8/8, followed by ATOM with 4/8, then CFWCR, ECO, LADCF, MDNET, and STRCF with 2/8, and finally SIAMRPN++ with 1/8.

Table 12 confirms that deep-learning trackers (here ATOM and DIMP), correlation filter-based trackers (here LADCF and STRCF), or a combination of both tracking principles (here CFWCR and ECO) well compete against each other. We also note from Table 6 that the fastest of those top trackers are ATOM and DIMP, each running at 25.6 and 27.3 FPS, respectively. Overall, the top tracker in our benchmark is DIMP.

2.4.7.2 Detailed Output of Proposed Algorithm

In the proposed algorithm, each tested tracker i is evaluated over each video sequence l to yield scores $\{s_i\}$. We calculate the MAD threshold on two rounds to filter out top performing and second top performing trackers from the rest of $\{t_i\}$. Table 13 illustrates how our scoring algorithm works, where the performance measure is AOR, and all test sequences with their calculated MAD thresholds d_q (denoted thresh1 and thresh2 in Table 13) for best and second best scoring are displayed.

2.5 Conclusion

Numerous trackers are proposed yearly and compete with the state-of-the-art; our goal in this study was to identify trackers which were of similar AOR and FR performances across all test sequences. Our motivation for this chapter was to provide a tracker ranking protocol that accounted for data dispersion and did not only rely on averaging average results. Our proposed method scored and ranked trackers using the MAD dispersion estimator and calculated at each sequence the similarities in the quality

Table 13: Average AOR per tracker and per sequence and calculated deviation thresholds d_q for best and second best scores, for the OTB-100 dataset.

Sequences	ATOM	CFWCR	CSRDCF	CREST	DASIAMRPN	DAT	DIMP	DLST	DSST	ECO	IBCCF	KCF	LADCF	MCCT	MDNET	SAMF	SIAMFC	SIAMRPN++	STAPLE	STRCF	thresh1	thresh2
Basketball	0.4677	0.7551	0.4393	0.6886	0.36664	0.7443	0.6614	0.3212	0.5587	0.7379	0.7499	0.6764	0.7167	0.78	0.717	0.736	0.7667	0.62972	0.7113	0.7838	0.0468	0.0556
Biker	0.6911	0.6009	0.3621	0.3476	0.34732	0.3947	0.701	0.3432	0.2745	0.5113	0.6392	0.2495	0.4517	0.4048	0.3632	0.2424	0.4799	0.41943	0.2566	0.3822	0.0774	0.0562
Bird1	0.2485	0.1345	0.0297	0.3096	0.17351	0.2436	0.2128	0.0269	0.1163	0.1718	0.1832	0.0538	0.2547	0.206	0.2044	0.1017	0.2147	0.25732	0.1885	0.1914	0.0545	0.0401
Bird2	0.65	0.6445	0.5905	0.7804	0.59036	0.7427	0.6426	0.631	0.6158	0.809	0.6931	0.5751	0.4693	0.6631	0.7298	0.5023	0.6373	0.58976	0.7813	0.4479	0.0517	0.0407
BlurBody	0.8004	0.7115	0.6759	0.752	0.56138	0.3777	0.8287	0.675	0.4614	0.7267	0.6138	0.6722	0.6879	0.737	0.7938	0.6936	0.7193	0.58341	0.7279	0.6982	0.0366	0.0388
BlurCar1	0.8333	0.8572	0.7227	0.8556	0.64219	0.0144	0.8492	0.5931	0.7671	0.8697	0.824	0.8024	0.8545	0.5376	0.8212	0.8536	0.8028	0.64331	0.5301	0.8639	0.0451	0.1196
BlurCar2	0.8625	0.9019	0.8425	0.8582	0.65095	0.5071	0.8655	0.8558	0.904	0.9014	0.8688	0.7596	0.8958	0.8779	0.8643	0.8697	0.873	0.65665	0.8927	0.8922	0.0248	0.011
BlurCar3	0.8679	0.8721	0.84	0.836	0.57898	0.1808	0.8809	0.8183	0.8418	0.8666	0.8597	0.8109	0.8714	0.8379	0.8449	0.8474	0.8314	0.57485	0.8312	0.8807	0.0242	0.0131
BlurCar4	0.8677	0.8961	0.8897	0.8544	0.66203	0.8073	0.8724	0.8309	0.8973	0.8804	0.8904	0.8391	0.901	0.8742	0.8382	0.857	0.8813	0.67458	0.8913	0.8951	0.0204	0.0233
BlurFace	0.8211	0.8689	0.8691	0.8209	0.51862	0.4683	0.862	0.8486	0.8721	0.8649	0.8639	0.8367	0.8713	0.8828	0.8236	0.8701	0.8267	0.5476	0.8508	0.8779	0.0177	0.0242
BlurOwl	0.822	0.7761	0.7888	0.7928	0.43823	0.7712	0.8509	0.0791	0.2017	0.8283	0.7569	0.195	0.8421	0.7889	0.7814	0.2001	0.7716	0.46126	0.4323	0.8491	0.0614	0.1746
Board	0.1837	0.7033	0.8022	0.7737	0.69728	0.0446	0.5136	0.6405	0.7333	0.8056	0.7578	0.6486	0.7932	0.6748	0.8357	0.7682	0.7853	0.33567	0.5867	0.8006	0.0763	0.0737
Bolt	0.0205	0.0114	0.6677	0.6422	0.64635	0.7388	0.6593	0.6329	0.7224	0.6313	0.5938	0.6786	0.6882	0.7022	0.7159	0.7205	0.5507	0.53544	0.7808	0.6887	0.0455	0.0426
Bolt2	0.6154	0.5692	0.6758	0.0109	0.58653	0.4985	0.6167	0.6859	0.0113	0.5626	0.4557	0.0113	0.5468	0.7224	0.4897	0.0115	0.6693	0.36089	0.681	0.5377	0.1068	0.0838
Box	0.7579	0.3314	0.3413	0.6954	0.75218	0.0531	0.7993	0.3249	0.3443	0.7572	0.3257	0.3022	0.7416	0.3836	0.3301	0.7344	0.7963	0.70673	0.3597	0.7552	0.2111	0.0121
Boy	0.8392	0.8618	0.7272	0.8327	0.59779	0.6804	0.8534	0.8379	0.8394	0.8654	0.8208	0.7775	0.8519	0.7892	0.8262	0.7509	0.7947	0.61156	0.7996	0.838	0.0294	0.0401
Car1	0.7946	0.8553	0.6513	0.2061	0.70012	0.0585	0.5882	0.1661	0.63	0.8569	0.8435	0.1396	0.8851	0.6908	0.7581	0.4682	0.7329	0.70808	0.6596	0.8748	0.1276	0.0781
Car2	0.8544	0.854	0.8321	0.8337	0.78247	0.0654	0.8363	0.8372	0.912	0.8765	0.9002	0.6839	0.9264	0.8004	0.8679	0.8517	0.8613	0.7558	0.8576	0.8811	0.0222	0.0202
Car4	0.8772	0.8797	0.8208	0.6611	0.81316	0.0058	0.873	0.781	0.8964	0.8722	0.8829	0.4837	0.8755	0.8402	0.8035	0.7479	0.8514	0.86984	0.8735	0.8865	0.0241	0.0517
Car24	0.8274	0.8339	0.4241	0.5953	0.77783	0.3567	0.5585	0.7477	0.463	0.9074	0.8691	0.4269	0.8902	0.8086	0.7914	0.4916	0.8056	0.76987	0.4566	0.9039	0.1232	0.0676
CarDark	0.7206	0.8665	0.7333	0.7603	0.66995	0.0309	0.7463	0.7835	0.8451	0.8785	0.5956	0.6148	0.8743	0.8519	0.7777	0.7294	0.839	0.34116	0.871	0.8576	0.0858	0.0539
CarScale	0.89	0.6938	0.7117	0.6094	0.84263	0.4117	0.8857	0.5151	0.7435	0.7855	0.7418	0.4199	0.7382	0.6785	0.6421	0.5804	0.7056	0.73335	0.7777	0.72	0.0657	0.0379
ClifBar	0.3719	0.5741	0.6809	0.1828	0.40711	0.265	0.3792	0.1999	0.6412	0.586	0.5011	0.2598	0.6224	0.6617	0.6612	0.3906	0.3299	0.18774	0.4631	0.5929	0.1639	0.0885
Coke	0.5845	0.5261	0.5265	0.5904	0.55603	0.3898	0.5876	0.5073	0.5732	0.5495	0.6686	0.5498	0.6463	0.5637	0.5544	0.5984	0.5108	0.47644	0.5863	0.5678	0.0291	0.0292
Couple	0.8145	0.6808	0.5982	0.5989	0.52072	0.4151	0.803	0.5821	0.0904	0.6706	0.5058	0.2009	0.7304	0.6167	0.6151	0.411	0.4997	0.52105	0.5307	0.6887	0.0874	0.0844
Coupon	0.4742	0.9001	0.9026	0.3836	0.33336	0.863	0.4007	0.2574	0.8982	0.3803	0.5578	0.9443	0.9318	0.9101	0.3471	0.9353	0.8523	0.3953	0.8994	0.3698	0.2305	0.0268
Crossing	0.8329	0.8078	0.7042	0.768	0.79737	0.6634	0.8394	0.7647	0.7906	0.8053	0.7655	0.7102	0.8055	0.8019	0.7475	0.7559	0.7877	0.78113	0.7768	0.8245	0.0209	0.0209
Crowds	0.7381	0.7187	0.7143	0.6707	0.64382	0.0212	0.0703	0.7344	0.731	0.7074	0.7139	0.794	0.7334	0.6712	0.1068	0.7278	0.7734	0.73752	0.8027	0.7119	0.0213	0.0218
Dancer	0.7388	0.7771	0.7596	0.7444	0.70196	0.5726	0.7614	0.747	0.7718	0.7895	0.7629	0.6467	0.7217	0.7897	0.712	0.7301	0.76	0.67498	0.7766	0.7514	0.025	0.0212
Dancer2	0.7537	0.7379	0.7385	0.727	0.77371	0.7008	0.7549	0.7749	0.7633	0.7452	0.7723	0.7739	0.7324	0.7817	0.7949	0.7758	0.728	0.66719	0.7851	0.7227	0.0211	0.0155
David	0.7252	0.8426	0.7782	0.7009	0.51751	0.4705	0.8166	0.7911	0.8099	0.8476	0.7878	0.5384	0.8209	0.7773	0.7775	0.7054	0.7566	0.62165	0.7941	0.7737	0.0414	0.0345
David2	0.6826	0.799	0.7472	0.7557	0.74994	0.3592	0.776	0.7971	0.8105	0.8321	0.77	0.8279	0.8193	0.8264	0.6895	0.8057	0.7157	0.61962	0.7933	0.8613	0.0361	0.0395
David3	0.7636	0.781	0.737	0.7444	0.72466	0.6904	0.6103	0.7263	0.4845	0.7687	0.7922	0.7723	0.7772	0.7829	0.7588	0.7867	0.3949	0.70357	0.7727	0.7709	0.023	0.0342
Deer	0.5363	0.7824	0.7655	0.7354	0.40675	0.1185	0.6127	0.6318	0.642	0.7869	0.7578	0.6235	0.8185	0.7638	0.7036	0.6948	0.7356	0.36769	0.777	0.8084	0.0725	0.0836
Diving	0.5864	0.1111	0.3506	0.1364	0.56588	0.2996	0.6811	0.4044	0.214	0.296	0.331	0.3169	0.2696	0.1812	0.3989	0.2285	0.1072	0.57169	0.2445	0.2195	0.0925	0.082
Dog	0.3649	0.6088	0.5639	0.4417	0.48379	0.3558	0.3484	0.5294	0.5221	0.5153	0.5812	0.3505	0.6087	0.5468	0.5148	0.4867	0.5813	0.47948	0.5387	0.6104	0.0538	0.0439
Dog1	0.8599	0.8117	0.6768	0.7388	0.75522	0.1101	0.8522	0.8394	0.7602	0.8348	0.7361	0.551	0.8458	0.7719	0.8249	0.6881	0.7993	0.73121	0.8173	0.8405	0.0516	0.0357
Doll	0.8138	0.8615	0.8308	0.6145	0.68584	0.3897	0.7793	0.8354	0.853	0.8618	0.8438	0.5345	0.8284	0.8511	0.8385	0.6314	0.8009	0.72174	0.8463	0.8345	0.0261	0.0561
DragonBaby	0.7683	0.7125	0.4664	0.5963	0.45135	0.2969	0.7658	0.4137	0.0534	0.7171	0.534	0.3125	0.6651	0.6021	0.7198	0.5459	0.3351	0.49174	0.5019	0.4988	0.1257	0.0661
Dudek	0.8765	0.7589	0.8287	0.7961	0.86281	0.1696	0.8772	0.8505	0.7881	0.8621	0.8567	0.7275	0.8388	0.7615	0.8615	0.8216	0.8324	0.82852	0.7158	0.825	0.0332	0.0346
FaceOcc1	0.5259	0.7696	0.7677	0.7083	0.59798	0.7174	0.5171	0.622	0.7657	0.8015	0.7241	0.7744	0.7862	0.7782	0.6861	0.7944	0.7772	0.61039	0.7931	0.7706	0.0313	0.0714
FaceOcc2	0.648	0.7424	0.7946	0.7065	0.67094	0.2759	0.7366	0.5057	0.7801	0.7469	0.7115	0.7511	0.745	0.672	0.7584	0.7536	0.5792	0.66378	0.7773	0.7398	0.0354	0.0396
Fish	0.8233	0.8599	0.8831	0.7989	0.72863	0.0823	0.8354	0.8305	0.8037	0.8676	0.8318	0.8394	0.8522	0.8	0.8583	0.8257	0.8666	0.74795	0.7839	0.8546	0.0281	0.0234
FleetFace	0.672	0.6355	0.6441	0.7387	0.74579	0.3359	0.773	0.6208	0.6341	0.7451	0.6202	0.5893	0.7288	0.6597	0.7847	0.6359	0.6358	0.67347	0.6474	0.7061	0.033	0.0259
Football	0.7149	0.5547	0.5541	0.7383	0.68403	0.0366	0.6963	0.1601	0.5523	0.6865	0.7209	0.5522	0.7399	0.6103	0.1541	0.5992	0.5279	0.6297	0.5829	0.7162	0.0805	0.0378
Football1	0.3465	0.5202	0.5586	0.679	0.57983	0.6607	0.4036	0.4155	0.5004	0.5146	0.6493	0.7101	0.7892	0.6828	0.7603	0.6821	0.4426	0.57661	0.7433	0.7329	0.0978	0.1021
Freeman1	0.7065	0.5163	0.5885	0.6284	0.65702	0.1938	0.7171	0.6192	0.2449	0.528	0.6601	0.2146	0.6473	0.5338	0.6674	0.3027	0.6429	0.58053	0.6572	0.5997	0.0543	0.0606
Freeman3	0.7979	0.7911	0.3299	0.3531	0.74433	0.2961	0.7893	0.4305	0.3367	0.7894	0.6258	0.3243	0.8199	0.3488	0.7503	0.2949	0.7608	0.66975	0.3375	0.7413	0.1611	0.0144
Freeman4	0.4613	0.5685	0.4898	0.4383	0.49299	0.2283	0.6884	0.1227	0.4562	0.5712	0.6193	0.1752	0.7191	0.6337	0.4992	0.4473	0.5398	0.26858	0.4086	0.6434	0.0813	0.0784
Girl	0.7576	0.7974	0.1781	0.7696	0.71042	0.4596	0.7884	0.6815	0.3921	0.7974	0.6827	0.5457	0.743	0.6976	0.7547	0.6535	0.7228	0.73439	0.5364	0.7117	0.052	0.0465
Girl2	0.5109	0.0643	0.0621	0.5269	0.33492	0.4169	0.4543	0.6722	0.0952	0.752	0.0627	0.0579	0.6351	0.6346	0.2443	0.6354	0.0623	0.351	0.0848	0.638	0.2528	0.03
Gym	0.5896	0.5088	0.3796	0.47	0.72501	0.3754	0.6833	0.4053	0.2547	0.4792	0.5273	0.4263	0.4669	0.4335	0.5116	0.4998	0.5051	0.71314	0.3506	0.4047	0.061	0.0446
Human2	0.7814	0.7678	0.3999	0.6502	0.75592	0.2654	0.7815	0.161	0.4144	0.793	0.7352	0.2471	0.7604	0.7338	0.7514	0.6522	0.6201	0.70129	0.7315	0.7521	0.0488	0.0994
Human3	0.017	0.626	0.0193	0.0452	0.016867	0.0653	0.1528	0.012	0.0234	0.6239	0.0227	0.0051	0.6447	0.6187	0.5383	0.0053	0.0079	0.015638	0.024	0.6262	0.0185	0.0181
Human4	0.3769	0.6175	0.6293	0.337	0.36914	0.4295	0.409	0.3423	0.6443	0.6043	0.668	0.3702	0.6506	0.5459	0.6053	0.6804	0.6129	0.6336	0.6767	0.6668	0.061	0.0796
Human5	0.769	0.8137	0.5337	0.5237	0.67836	0.0313	0.7994	0.6405	0.1994	0.8134	0.7106	0.1833	0.8	0.7839	0.7407	0.1844	0.7953	0.62328	0.6987	0.7943	0.0927	0.165
Human6	0.7583	0.8187	0.7546	0.2979	0.71202	0.2694	0.7673	0.6842	0.3784	0.8035	0.7124	0.2072	0.8172	0.7983	0.7008	0.4651	0.6574	0.68089	0.8156	0.8084	0.0887	0.055
Human7	0.8489	0.828	0.7114	0.6887	0.47832	0.1582	0.8644	0.534	0.3605	0.848	0.4026	0.2837	0.845	0.3847	0.8168	0.3248	0.3845	0.48911	0.8107	0.8647	0.2268	0.0768
Human8	0.841	0.7548	0.7037	0.0594	0.76247	0.3826	0.8495	0.2462	0.8248	0.7901	0.7627	0.5107	0.8028	0.7038	0.6646	0.5436	0.6741	0.74833	0.7687	0.801	0.0623	0.0737
Human9	0.8508	0.4165	0.6938	0.5326	0.49194	0.1605	0.8378	0.6944	0.3196	0.8359	0.3531	0.393	0.503	0.6324	0.6152	0.3268	0.1225	0.49585	0.4622	0.5302	0.1397	0.1232
Ironman	0.1224	0.3943	0.1183	0.4464	0.43699	0.055	0.4862	0.2982	0.121	0.4076	0.3764	0.1408	0.0177	0.1886	0.4499	0.1292	0.1179	0.49553	0.1225	0.1731	0.1216	0.0126
Jogging_1	0.8236	0.7918	0.7	0.164	0.66643	0.1765	0.7976	0.2321	0.1851	0.8009	0.7556	0.1854	0.8173	0.7583	0.735	0.7939	0.1687	0.6164	0.1773	0.8126	0.0975	0.0189
Jogging_2	0.8451	0.7453	0.7377	0.7559	0.60195	0.1447	0.7756	0.7711	0.1397	0.7196	0.6965	0.1241	0.7245	0.7467	0.7601	0.7743	0.6284	0.73959	0.1382	0.7365	0.0356	0.0346
Jump	0.0852	0.0277	0.1634	0.0763	0.54542	0.0314	0.653	0.2274	0.0914	0.0599	0.1801	0.0979	0.0363	0.0922	0.1068	0.094	0.1288	0.12535	0.0595	0.0892	0.0334	0.0326
Jumping	0.5965	0.7599	0.7234	0.6797	0.40176	0.0477	0.6879	0.2255	0.1369	0.7275	0.6887	0.2746	0.6665	0.6025	0.7255	0.2533	0.2472	0.47472	0.2387	0.6852	0.1271	0.1164
KiteSurf	0.7672	0.7365	0.3351	0.7204	0.27218	0.2419	0.7358	0.3055	0.3225	0.7632	0.3195	0.4747	0.7788	0.3669	0.7164	0.3443	0.3939	0.54496	0.6997	0.3642	0.1454	0.0307
Lemming	0.793	0.7018	0.2322	0.774	0.69686	0.5759	0.7863	0.6877	0.3198	0.8146	0.6389	0.3842	0.7957	0.751	0.737	0.769	0.65	0.68908	0.2384	0.7598	0.0721	0.063
Liquor	0.5501	0.8854	0.7865	0.7206	0.64185	0.3362	0.5424	0.8106	0.4078	0.8793	0.7953	0.4317	0.8651	0.7705	0.8373	0.5192	0.827	0.48512	0.74	0.8518	0.1116	0.1106
Man	0.8561	0.8488	0.7941	0.8057	0.77241	0.506	0.8214	0.8072	0.8428	0.8105	0.843	0.8311	0.8327	0.8483	0.8085	0.8461	0.8873	0.7338	0.8512	0.8543	0.0219	0.0206
Matrix	0.528	0.5003	0.0936	0.4164	0.44095	0.0683	0.5279	0.3574	0.1977	0.5004	0.3776	0.119	0.2323	0.234	0.2931	0.2456	0.1966	0.51919	0.1169	0.2175	0.1487	0.0756
Mhyang	0.5986	0.8095	0.7805	0.8328	0.6754	0.59	0.7693	0.8009	0.8022	0.8381	0.8132	0.7966	0.8254	0.821	0.8433	0.8663	0.732	0.76342	0.7824	0.8116	0.0276	0.0244
MotorRolling	0.772	0.6818	0.09	0.1327	0.66016	0.0886	0.75	0.6043	0.0917	0.0941	0.6858	0.0925	0.0916	0.0941	0.5879	0.0925	0.1583	0.16694	0.0961	0.0943	0.0251	0.0059
MountainBike	0.7936	0.673	0.719	0.7087	0.82445	0.0964	0.7862	0.7141	0.7303	0.6997	0.7407	0.7116	0.6502	0.6805	0.7269	0.7032	0.6936	0.70857	0.7172	0.643	0.0185	0.0183
Panda	0.6164	0.2237	0.4832	0.5111	0.59836	0.5557	0.6234	0.5546	0.1269	0.3138	0.3603	0.1589	0.4528	0.4946	0.5871	0.2844	0.2719	0.59155	0.313	0.3904	0.1269	0.0834
RedTeam	0.728	0.7206	0.5136	0.6139	0.72229	0.4589	0.727	0.6647	0.5641	0.7328	0.6613	0.5006	0.7588	0.6761	0.6023	0.6577	0.6542	0.38734	0.568	0.764	0.068	0.059
Rubik	0.7103	0.7326	0.3913	0.6891	0.70004	0.4908	0.7144	0.6033	0.722	0.8072	0.6897	0.6137	0.5196	0.3535	0.6843	0.6558	0.7446	0.64475	0.6804	0.4209	0.045	0.0417
Shaking	0.7967	0.7531	0.1969	0.7563	0.76638	0.0514	0.6391	0.0731	0.7167	0.019	0.6336	0.0396	0.76	0.7375	0.7001	0.1209	0.7251	0.70718	0.0393	0.7483	0.0636	0.1778
Singer1	0.7869	0.8752	0.8562	0.4487	0.79989	0.3332	0.7764	0.4037	0.8245	0.8649	0.7595	0.3555	0.8794	0.7954	0.797	0.5395	0.8283	0.68617	0.815	0.886	0.0644	0.0555
Singer2	0.1041	0.0339	0.7279	0.0322	0.62035	0.0191	0.6996	0.6135	0.7795	0.0373	0.0359	0.7324	0.0335	0.7829	0.7341	0.0412	0.0376	0.61971	0.7943	0.8106	0.1858	0.0046
Skater	0.7734	0.2257	0.605	0.5983	0.71239	0.5676	0.7717	0.5508	0.568	0.5629	0.6427	0.6105	0.5229	0.6153	0.6225	0.6063	0.6291	0.65266	0.6046	0.5239	0.0379	0.037
Skater2	0.7549	0.6115	0.4416	0.5951	0.67473	0.6075	0.7138	0.5017	0.1748	0.4868	0.6518	0.5661	0.5921	0.6012	0.6943	0.564	0.5552	0.67684	0.4544	0.5864	0.0697	0.0311
Skating1	0.505	0.3988	0.3003	0.5762	0.72058	0.0734	0.5313	0.4665	0.5314	0.5161	0.5511	0.4893	0.5356	0.469	0.4724	0.6216	0.444	0.70463	0.5924	0.4444	0.0548	0.0456
Skating2_1	0.4875	0.4404	0.5411	0.0354	0.40309	0.4013	0.4815	0.4589	0.1048	0.4926	0.5128	0.4197	0.4288	0.5461	0.5193	0.236	0.3685	0.37051	0.3349	0.4764	0.0611	0.0475
Skating2_2	0.5176	0.2564	0.4056	0.3683	0.54788	0.0908	0.4696	0.1008	0.1435	0.3534	0.4112	0.375	0.3918	0.4313	0.5608	0.4279	0.3699	0.4463	0.2197	0.4632	0.0561	0.0563
Skiing	0.7271	0.4463	0.0671	0.4659	0.47432	0.4884	0.5613	0.5514	0.0658	0.4473	0.4087	0.0507	0.0756	0.0947	0.5192	0.0531	0.0753	0.5033	0.1044	0.0703	0.2167	0.0476
Soccer	0.2509	0.532	0.2097	0.3474	0.42536	0.1794	0.5158	0.1468	0.4957	0.5271	0.2088	0.4221	0.1775	0.198	0.4219	0.1786	0.186	0.43438	0.2143	0.4493	0.1222	0.0185
Subway	0.8051	0.6986	0.6256	0.7239	0.1884	0.7228	0.8014	0.387	0.1785	0.6983	0.7167	0.7546	0.7278	0.7783	0.6739	0.758	0.184	0.61125	0.7448	0.7557	0.0421	0.0561
Surfer	0.7356	0.6919	0.6721	0.6608	0.55777	0.4374	0.5191	0.6649	0.2321	0.7039	0.7198	0.4652	0.7263	0.7036	0.7603	0.6785	0.6489	0.43807	0.2194	0.6855	0.0546	0.0489
Suv	0.7653	0.8408	0.8302	0.7808	0.60314	0.0974	0.723	0.6951	0.8116	0.7841	0.8279	0.883	0.8363	0.8437	0.7151	0.8584	0.4734	0.35211	0.8332	0.7932	0.0536	0.0702
Sylvester	0.7031	0.5829	0.5525	0.7429	0.74022	0.2344	0.7279	0.7532	0.6288	0.6511	0.7345	0.6439	0.748	0.6166	0.7367	0.6296	0.6873	0.74958	0.5552	0.7484	0.053	0.0341
Tiger1	0.258	0.6065	0.7417	0.6272	0.54417	0.4206	0.2566	0.6688	0.6204	0.683	0.6343	0.7856	0.7697	0.746	0.6732	0.617	0.1458	0.39715	0.7636	0.7733	0.1131	0.0762
Tiger2	0.5928	0.438	0.6777	0.532	0.48315	0.4484	0.6161	0.615	0.326	0.6785	0.5512	0.3541	0.5164	0.4936	0.637	0.5368	0.4345	0.45235	0.6832	0.5222	0.0833	0.046
Toy	0.6245	0.6346	0.6735	0.6083	0.58863	0.0578	0.6898	0.7048	0.7022	0.698	0.5485	0.4755	0.5744	0.6249	0.6641	0.615	0.6827	0.56143	0.6591	0.6217	0.0495	0.0263
Trans	0.7404	0.5471	0.4327	0.536	0.69356	0.389	0.7944	0.5443	0.4616	0.4863	0.6054	0.3845	0.5176	0.4951	0.5783	0.5321	0.5304	0.67388	0.5519	0.5453	0.0495	0.0424
Trellis	0.7075	0.7506	0.7275	0.725	0.70097	0.5689	0.5846	0.5658	0.781	0.7706	0.7335	0.631	0.7472	0.8114	0.7835	0.8383	0.8123	0.63819	0.8378	0.7759	0.0419	0.047
Twinnings	0.7783	0.7825	0.7683	0.6333	0.5832	0.4884	0.763	0.3977	0.7904	0.6323	0.7731	0.5659	0.8053	0.7626	0.7558	0.7061	0.5391	0.71949	0.7666	0.7611	0.0355	0.0622
Vase	0.7192	0.5415	0.4424	0.4883	0.5898	0.1973	0.7135	0.5307	0.5368	0.5377	0.3812	0.3168	0.4372	0.4883	0.4958	0.4482	0.5223	0.61136	0.5466	0.4694	0.0503	0.0476
Walking	0.8121	0.7123	0.7515	0.6521	0.79364	0.5468	0.8089	0.7082	0.7442	0.7397	0.7787	0.5301	0.7441	0.7621	0.7251	0.7115	0.7786	0.79025	0.7197	0.7366	0.0321	0.02
Walking2	0.8249	0.7918	0.3289	0.3395	0.28195	0.242	0.3721	0.6089	0.8	0.7969	0.7878	0.395	0.8018	0.3401	0.7963	0.6514	0.804	0.29663	0.7775	0.8149	0.103	0.0492
Woman	0.6933	0.8076	0.7109	0.7429	0.61472	0.1329	0.6787	0.7353	0.6943	0.788	0.6685	0.7051	0.7641	0.7691	0.7993	0.6744	0.7702	0.63467	0.7331	0.788	0.0473	0.0247

data. We then ran an iterative algorithm to cluster all trackers into groups and sub-groups of similar performance. To extensively test our method, we ranked our tested trackers over three short-term datasets, individually and altogether, and one long-term dataset. Our observations over all ranking outputs indicated that DIMP was the best performing tracker, followed by ATOM, and then by a group of similarly performing trackers CFWCR, ECO, LADCF, MDNET, and STRCF.

We have shown in this chapter that data dispersion within tracking results can play an important role in estimating a tracker's level of performance. Even though our method proved to be robust, it was purely applied. In the next chapter, we gain inspiration from the use of robust norms applied in image denoising and look into employing robust statistics to propose a scoring and ranking method that has more scientific footing.

Chapter 3

Robust Error Norm-Based Scoring and Ranking of Object Trackers

3.1 Introduction

In Chapter 2, we achieve a robust scoring and ranking method to evaluate trackers and rank them in groups of similar performance, the motivation being that current tracking evaluation only relies on averaging performance measures which are averaged across each video of a dataset [91, 48]. Using the median absolute deviation (MAD), our evaluation estimates on a per-sequence basis which trackers perform similarly and scores them accordingly. Even though we have shown that our similarity-based ranking is robust, in this chapter, we go a step further and aim to use robust statistics concepts used in anisotropic diffusion for image denoising to propose a theoretically sold approach, where we apply robust error norms to score and rank trackers with respect to each other.

In the rest of the chapter, section 3.2 presents the concepts which motivate our work; section 3.3 briefly mentions related work; section 3.4 proposes our new ranking method; section 3.5 provides simulation results and compares its ranks with the

similarity-based ranking's results; finally, section 3.6 concludes the chapter.

3.2 Robust Error Norms in Image Denoising

Error norm has been used in image processing before, for example, in anisotropic diffusion for image denoising [7, 76]. Given a noisy image $I = I_{clean} + noise$ where $noise$ is assumed to be additive white Gaussian noise, let s be the center pixel and p be a neighbor of s within a neighborhood η_s. Let I_s and I_p be the intensities of pixels s and p, respectively. For all neighbors p in η_s, we define the difference, or error, $e_p = I_p - I_s$ as the image gradient from pixels s to p. Therefore, in a piecewise constant image region, error e_p is small and zero-mean normally distributed. An optimal estimator for non-noisy (clean) I_s minimizes e_p^2.

The MAD allows to determine whether the median reliably represents the values within the set: the larger the MAD, the greater variability there is, and the less representative the median becomes in the set. Let c be the scale factor of a known distribution family of a set of data $\{e_p\}$. The scale factor is the inverse of the 75th percentile of the standard known cumulative distribution; for example, $c = 1.4826$ in a Gaussian distribution [66]. The MAD is used for the robust scale estimator $\sigma = c \cdot \text{MAD}\{e_p\}$ [75] in order to reject outliers, or neighbors p of pixel s in a noisy image I. Determining the clean image from I can be formulated as the following minimization problem [7]

$$\min_I \sum_{s \in I} \sum_{s \in \eta_s} \rho(e_p; \sigma), \tag{12}$$

where $\rho(\cdot)$ is an error norm which minimizes the effect of the outliers and is more robust than the quadratic norm [7] defined as

$$\rho(e; \sigma) = \frac{e^2}{\sigma^2}. \tag{13}$$

Assuming that pixels p are of intensity close to that of s, the optimization (12)

can be solved by gradient descent,

$$I_s^{t+1} = I_s^t + \lambda \cdot \frac{1}{|\eta_s|} \sum_{p \in \eta_s} \psi(e_p^t; \sigma), \tag{14}$$

where t is the iteration, $e_p^t = I_p - I_s^t$, $\psi(\cdot) = \rho'(\cdot)$ is an influence function, λ is a constant rate of diffusion, and $|\eta_s|$ is the cardinality of region η_s (i.e., its number of neighbors). The influence function ψ influences (reduces or increases) the contribution of I_p on the solution [35].

By defining the edge-stopping function $h(e) = \frac{\psi(e)}{e}$, (14) becomes the Perona and Malik anisotropic diffusion equation [70],

$$I_s^{t+1} = I_s^t + \lambda \cdot \frac{1}{|\eta_s|} \sum_{p \in \eta_s} h(e_p^t; \sigma) \cdot e_p^t. \tag{15}$$

The statistical interpretation of the Perona and Malik anisotropic diffusion equation provides a means for detecting edges between the piecewise constant image regions; denoising stops at outliers (edges) depending on the scale parameter σ.

A robust error norm is one that rejects outliers, that is, one whose influence function reduces the contribution of outliers [7, 35]. Among these error norms is the Lorentzian norm [7, 70]:

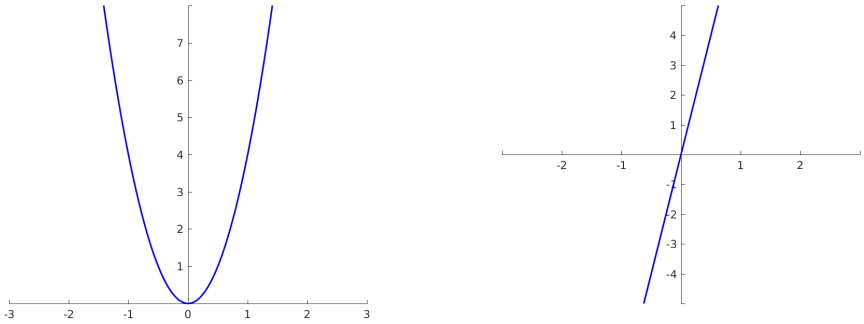
$$\rho(e; \sigma) = \log(1 + \frac{1}{2}\frac{e^2}{\sigma^2}), \tag{16}$$

$$\psi(e; \sigma) = e\frac{1}{1 + \frac{e^2}{2\sigma^2}}, \tag{17}$$

$$h(e; \sigma) = \frac{1}{1 + \frac{e^2}{2\sigma^2}}. \tag{18}$$

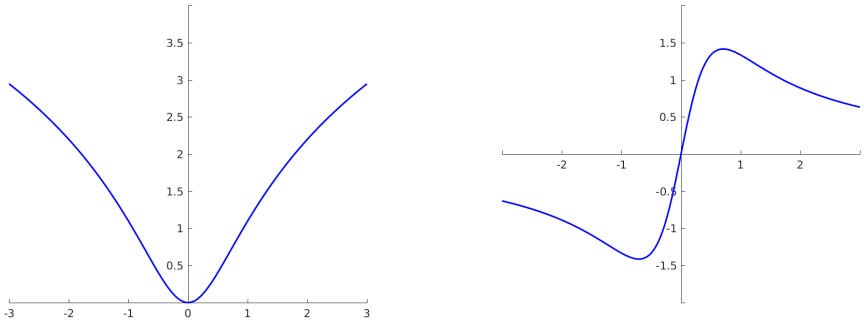
ρ is the error norm, $\psi(\cdot) = \rho'(\cdot)$ is the influence function, and $h(\cdot)$ is the edge-stopping function $h(\cdot)$. By substituting (18) in (14), one can replicate the Perona and Malik anisotropic diffusion equation [70], showing that anisotropic diffusion is the gradient descent of an estimation problem with a known robust error norm [7],

$$\begin{aligned} I_s^{t+1} &= I_s^t + \lambda \cdot \frac{1}{|\eta_s|} \sum_{p \in \eta_s} h(e_p^t; \sigma) \cdot e_p^t \\ &= I_s^t + \lambda \cdot \frac{1}{|\eta_s|} \sum_{p \in \eta_s} \frac{1}{1 + \frac{e^2}{2\sigma^2}} \cdot e_p. \end{aligned} \tag{19}$$



(a) $\rho(e, \sigma)$ (b) $\psi(e, \sigma)$

Figure 1: Quadratic error norm $\rho(e, \sigma)$ and influence $\psi(e, \sigma)$.



(a) $\rho(e, \sigma)$ (b) $\psi(e, \sigma)$

Figure 2: Lorentzian error norm $\rho(e, \sigma)$ and influence $\psi(e, \sigma)$.

Figures 1 and 2 show error norm and influence functions for the quadratic norm and the Lorentzian norm, respectively, for a zero-mean Gaussian distribution. Although the quadratic error norm $\rho(e, \sigma) = \frac{e^2}{\sigma^2}$ has an optimal local estimate of I_s^t, its influence is a boundless linearly increasing function which is sensitive to outliers. The Lorentzian error norm increases at a lower rate than the quadratic error norm and is much more robust to outliers. Its influence, however, does not go to zero [7].

3.3 Related Work

Work related to evaluation and ranking of trackers is cited in Chapter 2 section 2.2. Our similarity-based scoring and iterative ranking algorithm in Chapter 2 is the related work for the robust error norm-based ranking method we propose in this chapter. In fact, the MAD is used in both methods as a robust estimator of similarly performing trackers. In comparison, however, our proposed scoring and ranking method in this chapter uses robust statistics and error norms to achieve a scoring process that is statistically coherent. Different to the similarity-based method, our error norm-based scoring first computes a robust scale estimate to score higher neighboring trackers that are within the best region of performance. The scoring operation is achieved using an edge-stopping function which assigns the highest score to the best performing tracker and scores other trackers relatively to the top tracker, regardless of how high or low the performance mean of the top tracker is. Therefore, unlike our method in Chapter 2, our error norm-based scoring does not only assign scores to the best and second best performing trackers and it yields scores that are more spread across the domain $[0, \cdots, 1]$. Our grouping in this chapter also computes a robust scale estimate and groups trackers together if their performance differences are within the range described by that scale. In this work, we show that our proposed score is more representative of a tracker's performance than the widely used average of averages.

3.4 Proposed Tracker Evaluation Method

We borrow the idea of the use of error norms in image denoising to propose a robust method to estimate the performance of a tracker with respect to other trackers by relating its score and rank to an error norm. Two trackers can be considered neighbors in terms of their performance if the error, or the difference, between their performances

is not an outlier, that is, it remains within a robust scale defined as the MAD of the error. This can be interpreted as detecting the boundaries between the piecewise constant performance regions. Thus, trackers belong to a similar performing rank if their performance measures are similar to the best performance measure within that rank.

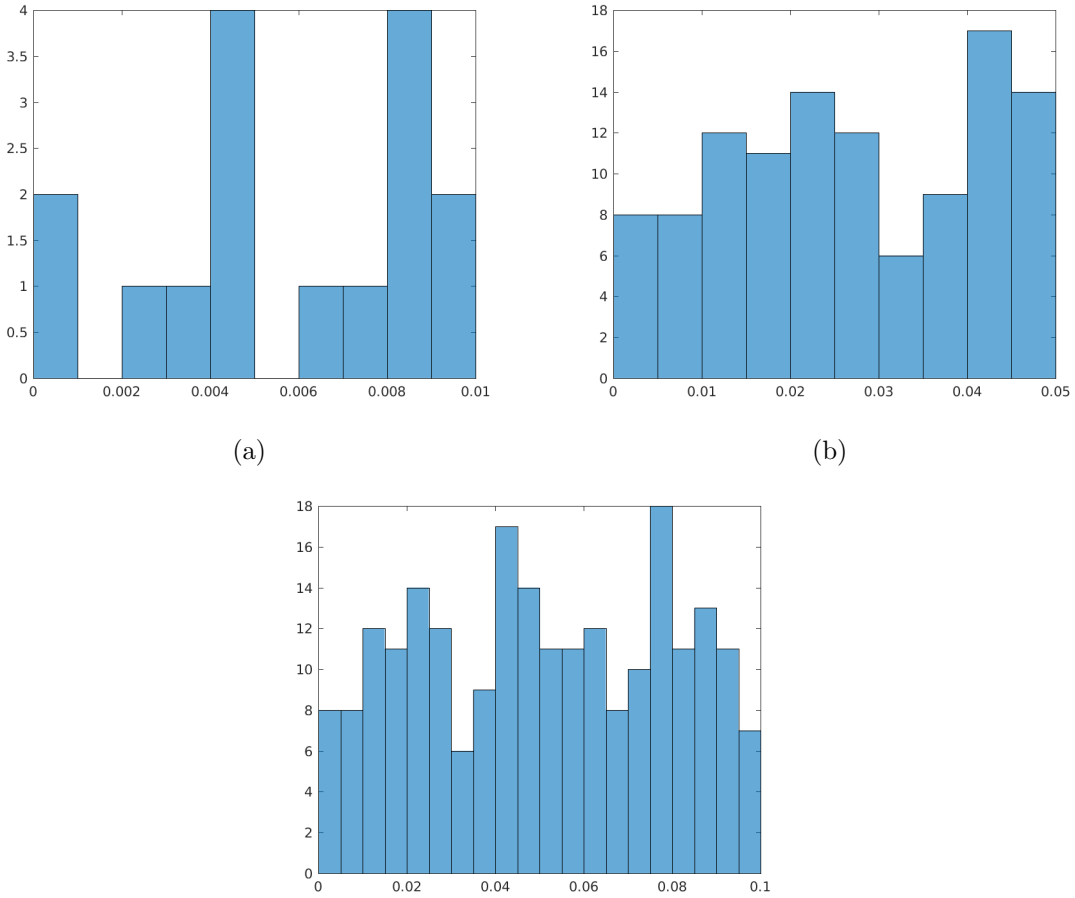
3.4.1 Robust Error Norm-Based Scoring

The input to the proposed scoring method is the performance (quality) data $\{q_{il}\}$ for all tested trackers $\{t_i, i = 1, \cdots, T\}$ over all test video sequences $\{v_l, l = 1, \cdots, L\}$, each having N_l frames $\{F_t, t = 1, \cdots, N_l\}$. Let the best performing tracker i over a sequence l yield q_{Bl}. For example, $q_{Bl} = \max\{\text{AOR}_i; i = 1, \cdots, T\}$ for the average overlap ratio, or $q_{Bl} = \min\{\text{FR}_i; i = 1, \cdots, T\}$ for the failure rate. Let $e_{il} = q_{Bl} - q_{il}$ be the difference in performances between the best tracker and other trackers in sequence v_l, where $0 \leq e_{il} \leq 1$. The set $\{e_{il}, i = 1, \cdots, T\}$ contains a population of sample errors where the difference in performance between the best tracker and its similarly performing (neighboring) trackers is small, i.e. $e_{il} \to 0$. The errors of neighboring trackers are from one distribution while the errors of far-neighbors are from another distribution. As can be seen on the histograms of Figure 3, the histogram (distribution) of AOR errors can be approximated as uniform as we decrease the range of the histogram.

We define a scoring function $h(e_{il})$ that assigns a high score when the error is small, i.e. $h(0) = \max\ h(e_{il})$, and computes a lower score the larger the error becomes, i.e. $\lim_{e_{il} \to 1} h(e_{il}) = 0$. We scale regions of neighboring trackers based on a robust scale estimator

$$\sigma_l = c_l \cdot \underset{1 \leq i \leq T}{\text{MAD}}\{e_{il}\}, \tag{20}$$

where σ_l is the robust scale of all errors e_{il} at sequence l, MAD (Median Absolute Deviation) is a robust measure of variability for detecting outliers in statistics, and



(a) (b)

(c)

Figure 3: Histograms of AOR differences across ranges (a) 0 to 0.01, (b) 0 to 0.05, and (c) 0 to 0.1, in the combined dataset OTB-100+VOT2018-ST+NfS-30.

c_l is a scale factor which depends on the distribution family. With respect to our assumption of uniform noise within each constant region of performance, $c_l = \sqrt{4/3} = 1.1547$ [66].

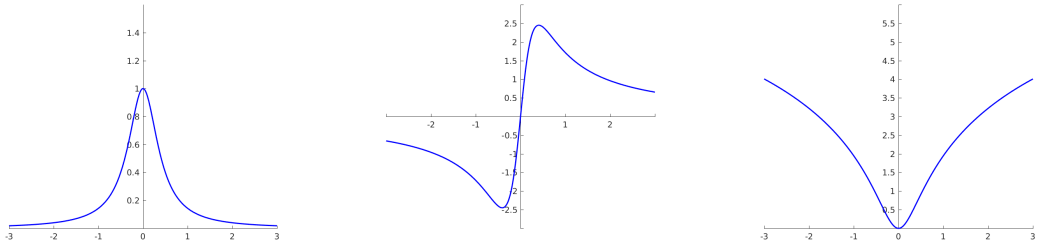
Error norms can be used in robust statistics to estimate sets of data which contain outliers. We look into three kinds of robust estimators, namely Lorentzian's edge-stopping (21), Huber's minmax (22), and Tukey's biweight (23) functions.

$$h_{Lorentz}(e_{il};\sigma_l) = \frac{1}{1+\frac{e^2}{2\sigma_l^2}} \tag{21}$$

$$h_{Huber}(e_{il};\sigma_l) = \begin{cases} \frac{1}{\sigma_l} & : |e_{il}| \leq \sigma_l \\ \frac{\text{sign}(e_{il})}{e_{il}} & : |e_{il}| \leq \sigma_l \end{cases} \tag{22}$$

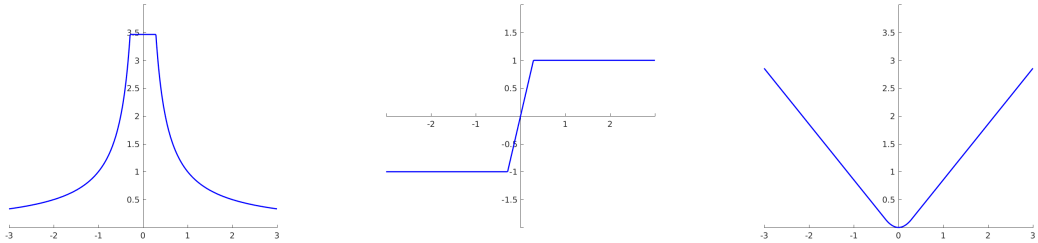
$$h_{Tukey}(e_{il};\sigma_l) = \begin{cases} \frac{1}{2}(1-\frac{e_{il}^2}{\sigma_l^2})^2 & : |e_{il}| \leq \sigma_l \\ 0 & : |e_{il}| \leq \sigma_l \end{cases} \tag{23}$$

Figures 4 to 6 show Lorentzian, Huber, and Tukey error norms with their corresponding error norms and influence functions for a uniform distribution sampled on the interval $[0,\cdots,1]$, the same interval AOR and FR performance measures are computed on. None of the magnitudes are re-scaled; however, we can re-scale Huber and Tukey functions so that they return values in the range $[0,\cdots,1]$, similar to the Lorentzian function. Each error norm is more robust than the quadratic norm and presents its own set of advantages. For instance, Tukey's biweight (23) descends all the way to zero, which means it can ignore the contributions of the worst performing trackers. Huber's minmax (22) is constant for small errors and can, therefore, assign the same highest score to trackers similar to the best performing tracker in a sequence. However, its scoring can get unstable and produce too many inliers when σ_l is high. The Lorenztian edge-stopping (21) delivers a good balance in-between Huber's minmax and Tukey's biweight. Thus, at a sequence l, we choose to assign a tracker i a score $s_{il} = h_{Lorentz}(e_{il};\sigma_l)$. As a result, our method assigns higher scores to trackers that have close performance to the top tracker and low scores to outliers



(a) $h(e_{il})$ (b) $\psi(e_{il})$ (c) $\rho(e_{il})$

Figure 4: Perona and Malik edge-stopping function and Lorentzian error norm.



(a) $h(e_{il})$ (b) $\psi(e_{il})$ (c) $\rho(e_{il})$

Figure 5: Huber's minmax.

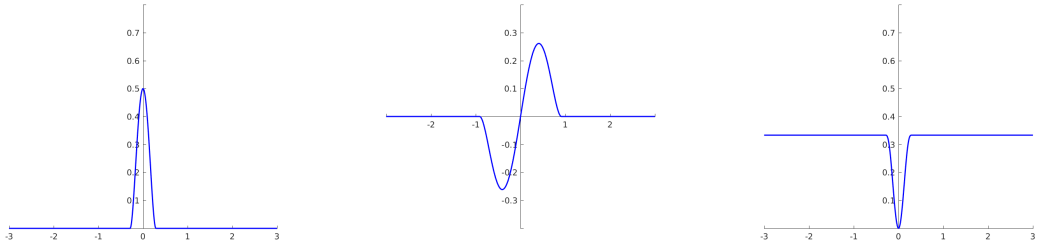
(far neighbors) for each test sequences.

Note that since (21) is a continuously decreasing function, if a tracker t_1 has a better mean performance than a tracker t_2 for an objective measure, then $s_{1l} > s_{2l}$.

The Lorentzian edge-stopping presents issues in cases where there are no outliers (for example when all FR values are zero); the MAD is zero, hence $\sigma_l = 0$ and $h_{Lorentz}(e_{il}; \sigma_l)$ is undefined. To solve this problem, we propose

$$\overset{\Uparrow}{s_{il}} = h_{Lorentz}(e_{il}; \sigma_l) \;\; = \begin{cases} \frac{1}{1+\frac{e^2}{2\sigma^2}} & : \sigma_l \neq 0 \\ q_{il} \cdot (1 - e_{il}) & : \sigma_l = 0, \end{cases} \tag{24}$$

where $\overset{\Uparrow}{s_{il}}$ designates the score of quality data q_{il} such that the best value for q_{il} is 1 (for example AOR). However, when the quality data q_{il} has 0 for its best value (for



(a) $h(e_{il})$ (b) $\psi(e_{il})$ (c) $\rho(e_{il})$

Figure 6: Tukey's biweight.

example FR), we propose

$$s_{il}^{\downarrow\downarrow} = h_{Lorentz}(e_{il};\sigma_l) \;= \begin{cases} \frac{1}{1+\frac{e^2}{2\sigma^2}} & : \sigma_l \neq 0 \\ (1-q_{il})\cdot(1-e_{il}) & : \sigma_l = 0. \end{cases} \tag{25}$$

After all tracker scores are computed over each sequence, we apply the mean to combine all scores s_{il} of each t_i over the L test sequences into one combined score for each tracker,

$$s_i = \underset{1\leq l\leq L}{\text{Mean}}\{s_{il}\}. \tag{26}$$

For (26), other measures of central tendency such as the median, midmean, and winsored mean can be used. However, the mean is a better measure of the score's center if we have a symmetric distribution of scores [84]. For this, we plot in Figure 7 the histograms of tracker scores $\{s_{il}\}$ for all trackers as well as for three groups of trackers (best, middle, and lowest) across the combined dataset OTB-100+VOT2018-ST+NfS-30. We indicate the mean of the scores on each histogram. Figure 7a shows that the distribution of scores over all trackers can be assumed symmetric; therefore, we choose the mean of scores as a robust sample estimator of center. A symmetric distribution can also be observed in the histograms of the middle and best trackers, in Figures 7c and 7d, respectively. Unsurprisingly, the histogram of the lowest trackers in Figure 7b is not as symmetric as the two other groups since trackers within the lowest category do not score as high as the trackers in the middle and best category.

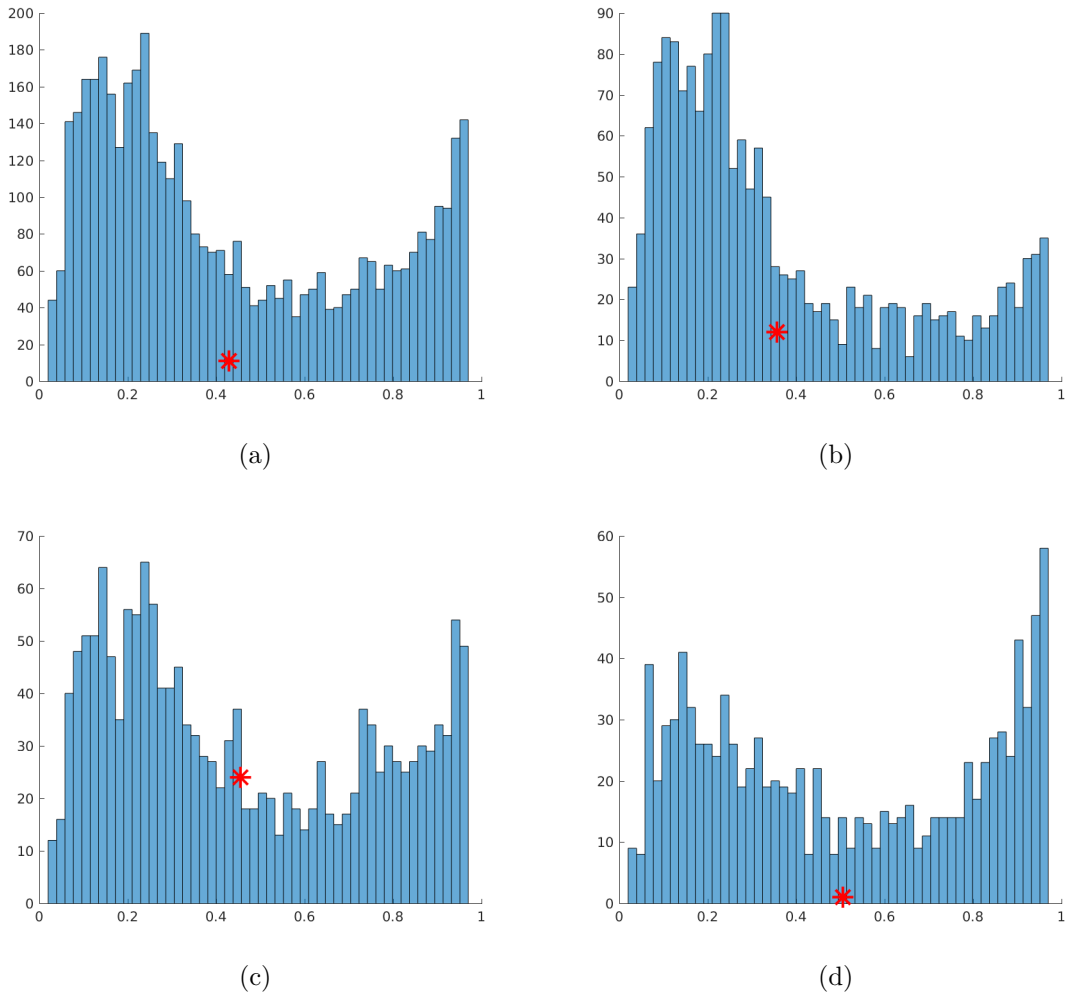


Figure 7: Histograms of AOR scores $\{s_{il}\}$ for four categories of trackers: (a) all trackers, (b) lowest category, (c) middle category, and (d) best category, in the combined dataset (i.e., OTB-100+VOT2018-ST+NfS-30).

3.4.2 Error Norm-Based Grouping

Grouping divides the T test trackers into a set of groups $\{g_i, i = 1, \cdots, G \leq T\}$ of similarly performing trackers based on the T scores $\{s_i\}$ of an objective measure over all test sequences $\{v_l\}$.

To assign a group, let the best score of the best performing unranked tracker in the combined scores $\{s_i\}$ be s_B, and let the error between s_B and a score s_i in $\{s_i\}$ of another unranked tracker be $\eta_i = s_B - s_i$. Let the scale estimate be

$$\sigma_\eta = c_\eta \cdot MAD(\{\eta_i\}), \tag{27}$$

where c_η is the scale factor of the family distribution of $\{\eta_i\}$. A tracker t_i belongs in the same group as the best performing unranked tracker if $\eta_i \leq \sigma_\eta$. If there still exist unranked trackers, we repeat the process: we update s_B using the scores of unranked trackers, compute errors $\{\eta_i\}$, measure the scale estimate σ_η, and compare η_i to σ_η for each tracker, until all trackers are assigned a group.

To determine c_η in (27), we determine the distribution of errors $\{\eta_{il}\}$ at each sequence v_l. For a sequence l and for all trackers i, let the best score in $\{s_{il}\}$ be s_{Bl}, and let $\eta_{il} = s_{Bl} - s_{il}$ be the error between s_{Bl} and the score of a tracker. Figure 8 displays the histograms of errors $\{\eta_{il}\}$ across all sequences. As can be seen, the tails are heavy, meaning they do not go down to zero. Overall, the histograms are not conclusive about the type of probability distribution errors $\{\eta_{il}\}$ represent. The role of c_η is important for grouping since it determines which trackers are grouped together based on how similar their combined scores $\{s_i\}$ are. The smaller c_η is, the more restrictive (27) is in grouping similar trackers together. In [66], values of c_η for different distributions are suggested. We test three values of c_η and found that $c_\eta = 0.9102$ gives the most restrictive grouping, meaning that it is easier for a tracker to become an outlier of a group. Therefore, we set $c_\eta = 0.9102$.

Algorithm 4 shows how grouping is performed. First, it calculates the best score

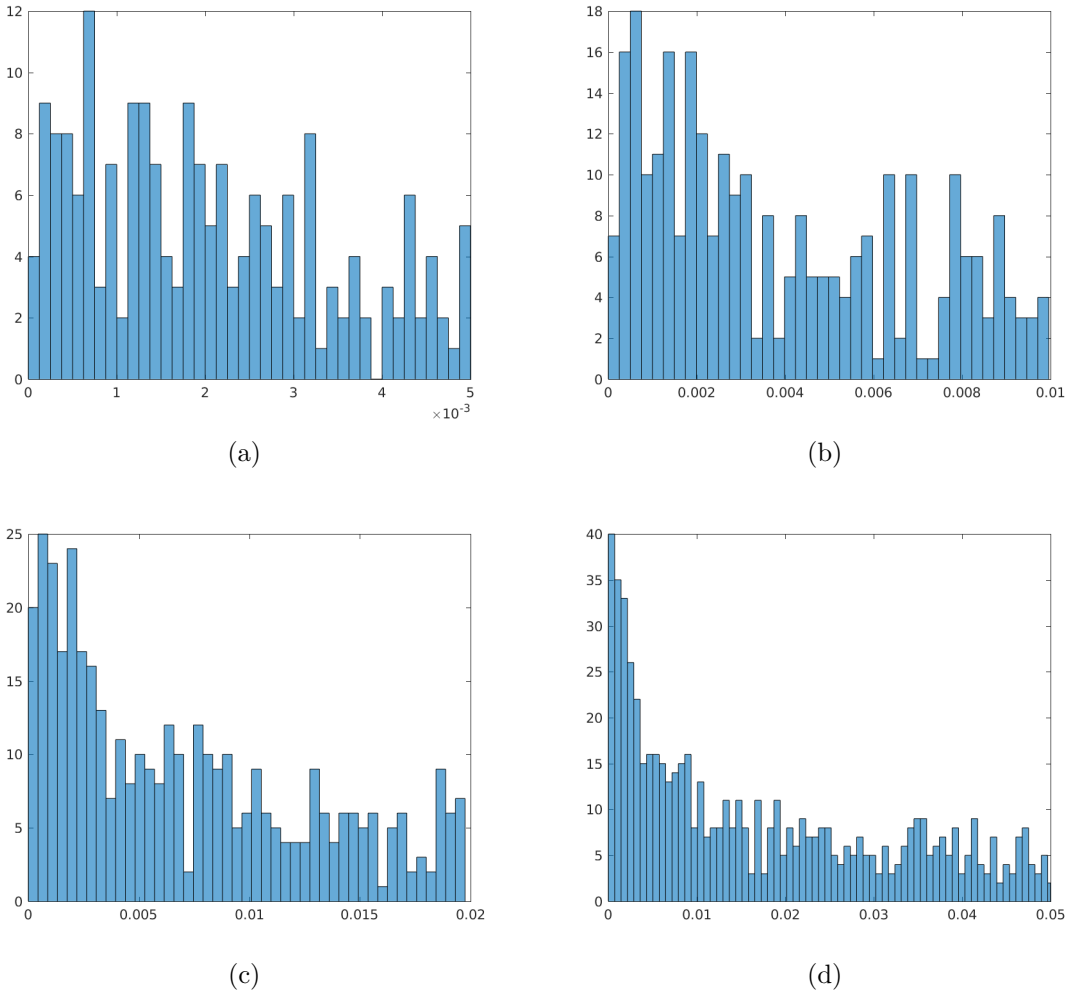


Figure 8: Histograms of score differences $\{\eta_{il}\}$ across ranges (a) 0 to 0.005, (b) 0 to 0.01, (c) 0 to 0.02, and (d) 0 to 0.05 in the combined dataset (i.e., OTB-100+VOT2018-ST+NfS-30).

s_B over the T scores s_i of the unassigned (not yet grouped) trackers in $\{t_i\}$. Then, it computes the T errors $\{\eta_i\}$ and the scale estimate σ_η. An error η_i which is within the range defined by σ_η has its corresponding tracker assigned a counter *count* as its group g_i; the counter starts at 1 and increments after each round. A new round of grouping over the unassigned trackers starts if there still exists an unassigned tracker in $\{t_i\}$. The grouping algorithm ends whenever each t_i is assigned a g_i.

Algorithm 4: Grouping of all trackers $\{t_i\}$.

Data: Scoring data $\{s_i\}$ of all trackers $\{t_i; i = 1, \cdots, T\}$.
Result: Group g_i for each t_i.

$c_\eta = 0.9102$;
$count = 1$;
while $\exists$ *unranked* t_i *in* $\{t_i\}$ **do**
  $s_B = \max(\{s_i\})$;
  for *each unassigned* t_i *in* $\{t_i\}$ **do**
    $\eta_i = s_B - s_i$;
  end
  $\sigma_\eta = c_\eta \cdot MAD(\{\eta_i\})$;
  for *each unassigned* t_i *in* $\{t_i\}$ **do**
    if $\eta_i \leq \sigma_\eta$ **then**
      $g_i = count$;
    end
  end
  $count++$;
end

Note that the proposed scoring in section 3.4.1 assigns similarly-performing trackers similar scores; this facilitates the task of Algorithm 4 to group these similarly-performing trackers into the same group.

3.4.3 Iterative Ranking

The T tested trackers are iteratively scored and ranked based on the quality data $\{q_{il}\}$ over all test sequences $\{v_l\}$. At the start of each iteration k, each tracker is placed in a set j of trackers with the same assigned rank $\{t_i\}_j$, $j = 1, \cdots, J \leq T$.

Initially, none of the t_i are ranked and they all belong in the same set j of unranked trackers (i.e., $r_i^0 = 0$ and $J = 1$). The first iteration is similar to grouping; therefore, $r_i^1 = g_i$. At iteration k, a rank is assigned at each tracker and multiple trackers may have the same rank. The ranking algorithm ends whenever all ranks in the final set $\{r_i^{1:k}\}$ are all distinct from each other or when $k = T$. For more simplicity, the final rank obtained at the last iteration is named *rank* and is noted r_i. As an example, iterations operate as follows. In the first iteration, the ranking algorithm ranks and partitions the T trackers into J^1 sets j^1 based on the trackers' scores $\{s_i\}$ computed from the quality values $\{q_{il}\}$; best trackers are in set 1, second best trackers are in set 2, third best trackers are in set 3, etc. In the second iteration, the J^1 sets j^1 are partitioned into $J^2 > J^1$ subsets j^2 based on the scores re-computed from the quality values $\{q_{il}\}_j$ of the t_i in each set j^1. Scores are re-computed in each set to better determine how trackers within the same set perform with respect to each other. In the following iterations, the same process is repeated: subsets are partitioned into smaller subsets, and all trackers in the same newly partitioned subset are scored and ranked against each other. In the last iteration, there can be up to $J = T$ sets, that is, when all trackers are ranked differently.

Algorithm 3 shows more precisely how the entire ranking process is achieved. First, we use the quality data $Q_j = \{q_{il}\}_j$, in other words, the quality values of trackers i in the set of trackers $\{t_i\}_j$ over all test sequences $\{v_l\}$. Q_j is a matrix of size $T_j \times L$, where $T_j = |\{t_i\}_j|$ is the number of trackers in $\{t_i\}_j$ at iteration k; a row in Q_j designates a tracker i from $\{t_i\}_j$ and a column designates a sequence l. Then, we *score()* each tracker in $\{t_i\}_j$ as per section 2.3.1 to produce the T_j scores of set j, namely $\{s_i\}_j$. We take the maximum score s_B in $\{s_i\}_j$ to calculate errors differences η_i between s_B and all the other scores in $\{s_i\}_j$. We compute the scale estimator σ_η using the MAD of errors $\{\eta_i\}$. An error η_i which is within the range defined by σ_η has its corresponding tracker assigned a counter *count* as its rank r_i^k; the counter starts

at 1 and increments after each round of ranking ("while" loop in Algorithm 5), that is, after at least one tracker in $\{t_i\}_j$ has been ranked. A new round starts if there still exists an unranked t_i in $\{t_i\}_j$. Once all trackers are ranked in iteration k, we take the new sets $\{t_i\}_j$ of similarly ranked trackers for iteration $k+1$. The sets j for iteration $k+1$ depend on the tracker ranks at iteration k, and j varies from 1 and J. The algorithm iterates if $k \leq T$ and if at least two trackers are of the same rank. The algorithm stops either when $k = T+1$ or when all trackers are ranked differently. In fact, T is the minimal amount of iterations required to fully achieve ranking, without exception, over all $\{t_i\}$. Hence, a final rank $1 \leq r_i \leq T$ and only trackers that have the same score end with the same rank.

Essentially, algorithm 3 in its first iteration $k = 1$ divides all trackers $\{t_i\}$ into groups (same as algorithm 2); then, in each of the following iterations, each group is divided into sub-groups, and each tracker in each sub-group is given a rank. Ranking is based on the scores which are recomputed at each iteration. This is repeated until all trackers are ranked.

At each new iteration, each trackers within a same set j have its score recomputed. The score at $k = 1$ is the most relevant to display for analysis since it is used for grouping and shows how all T trackers compare with respect to the top performing trackers. At the last iteration, the final score of a tracker i is simply the score of the tracker in the last set it was assigned in. This means the final score is not meaningful to show or interpret on its own. However, the final rank r_i depends on all scores, from the first iteration to the last iteration, hence the importance of computing scores at each iteration.

3.4.4 How Robust is our Method?

To demonstrate the robustness of our scoring and ranking algorithms, we run a total of M experiments where the original AOR quality data of each frame of a test video

Algorithm 5: Iterative ranking of all trackers $\{t_i\}$.

Data: Quality data $\{q_{il}\}$ of all trackers $\{t_i; i = 1, \cdots, T\}$ on all test sequences $\{v_l; l = 1, \cdots, L\}$.

Result: Rank r_i for each t_i.

$k = 1$;
$j = 1$;
$c_\eta = 0.9102$;
$\{t_i\}_j = \{t_i\}$;
do
  for *each* $\{t_i\}_j$ **do**
    $Q_j = \{q_{il}\}_j$;
    $\{s_i\}_j = score(Q_j)$;
    $count = 1$;
    while $\exists$ *unranked* t_i *in* $\{t_i\}_j$ **do**
      $s_B = \max(\{s_i\}_j)$;
      for *each unranked* t_i *in* $\{t_i\}_j$ **do**
        $\eta_i = s_B - s_i$;
      end
      $\sigma_\eta = c_\eta \cdot MAD(\{\eta_i\})$;
      for *each unranked* t_i *in* $\{t_i\}_j$ **do**
        if $\eta_i \leq \sigma_\eta$ **then**
          $r_i^k = count$;
        end
      end
      $count++$;
    end
  end
  $k++$;
  $\{t_i\}_j = update(\{t_i\}_j)$;
while $k \leq T$ *and* $\exists\ t_{i_1} \wedge t_{i_2}$ *such that* $r_{i_1} = r_{i_2}$;

signal and for all trackers are subjected to variations of signal independent noise; then, we evaluate our algorithm's response to impulse and non-zero mean Gaussian noise.

In the context of this experiment, impulse noise can be interpreted as the result of random sharp and sudden image changes such as occlusion or fast motion. We can also interpret the additivity of Gaussian noise to $\{q_{il}\}$ as random gradual changes through frames such as illumination variation or motion blur.

To show the robustness of our scoring method, given T tested trackers $\{t_i; i = 1, \cdots, T\}$, let their scores be $\{s_i\}$ and mean measures be $\{p_i\}$ generated from the original quality data $\{q_{il}\}$ of an objective measure (such as AOR) over all sequences $\{v_l\}$, and let their noisy scores be $\{s_i^n\}$ and noisy mean measures be $\{p_i^n\}$ generated from the noisy quality data $\{q_{il}^n\}$ of that same objective measure. Our scoring is more robust than averaging (mean) if, after M experiments,

$$\forall i, \quad \Psi(s_i, \mu_{s_i^n}) > \Psi(p_i, \mu_{p_i^n}), \tag{28}$$

where $\Psi(\cdot) = \frac{min(\cdot)}{max(\cdot)} \in [0, \cdots, 1]$ is a min-max ratio applied for normalization,

$$\mu_{s_i^n} = \frac{1}{K} \sum_{k=1}^{K} s_i^k, \tag{29}$$

$$\mu_{p_i^n} = \frac{1}{K} \sum_{k=1}^{K} p_i^k. \tag{30}$$

$\mu_{s_i^n}$ is the average score of tracker i over K different levels of a noise distribution under test, and $\mu_{p_i^n}$ is the average performance mean of tracker i over the same K noise levels. In other words, our scoring is more robust than the mean if, overall, each t_i yields a ratio of scores $\Psi(s_i, \mu_{s_i^n})$ closer to 1 than the ratio of means $\Psi(p_i, \mu_{p_i^n})$.

To test if our ranking method is robust, let the ranks of all the T trackers be $\{r_i\}$ generated from the original quality data $\{q_{il}\}$ of an objective measure over each sequence l, and let their noisy ranks be $\{r_i^n\}$ generated from the noisy quality data

$\{q_{il}^n\}$ of that same objective measure. Our ranking is robust if, after M experiments,

$$\forall i, \quad |r_i - \mu_{r_i^n}| \leq 1, \tag{31}$$

where $\mu_{r_i^n}$ is the average rank of each tracker i over K different levels of a noise distribution,

$$\mu_{r_i^n} = \frac{1}{K} \sum_{k=1}^{K} r_i^k. \tag{32}$$

In other words, our ranking is robust if, overall, each t_i yields at most a difference of 1 between its original r_i and its corresponding average of noisy ranks $\mu_{r_i^n}$.

Given the original overlap ratio quality data $\{q_{il}\}$, that is, AOR_l values at each test video sequence v_l for each tracker t_i, we add noise to AOR_l to get $\{q_{il}^n\}$. For impulse noise, the noisy quality data $\{q_{il}^n\}$ is the original quality data to which we apply a binary-state sequence, with a noise density varying from 0.05 to 0.5, specifically, 0.05, 0.2, 0.35, and 0.5. For Gaussian noise, we pick three distributions with different standard deviations *std*, namely 0.01, 0.05, and 0.1. To each Gaussian distribution, we vary the mean from 0.1 to 0.4, specifically, 0.1, 0.2, 0.3, and 0.4. We run the experiments $M = 50$ times across the combined dataset OTB-100+VOT2018-ST+NfS-30.

Tables 14 and 15 display $\Psi(p_i, \mu_{p_i^n})$ (noted 'Mean ratio'), $\Psi(s_i, \mu_{s_i^n})$ (noted 'Score ratio') and $|r_i - \mu_{r_i^n}|$ (noted 'Rank diff.') results under impulse noise and Gaussian noise, respectively. Both tables show that our proposed ranking responds moderately to even strong variations in the input data not just on average, but for every tracker i. Indeed, the rank difference is less or equal to 1, affirming the robustness of our ranking methodology. We also see that for each t_i, the score ratio does not goes below 0.85, which is comparatively much more robust than the mean ratio where the ratio can reach 0.55. In all cases of noise disturbances, the score performs much better than the mean.

In Table 16, we show the percentage of trackers with mean ratio and score ratio above a certain threshold th_s over all 16 cases of noise levels: 4 levels of impulse noise

Table 14: AOR mean ratio, score ratio, and rank difference results under impulse noise and averaged over $M = 50$ runs for each tracker t_i in the combined dataset OTB-100+VOT2018-ST+NfS-30.

(a) Impulse noise of density 0.05 and 0.2

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.981078654	0.999205714	0
CFWCR	0.996780421	0.99955136	0
CSRDCF	0.988732015	0.999351942	0
CREST	0.983288468	0.999153937	0
DASIAMRPN	0.991781021	0.999536976	0
DAT	0.917651148	0.999265949	0
DIMP	0.97464534	0.999855262	0
DLST	0.983835936	0.998633037	0
DSST	0.957685338	0.997978372	0
ECO	0.992843112	0.999950358	0
IBCCF	0.996848717	0.999041002	0
KCF	0.936910052	0.997481759	0
LADCF	0.996321597	0.999885663	1
MCCT	0.99156707	0.999265677	0
MDNET	0.994658004	0.999961849	1
SAMF	0.969248802	0.999209337	0
SIAMFC	0.975777956	0.999034748	0
SIAMRPN++	0.995545584	0.999224445	0
STAPLE	0.975109287	0.999384474	0
STRCF	0.999360849	0.999639455	0
Mean	0.979983469	0.999230566	0.1

(b) Impulse noise of density 0.35 and 0.5

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.934882845	0.998770995	0
CFWCR	0.990490375	0.99942841	0.5
CSRDCF	0.962662934	0.9944358	0
CREST	0.945035584	0.994056754	0
DASIAMRPN	0.97336923	0.995614467	0.5
DAT	0.766090635	0.996737743	0
DIMP	0.91390958	0.996882138	0
DLST	0.946237705	0.994822673	0.5
DSST	0.868971616	0.996553021	0
ECO	0.976099981	0.996368589	0
IBCCF	0.990599838	0.994742217	0
KCF	0.812712296	0.992592969	0
LADCF	0.987156695	0.995949764	0.5
MCCT	0.972919019	0.998923836	0
MDNET	0.982325837	0.998752756	1
SAMF	0.903102689	0.995155999	0
SIAMFC	0.921908232	0.994944481	0
SIAMRPN++	0.984727234	0.997764493	0
STAPLE	0.9194122	0.99583698	0
STRCF	0.997719301	0.995994194	0
Mean	0.937516691	0.996216414	0.15

(c) Impulse noise of density 0.05 to 0.5

Impulse noise	Mean ratio	Score ratio	Rank diff.
ATOM	0.95798075	0.999782956	0
CFWCR	0.993635398	0.999938365	0.25
CSRDCF	0.975523343	0.997542139	0
CREST	0.963782608	0.996605346	0
DASIAMRPN	0.982488874	0.998038853	0.25
DAT	0.835049599	0.998736167	0
DIMP	0.94427746	0.9983687	0
DLST	0.96467061	0.996727855	0.25
DSST	0.911174234	0.999289372	0
ECO	0.984471546	0.998159474	0
IBCCF	0.993714454	0.997851068	0
KCF	0.870403057	0.995037364	0
LADCF	0.991739146	0.997917714	0.75
MCCT	0.982154536	0.999829349	0
MDNET	0.988491921	0.999395454	1
SAMF	0.935007346	0.997182668	0
SIAMFC	0.948078492	0.997955333	0
SIAMRPN++	0.990106859	0.999270325	0
STAPLE	0.946442023	0.998226443	0
STRCF	0.998540075	0.998177434	0
Average	0.957886617	0.998201619	0.125

Table 15: AOR mean ratio, score ratio, and rank difference results under Gaussian noise and averaged over $M = 50$ runs for each tracker t_i in the combined dataset OTB-100+VOT2018-ST+NfS-30.

(a) Gaussian noise with $std = 0.01$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.748214507	0.963073661	0
CFWCR	0.709223798	0.939084357	0
CSRDCF	0.676218643	0.903943365	0.25
CREST	0.665404088	0.893662382	0.25
DASIAMRPN	0.676245202	0.891437569	1
DAT	0.551333114	0.918716621	0
DIMP	0.763261544	0.971502462	0
DLST	0.667779477	0.909896947	1
DSST	0.626906324	0.92097128	0
ECO	0.719521345	0.950349749	0
IBCCF	0.697200361	0.92612956	0
KCF	0.587551475	0.901167953	0
LADCF	0.715058153	0.949385065	0
MCCT	0.687020924	0.925361883	0
MDNET	0.712601568	0.937525427	0
SAMF	0.643980694	0.899606313	0
SIAMFC	0.656223841	0.912024883	0
SIAMRPN++	0.679283962	0.874485865	0
STAPLE	0.653835926	0.90913283	0
STRCF	0.708117988	0.947135646	0
Average	0.677249147	0.922229691	0.125

(b) Gaussian noise with $std = 0.05$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.750081733	0.963138397	0
CFWCR	0.71079911	0.938447283	0
CSRDCF	0.677649947	0.904401311	0.25
CREST	0.666787461	0.894723739	0.25
DASIAMRPN	0.677679033	0.891919471	1
DAT	0.552113129	0.919681491	0
DIMP	0.765255539	0.971502462	0
DLST	0.669141625	0.910028409	1
DSST	0.628073436	0.920750291	0
ECO	0.721142927	0.949846804	0
IBCCF	0.698786194	0.925936375	0
KCF	0.588503664	0.902401772	0
LADCF	0.716701809	0.949422806	0.25
MCCT	0.688532417	0.92503552	0
MDNET	0.714264459	0.937596045	0.25
SAMF	0.64521491	0.900153368	0
SIAMFC	0.657564815	0.912403061	0
SIAMRPN++	0.680738166	0.875100456	0
STAPLE	0.655113483	0.909273919	0
STRCF	0.709728753	0.947096478	0
Average	0.678693631	0.922442973	0.15

(c) Gaussian noise with $std = 0.1$ and means 0.1, 0.2, 0.3, and 0.4

Trackers	Mean ratio	Score ratio	Rank diff.
ATOM	0.755364431	0.963883484	0
CFWCR	0.715102504	0.938634568	0
CSRDCF	0.681508377	0.906739884	0.25
CREST	0.670359174	0.897880182	0.25
DASIAMRPN	0.681550049	0.894134869	0.75
DAT	0.553648704	0.922993475	0
DIMP	0.770917229	0.971711471	0
DLST	0.672796428	0.911389085	0.75
DSST	0.630710363	0.921800934	0
ECO	0.725544081	0.949272662	0
IBCCF	0.702987379	0.927251625	0
KCF	0.590429639	0.906054384	0
LADCF	0.721047971	0.94983816	0.25
MCCT	0.692516513	0.925770161	0
MDNET	0.718964236	0.938727383	0.25
SAMF	0.648278495	0.903199149	0
SIAMFC	0.660866444	0.914203701	0.25
SIAMRPN++	0.684797439	0.878031599	0
STAPLE	0.658331094	0.911112074	0.25
STRCF	0.713915243	0.947566715	0
Average	0.68248179	0.924009778	0.15

Table 16: Percentage of trackers with mean ratio or score ratio above th_s.

Threshold th_s	Mean ratio % trackers	Score ratio % trackers
0.85	38	100
0.9	37	89
0.95	28	45

as per Table 14 and 12 levels of Gaussian noise as per Table 15. We can observe that for $th_s = 0.85$, 100% of trackers are robustly scored using the score. Comparatively, the mean robustly scores 38% of all trackers with the same threshold. Therefore, the score is much more robust than the mean.

We applied our robustness experiment only to the AOR values AOR_l at each sequence but not to the FR values FR_l because the latter are too small. In fact, the average of all our AOR data is 0.46331, whereas that of our FR data is 0.2535, and larger levels of noise negatively affects the experiment if our measure is, overall, closer to 0 or to 1 (here the FR). To keep our noise distribution as unaltered as possible, it is best to operate on a measure that is, overall, closer to 0.5 (here the AOR).

3.5 Results and Discussion

3.5.1 Experimental Setup

Our experimental setup for this ranking methodology is the same as in Chapter 2. We collect the mean, score, group, and rank of $T = 20$ test trackers for AOR and FR measures over three short-term datasets (OTB-100 [91], VOT2018-ST [48], and NfS-30 [42]), a long-term dataset (VOT2018-LT [48]), and all three short-term datasets combined. Datasets are briefly described in Chapter 1 section 1.5.1 and trackers are presented in Chapter 2 section 2.4.1.

Table 17: AOR and FR mean, score, group, and rank for all trackers t_i over the OTB-100 dataset. The 5 top-ranked trackers in terms of AOR are: ECO, LADCF, DIMP, ATOM, STRCF; those in terms of FR are: DIMP, CFWCR, ECO, LADCF, MDNET.

AOR	Mean	Score	Group	Rank
ATOM	0.6604672	0.7937057	1	4
CFWCR	0.6547581	0.7813532	1	6
CSRDCF	0.5843171	0.6593592	2	14
CREST	0.5819746	0.6739853	2	12
DASIAMRPN	0.6003705	0.6709943	2	13
DAT	0.3344862	0.3242475	5	20
DIMP	0.6781111	0.8280003	1	3
DLST	0.5487419	0.6137969	3	17
DSST	0.523439	0.5966455	3	18
ECO	0.6783898	0.8274936	1	1
IBCCF	0.6367119	0.7681573	1	8
KCF	0.4814151	0.4947638	4	19
LADCF	0.6751837	0.8126713	1	2
MCCT	0.6386578	0.7574106	1	9
MDNET	0.6614607	0.7985909	1	7
SAMF	0.5632847	0.6401123	3	16
SIAMFC	0.5816725	0.6694178	2	11
SIAMRPN++	0.5736622	0.6146818	3	15
STAPLE	0.5901572	0.695049	2	10
STRCF	0.6679971	0.7987394	1	5

FR	Mean	Score	Group	Rank
ATOM	0.0617093	0.9305636	1	6
CFWCR	0.0670644	0.9450902	1	2
CSRDCF	0.1142575	0.8950142	2	12
CREST	0.1221244	0.8894272	2	13
DASIAMRPN	0.0734687	0.9249542	1	10
DAT	0.342127	0.5686148	4	20
DIMP	0.0395621	0.961019	1	1
DLST	0.1476421	0.8353379	2	16
DSST	0.2122907	0.7624949	3	18
ECO	0.0569944	0.9552191	1	3
IBCCF	0.0809925	0.9287144	1	9
KCF	0.1994879	0.7769466	3	19
LADCF	0.0703238	0.9462413	1	4
MCCT	0.0885177	0.9143517	1	11
MDNET	0.0617588	0.9452098	1	5
SAMF	0.1479902	0.8549461	2	17
SIAMFC	0.1450552	0.8438953	2	15
SIAMRPN++	0.0632414	0.9360453	1	8
STAPLE	0.1500159	0.8509796	2	14
STRCF	0.0749524	0.9305925	1	7

3.5.2 Experimentation on OTB-100 Dataset

Mean, score, group, and rank results over the 100 sequences of OTB-100 are displayed in Table 17. Out of all our benchmark experiments, OTB-100 presents the least total amount of groups with 5 groups for AOR and 4 groups for FR. In fact, group 1 consists of 9 trackers for AOR and 11 trackers for FR. Given that OTB-100 is to this day a well known and used dataset, many trackers are robust to this dataset's video challenges. Only a few trackers fail across the sequences, which is emphasized by the high FR scores and the small amount of FR groups.

3.5.3 Experimentation on VOT2018-ST Dataset

Table 18 displays mean, score, group, and rank information for all trackers over the 60 sequences of VOT2018-ST. Compared to OTB-100, we notice that the number of groups has increased: 10 groups for AOR and 7 groups for FR. In fact, the scores

Table 18: AOR and FR mean, score, group, and rank for all trackers t_i over the VOT2018-ST dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, MDNET, SIAMRPN++, IBCCF; those in terms of FR are: DIMP, SIAMRPN++, ATOM, MDNET, CFWCR.

AOR	Mean	Score	Group	Rank
ATOM	0.482828	0.6708311	2	2
CFWCR	0.3870565	0.5182375	4	9
CSRDCF	0.3749558	0.513232	4	10
CREST	0.3558382	0.4912048	4	11
DASIAMRPN	0.402751	0.5604237	3	6
DAT	0.2711128	0.3555812	8	18
DIMP	0.5482059	0.7985742	1	1
DLST	0.3556825	0.4865725	4	13
DSST	0.2600767	0.3517127	9	19
ECO	0.3847133	0.5479286	3	5
IBCCF	0.37951	0.5282464	4	7
KCF	0.2552798	0.3419272	10	20
LADCF	0.3631683	0.4992387	4	8
MCCT	0.374658	0.512846	4	12
MDNET	0.4766664	0.7000073	2	3
SAMF	0.3150993	0.4314304	6	16
SIAMFC	0.2866871	0.3912163	7	17
SIAMRPN++	0.4342422	0.6072277	3	4
STAPLE	0.3294024	0.4449213	5	15
STRCF	0.3526868	0.4728831	5	14

FR	Mean	Score	Group	Rank
ATOM	0.22094	0.771977	2	4
CFWCR	0.248847	0.7519466	2	5
CSRDCF	0.2891818	0.7049066	3	7
CREST	0.3406829	0.6610585	4	11
DASIAMRPN	0.2356522	0.752081	2	6
DAT	0.4168348	0.5392024	6	18
DIMP	0.1535168	0.8531173	1	1
DLST	0.3403929	0.6436126	4	10
DSST	0.4760989	0.5012136	7	20
ECO	0.2935217	0.70949	3	8
IBCCF	0.3342149	0.6768261	3	9
KCF	0.4671505	0.5233971	6	19
LADCF	0.3559108	0.6438706	4	13
MCCT	0.355839	0.645337	4	12
MDNET	0.214113	0.7965435	1	3
SAMF	0.4068028	0.6031668	5	16
SIAMFC	0.4240857	0.5803619	5	17
SIAMRPN++	0.1748644	0.8185867	1	2
STAPLE	0.3683486	0.6449755	4	15
STRCF	0.3735231	0.6306167	4	14

show more variability in the dispersion of quality data. AOR group 1 only consists of DIMP whereas FR group 1 includes DIMP, MDNET, and SIAMRPN++.

3.5.4 Experimentation on NfS-30 Dataset

We display in Table 19 the mean, score, group, and rank results over the 100 sequences of NfS-30, which is a lesser known dataset than OTB-100 or VOT2018-ST. There are 7 groups for both AOR and FR performance measures. In both cases, group 1 consists of DIMP and ATOM trackers.

3.5.5 Experimentation on Cross Datasets (Short-Term)

For OTB-100, we observed in Table 17 that practically half of the trackers are in group 1. Therefore, to reduce possible bias towards certain benchmarks, test the generalization ability of the trackers, and make the ranking more fair, we run the

Table 19: AOR and FR mean, score, group, and rank for all trackers t_i over the NfS-30 dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, ECO, CFWCR, LADCF; those in terms of FR are: DIMP, ATOM, CFWCR, ECO, SIAMRPN++.

AOR	Mean	Score	Group	Rank
ATOM	0.5856113	0.7846434	1	2
CFWCR	0.4452981	0.5967391	2	4
CSRDCF	0.3822804	0.498637	3	11
CREST	0.3477486	0.4599894	4	12
DASIAMRPN	0.3792108	0.4805913	4	14
DAT	0.2596969	0.3354613	6	19
DIMP	0.6230729	0.843832	1	1
DLST	0.3855128	0.5189785	3	9
DSST	0.280508	0.3842639	5	16
ECO	0.4664525	0.6216311	2	3
IBCCF	0.4072187	0.5388863	3	7
KCF	0.2088253	0.2770332	7	20
LADCF	0.4472722	0.586832	2	5
MCCT	0.356908	0.4802145	4	13
MDNET	0.4105017	0.5554385	3	8
SAMF	0.2863981	0.3692833	5	17
SIAMFC	0.3293154	0.4413291	4	15
SIAMRPN++	0.4207635	0.5314879	3	10
STAPLE	0.2917094	0.3782973	5	18
STRCF	0.4284926	0.5630206	2	6

FR	Mean	Score	Group	Rank
ATOM	0.1455286	0.8807551	1	2
CFWCR	0.2545074	0.766677	2	3
CSRDCF	0.3407397	0.6991046	3	12
CREST	0.3973243	0.6373365	4	13
DASIAMRPN	0.3182575	0.694433	3	11
DAT	0.4493075	0.5883171	5	17
DIMP	0.1407434	0.8922171	1	1
DLST	0.3415471	0.6951342	3	9
DSST	0.5211405	0.5205605	6	19
ECO	0.2645178	0.7600167	2	4
IBCCF	0.3448246	0.6718715	3	10
KCF	0.5659809	0.4555992	7	20
LADCF	0.2865056	0.7391654	2	6
MCCT	0.3975578	0.6464746	4	14
MDNET	0.3288137	0.7056717	3	8
SAMF	0.4834366	0.5716105	5	18
SIAMFC	0.4267401	0.6107371	4	15
SIAMRPN++	0.2245123	0.7948451	2	5
STAPLE	0.4358835	0.6134499	4	16
STRCF	0.2987297	0.7196054	3	7

scoring and ranking across all 260 video sequences of OTB-100, VOT2018-ST, and NfS-30 combined. Table 10 displays AOR and FR means, scores, ranks, and groups across the combined short-term dataset. There is a total of 7 groups for AOR and 7 groups for FR. ATOM and DIMP trackers are the only trackers which share group 1 for AOR and FR performance measures simultaneously.

3.5.6 Experimentation on Long-Term Dataset VOT2018-LT

We repeat the scoring and ranking for $T = 18$ trackers on the 35 long-term videos of VOT2018-LT. AOR and FR means, scores, ranks, and groups are displayed in Table 21. We notice that DIMP is the only tracker to be assigned group 1 on for AOR but that both DIMP and SIAMRPN++ are in group 1 for FR measures. We also observe that ATOM and DASIAMRPN are in group 2 for AOR and FR measures simultaneously. The ranking results well underline the tracking architecture of

Table 20: AOR and FR mean, score, group, and rank for all trackers t_i over the combined short-term dataset OTB-100+VOT2018-ST+NfS-30. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, ECO, MDNET, LADCF; those in terms of FR are: DIMP, ATOM, SIAMRPN++, CFWCR, ECO.

AOR	Mean	Score	Group	Rank
ATOM	0.5906724	0.7618645	1	2
CFWCR	0.5124136	0.6496314	2	6
CSRDCF	0.4582889	0.5638181	4	11
CREST	0.4396945	0.5494962	4	13
DASIAMRPN	0.469692	0.5722411	4	14
DAT	0.2910925	0.3357906	7	20
DIMP	0.6269563	0.8272992	1	1
DLST	0.4414042	0.5479701	4	12
DSST	0.3692259	0.4584384	5	18
ECO	0.5290973	0.6838	2	3
IBCCF	0.4890813	0.6246067	3	8
KCF	0.3243848	0.3757512	6	19
LADCF	0.5155204	0.6534801	2	5
MCCT	0.4693663	0.5943587	3	10
MDNET	0.5222897	0.6823205	2	4
SAMF	0.3995135	0.4877903	5	17
SIAMFC	0.4165323	0.5174898	4	15
SIAMRPN++	0.4826679	0.5809627	3	9
STAPLE	0.4151902	0.5154983	4	16
STRCF	0.5031144	0.6328815	2	7

FR	Mean	Score	Group	Rank
ATOM	0.130693	0.8748096	1	2
CFWCR	0.1811077	0.8318981	2	4
CSRDCF	0.2417332	0.7757934	3	11
CREST	0.2784071	0.7397688	3	12
DASIAMRPN	0.2050452	0.7963984	2	9
DAT	0.4005905	0.5694052	7	20
DIMP	0.1047752	0.9096563	1	1
DLST	0.2667019	0.7371691	3	14
DSST	0.3919579	0.6091475	5	18
ECO	0.1913943	0.8234345	2	5
IBCCF	0.2409023	0.7718006	3	10
KCF	0.402215	0.59484	6	19
LADCF	0.2193753	0.7968189	2	7
MCCT	0.2690688	0.7492418	3	13
MDNET	0.1996309	0.8187721	2	6
SAMF	0.336734	0.6878679	4	17
SIAMFC	0.3177872	0.6934037	4	16
SIAMRPN++	0.1510278	0.8546317	1	3
STAPLE	0.3103494	0.7120826	4	15
STRCF	0.2299215	0.7802184	2	8

ATOM, DASIAMRPN, DIMP, and SIAMRPN++ trackers. In fact, these trackers have an implemented target re-detection mechanism which allows them to undergo long-term challenges, such as disappearance, reappearance, and full occlusion, more advantageously than the other test trackers.

3.5.7 Analysis and Discussion

3.5.7.1 Which are the Best Performing Trackers?

We summarize in Table 12 which trackers achieve rank 1 (i.e., $r_i = 1$) at the end of our iterative algorithm and which ones achieve group 1 (i.e., $r_i^1 = 1$) after the first iteration. We see that ECO and DIMP achieve rank 1 in either accuracy or robustness, and that ATOM, CFWCR, ECO, DIMP, IBCCF, LADCF, MCCT, MDNET, and STRCF achieve group 1 for both AOR and FR, simultaneously, across at least one dataset. When all three short-term datasets are merged, ATOM and DIMP are placed

Table 21: AOR and FR mean, score, group, and rank for all trackers t_i over the VOT2018-LT dataset. The 5 top-ranked trackers in terms of AOR are: DIMP, ATOM, DASIAMRPN, SIAMRPN++, LADCF; those in terms of FR are: DIMP, SIAMRPN++, ATOM, DASIAMRPN, ECO.

AOR	Mean	Score	Group	Rank
ATOM	0.4303145	0.6844961	2	2
CFWCR	0.3246751	0.4742774	4	9
CSRDCF	0.2506819	0.3489375	6	13
CREST	0.3029719	0.4556385	4	11
DASIAMRPN	0.4468216	0.6999189	2	3
DAT	0.2079797	0.283074	7	17
DIMP	0.5117432	0.8804891	1	1
DSST	0.2289441	0.3301396	6	15
ECO	0.3344038	0.5186743	3	7
IBCCF	0.3126537	0.4746609	4	10
KCF	0.1504492	0.2293131	8	18
LADCF	0.3547751	0.5562116	3	5
MCCT	0.287423	0.3958591	5	12
MDNET	0.3362735	0.5160192	3	6
SAMF	0.2382136	0.336087	6	14
SIAMRPN++	0.4219827	0.6526328	2	4
STAPLE	0.2264454	0.3088976	7	16
STRCF	0.3238329	0.5118763	3	8

FR	Mean	Score	Group	Rank
ATOM	0.3573803	0.6981281	2	3
CFWCR	0.3685781	0.6689347	2	6
CSRDCF	0.5113552	0.5277801	4	12
CREST	0.3982958	0.6615586	2	10
DASIAMRPN	0.340461	0.7020253	2	4
DAT	0.5370542	0.4826416	5	15
DIMP	0.2638707	0.8541872	1	1
DSST	0.6127074	0.4433618	5	16
ECO	0.3854537	0.6964721	2	5
IBCCF	0.4987637	0.5763853	3	11
KCF	0.6938831	0.3346302	7	18
LADCF	0.4280497	0.6307304	2	7
MCCT	0.5172444	0.5222231	4	14
MDNET	0.4126278	0.635648	2	9
SAMF	0.539489	0.4996207	4	13
SIAMRPN++	0.3213667	0.7824473	1	2
STAPLE	0.6244623	0.4074101	6	17
STRCF	0.4388514	0.6256763	2	8

Table 22: Summarizing results: trackers grouped 1 and trackers ranked 1 (underlined and in bold) in all benchmark experiments.

Dataset	AOR	FR
OTB	**ECO**, LADCF, DIMP, ATOM, STRCF, CFWCR, MDNET, IBCCF, MCCT	**DIMP**, CFWCR, ECO, LADCF, MDNET, ATOM, STRCF, SIAMRPN++, IBCCF, DASIAMRPN, MCCT
VOT-ST	**DIMP**	**DIMP**, SIAMRPN++, MDNET
NfS	**DIMP**, ATOM	**DIMP**, ATOM
VOT-LT	**DIMP**	**DIMP**, SIAMRPN++
OTB+VOT-ST+NfS	**DIMP**, ATOM	**DIMP**, ATOM, SIAMRPN++

in group 1 for AOR and FR, and DIMP is ranked 1 in both cases.

ATOM and DIMP ranks are also preserved when looking into the four individual experiments (OTB-100, VOT2018-ST, NfS-30, and VOT2018-LT). By counting how often a tracker achieves group 1 in AOR and FR measures across all four benchmarks, we see that DIMP scores 8/8, followed by ATOM with 4/8, MDNET and SIAMRPN++ with 3/8, and finally CFWCR, ECO, IBCCF, LADCF, MCCT, STRCF with 2/8. Overall, the top trackers in our benchmark are DIMP and ATOM.

Table 23: Top five ranked trackers t_i based on the AOR and FR means and scores in all benchmark experiments.

Rank	OTB-100		VOT2018-ST		NfS-30		OTB-100+VOT2018-ST+NfS-30		VOT2018-LT	
	Mean	Score	Mean	Score	Mean	Score	Mean	Score	Mean	Score
1	ECO	ECO	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP
2	DIMP	LADCF	ATOM	ATOM	ATOM	ATOM	ATOM	ATOM	DASIAMRPN	ATOM
3	LADCF	DIMP	MDNET	MDNET	ECO	ECO	ECO	ECO	ATOM	DASIAMRPN
4	STRCF	ATOM	SIAMRPN++	SIAMRPN++	LADCF	CFWCR	MDNET	MDNET	SIAMRPN++	SIAMRPN++
5	ATOM	STRCF	DASIAMRPN	IBCCF	CFWCR	LADCF	LADCF	LADCF	LADCF	LADCF

Rank	OTB-100		VOT2018-ST		NfS-30		OTB-100+VOT2018-ST+NfS-30		VOT2018-LT	
	Mean	Score	Mean	Score	Mean	Score	Mean	Score	Mean	Score
1	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP	DIMP
2	ECO	CFWCR	SIAMRPN++	SIAMRPN++	ATOM	ATOM	ATOM	ATOM	SIAMRPN++	SIAMRPN++
3	ATOM	ECO	MDNET	ATOM	SIAMRPN++	CFWCR	SIAMRPN++	SIAMRPN++	DASIAMRPN	ATOM
4	MDNET	LADCF	ATOM	MDNET	CFWCR	ECO	CFWCR	CFWCR	ATOM	DASIAMRPN
5	SIAMRPN++	MDNET	DASIAMRPN	CFWCR	ECO	SIAMRPN++	ECO	ECO	CFWCR	ECO

3.5.7.2 Mean-Based Ranking vs. Score-Based Ranking

We show in Table 23 the trackers that have been assigned the five best ranks in each experimental benchmark based on the mean and on the score. DIMP and ATOM can be considered the two best trackers whether we look into the mean-based ranking or the score-based ranking. Most notable differences in terms of ranking go to CFWCR and DASIAMRPN which are ranked higher with the score-based ranking and the mean-based ranking, respectively. In both rankings combined, the most prevalent trackers are DIMP, ATOM, SIAMRPN++, ECO, LADCF, and MDNET.

We list in Table 24 the AOR ranks and FR ranks of all trackers based on the mean and on the score in two experimental benchmarks: the combined dataset OTB-100+VOT2018-ST+NfS-30 and the long-term dataset VOT2018-LT. The five best ranks are noted in bold. Overall, there is at most a difference of 1 between the mean-based ranks and the score-based ranks. However, for CREST, DASIAMRPN, DLST, and STRCF trackers, there is at least one benchmark or performance measure where the difference between the mean-based rank and the score-based rank is of 2 or more. Such differences emphasize the variability in the quality data of those four trackers that our error norm-based scoring accounts for.

Table 24: Ranks r_i of each tracker t_i based on the AOR mean, similarity-based score, and error norm-based score in the combined dataset OTB-100+VOT2018-ST+NfS-30 and the long-term dataset VOT2018-LT.

AOR	OTB+VOT-ST+NfS		VOT2018-LT	
	Mean rank	Score rank	Mean rank	Score rank
ATOM	**2**	**2**	**3**	**2**
CFWCR	6	6	8	9
CSRDCF	12	11	13	13
CREST	14	13	11	11
DASIAMRPN	10	14	**2**	**3**
DAT	20	20	17	17
DIMP	**1**	**1**	**1**	**1**
DLST	13	12	-	-
DSST	18	18	15	15
ECO	**3**	**3**	7	7
IBCCF	8	8	10	10
KCF	19	19	18	18
LADCF	**5**	**5**	**5**	**5**
MCCT	11	10	12	12
MDNET	**4**	**4**	6	6
SAMF	17	17	14	14
SIAMFC	15	15	-	-
SIAMRPN++	9	9	**4**	**4**
STAPLE	16	16	16	16
STRCF	7	7	9	8

FR	OTB+VOT-ST+NfS		VOT2018-LT	
	Mean rank	Score rank	Mean rank	Score rank
ATOM	**2**	**2**	**4**	**3**
CFWCR	**4**	**4**	**5**	6
CSRDCF	11	11	12	12
CREST	14	12	7	10
DASIAMRPN	7	9	**3**	**4**
DAT	19	20	14	15
DIMP	**1**	**1**	**1**	**1**
DLST	12	14	-	-
DSST	18	18	16	16
ECO	**5**	**5**	6	**5**
IBCCF	10	10	11	11
KCF	20	19	18	18
LADCF	8	7	9	7
MCCT	13	13	13	14
MDNET	6	6	8	9
SAMF	17	17	15	13
SIAMFC	16	16	-	-
SIAMRPN++	**3**	**3**	**2**	**2**
STAPLE	15	15	17	17
STRCF	9	8	10	8

3.5.7.3 How do the Score and Mean Compare?

Combined scores $\{s_i\}$ and mean measures $\{p_i\}$ generated from the original quality data $\{q_{il}\}$ of an objective measure are heavily correlated. In fact, the correlation factor between $\{s_i\}$ and $\{p_i\}$ for both AOR and FR performance measures are 0.9924 and -0.9948, respectively. However, there are important differences between the score and the mean that justify our usage of the score over the mean to characterize tracker performance.

A widely used method to summarize the overall (center) performance of a tracker is by averaging the averages of its performance data $\{q_{il}\}$ (e.g., AOR) in all video sequences of a dataset. The average (mean) is, however, not the best measure of center of a variable unless the distribution is symmetric [84]. To show this in our context, we group trackers into three categories of performance (best, middle, lowest) and plot the histograms of their average AOR values $\{q_{il}\}$ across the combined dataset OTB-100+VOT-ST+NfS-30 in Figure 9. We also indicate the AOR mean on each histogram. As can be seen, these distributions are not symmetric; therefore, the mean is not a good measure of central tendency. As we have seen in Figure 7, the scores generated from our scoring method do follow a symmetric distribution.

Using the score over the mean is more robust (as shown in section 3.4.4) as well as more representative of the distribution of the quality data $\{q_{il}\}$. Unlike the score, the mean does not compare tracker performances at each sequence. In fact, the score evaluates at a sequence-level which is the best performing tracker and then scores all other trackers relatively to the top tracker. Therefore, our ranking is based on the combined scores $\{s_i\}$ instead of the mean.

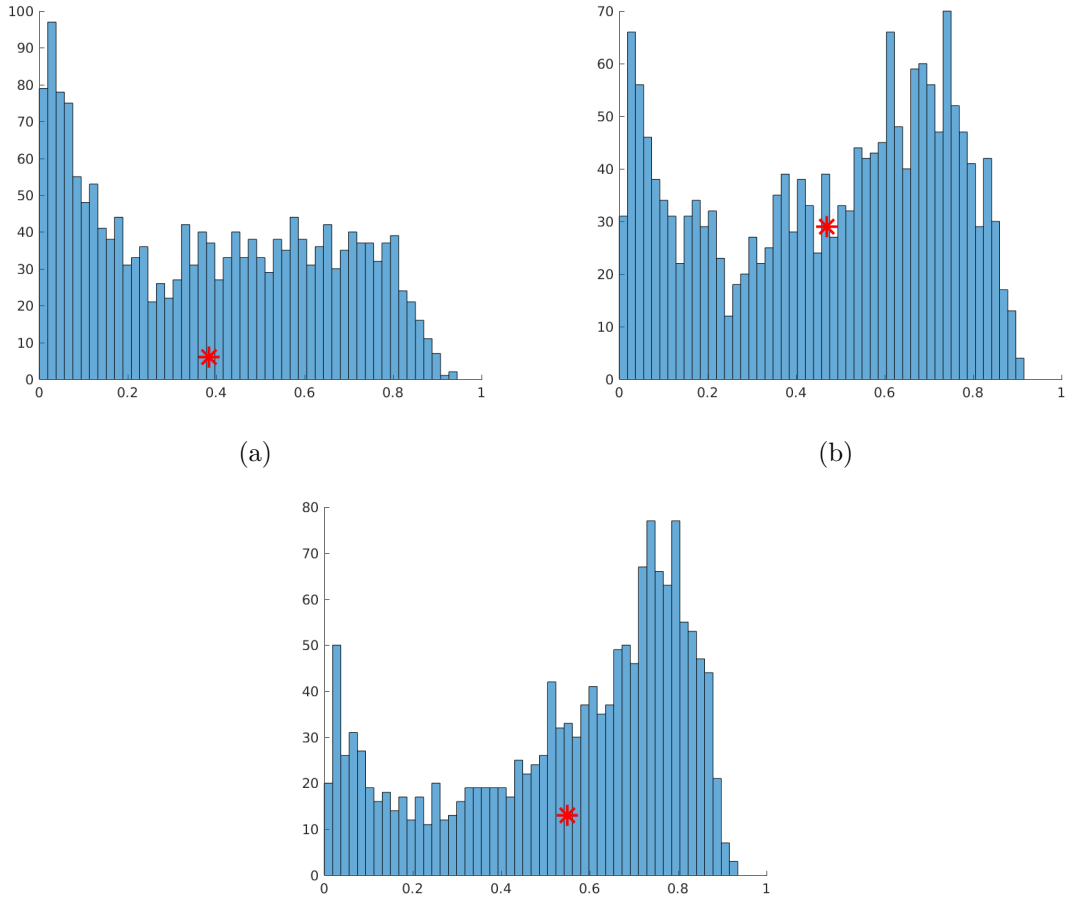


Figure 9: Histograms of AOR quality data $\{q_{il}\}$ for three categories of trackers: (a) lowest category, (b) middle category, and (c) best category, in the combined dataset (i.e., OTB-100+VOT2018-ST+NfS-30).

3.6 Conclusion

We developed a tracker ranking protocol inspired by the use of error norms in anisotropic diffusion for image denoising. For scoring, we use the Lorentzian edge-stopping function to robustly estimate the similarities in the AOR and FR quality data of all trackers at each sequence. We then ran an iterative algorithm to cluster all trackers into groups and sub-groups of similar performance. To extensively test our method, we ranked our tested trackers over three short-term datasets, individually and altogether, and one long-term dataset, and compared our ranks with those from the similarity-based ranking. Our observations over all ranking outputs indicated that DIMP was the best performing tracker, followed by ATOM, and then by a group of similarly performing trackers MDNET and SIAMRPN++. Our framework also displayed, overall, ranking results similar to those from our similarity-based scoring and ranking methodology.

Our scoring calculates the scores based on the mean averaged of a performance measure over all frames of a video sequence. Future work may include computing scores at the frame level, instead of at the sequence level, and then calculating the average score or applying another scoring function to these scores.

Chapter 4

Drift Detection and Correction using Region Proposal Networks

4.1 Introduction

Object tracking is a complex computer vision task which estimates the state of a target object at every frame of a video sequence. Despite all the progress made in this field, frequent challenging factors such as heavy occlusion, object deformation, strong motion, illumination variation, and background clutter still lead trackers to show inaccuracies, drift, and potentially fail [17, 18]. During drift, object labels or high level structures are more complicated to identify. Therefore, it begs to question if detecting the presence of an object rather than the object or its label could suffice to detect drift. For example, measuring a region's objectness, or likelihood to be an object, could inform that an object is drifting away.

A tracker may undergo partial drift or complete drift (failure), as shown in Figure 10; in the former case, the tracker's output still overlaps with the ground truth object, but not in the latter case. Detecting such occurrences has become essential in recent object tracking techniques as it allows a tracker to reinitialize its tracking

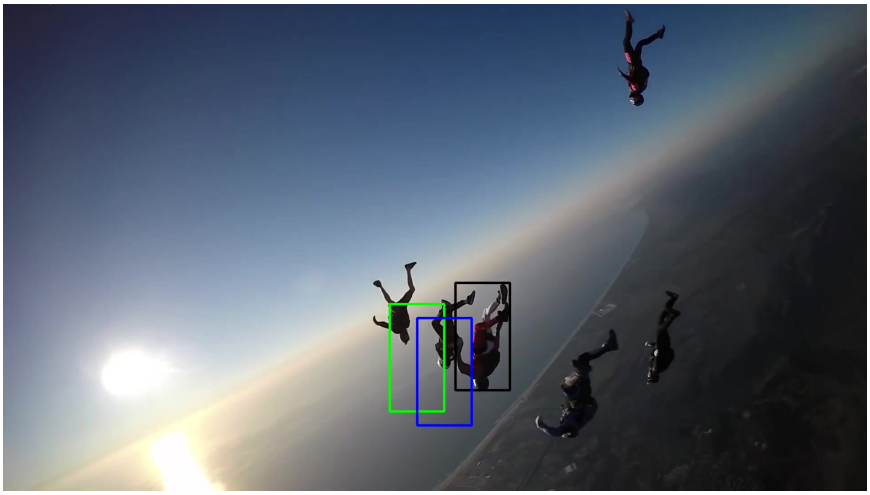


Figure 10: Examples of partial drift (blue) and complete drift (green) from the ground truth (black) in 'Skydiving' sequence.

algorithm using a drift correction mechanism, hence bettering its overall accuracy and robustness performances. Few drift detection approaches assess tracking failure at a tracker's output, that is, independently of the base tracker's design. In fact, most drift detection and correction algorithms are called while the target model is updating [24, 97, 23, 100, 50, 81]. The aim of this chapter is to formulate the drift detection and correction problem into a dual process which can be applied over any tracker, independently of its architecture.

In the rest of the chapter, section 4.2 presents related work; section 4.3 describes our saliency and objectness-based drift detection and region proposal network (RPN)-based drift correction methodologies; section 4.4 provides simulation results; finally, section 4.5 concludes this chapter.

4.2 Related Work

In the context of visual tracking, it is important to make the distinction between object detection and drift detection as separate but related tasks. Object detection is a term that is often used to refer to the localization and classification of an object in

an input image or video. It is also used to refer to localization (without classification) of objects. Its output is a bounding box. Drift detection refers to methods that detect if a tracker is about to drift or has drifted away while following a target object. Drift detection is always followed with a drift correction (or tracker initialization) step, where the output bounding box of a tracker is updated. Among popular object detectors, region proposal networks [32, 31, 74, 59, 36, 15] are deep architectures which predict image regions to be either foreground objects or part of the background. Saliency and objectness are measures which can be used for both object and drift detection, as these measures estimate the presence of an object in the form of a percentage score [21, 16, 27, 20, 80, 1, 10, 101].

Compared to the use of object detection for object tracking [24, 97, 23, 52, 100, 50, 81, 73, 6, 99, 88, 51], there is little research done in drift detection and correction alone [29, 69, 89]. This section first investigates drift detection and correction and object detection in video tracking strategies; then it presents different ways in which the presence of objects is detected, whether it be with region proposal networks or with saliency and objectness measures.

4.2.1 Drift Detection in Object Tracking

The research most related to our work is [29]; it presents a saliency-based drift detection and a post-tracking segmentation-based drift correction. It requires no prior information of the target object and computes at each frame the overall saliency of the object inside the tracker output to dictate whether drift is happening or not. If drift is assumed to happen, it then applies automatic GrabCut segmentation [9] combined with a robust selection of foreground and background seeds to filter out feature points belonging to the background, extract the target object, and hence correct the tracker drift. This method is tracker-independent so it can be embedded to any tracking architecture. Among other segmentation-based drift correction methods, [69] is a

model-free and closed loop fine Random-Walker segmentation scheme which uses the spatial properties of a segmented object to enhance the object's localization in the following frame and reduce drift; [89] presents a joint online tracking and segmentation algorithm where the tracking output at each frame is initially hypothesized through SLIC superpixel segmentation. Multi-part tracking and segmentation tasks are then formulated in a unified energy objective function to facilitate tracking of the target object.

4.2.2 Object Detection in Object Tracking

In recent years, CNN-based trackers have become the highlight of state-of-the-art tracking techniques. The methods [24, 97, 23, 52, 100, 50, 81, 73] combine object tracking and object detection tasks but the drift detection problem is often overlooked or formulated differently as either a bounding box regression task or a global re-detection task. [97, 24, 23] are examples of using bounding box regression to avoid tracking failure. [97] handles changes in the target aspect ratio by modeling uncertainties and ambiguities in the target state and combining confidence-based regression models with its own probabilistic regression model; [24] predicts accurate target state estimations by overlap maximization; [23] prevents inappropriate model updates by introducing a Meta-Updater which captures appearance variations from the target object and its background at each frame. It stops the tracker from updating whenever the update is deemed likely to cause drift.

In long-term video challenges, re-detection is a tracking task which has increasingly gained attention [48, 63]. It consists of detecting a target object that was out-of-scene or fully occluded and that reappears later in the same sequence. For example, DASIAMRPN [100] and SIAMRPN++ [51] are Siamese tracking architectures [6, 99, 52, 88] which have gained significant popularity for their robustness to long-term challenges

and their balance between tracking accuracy and speed. Siamese architectures consist of a convolutional neural network applied on two streams, each processing the input image and a training image separately and then cross-correlated to search for the test image in the following frames. DASIAMRPN [100] and SIAMRPN++ [51] propose a framework consisting of a Siamese subnetwork for feature extraction and a region proposal network (RPN) for generating object proposals. The RPN is divided into two branches, one for foreground and background classification, and one for refining object proposals. To furthermore improve tracking performances in contexts of occlusion, [100] triggers a local-to-global search strategy that switches depending on a detection score and increases incrementally until the target object is recovered, whereas [51] implements an effective spatial aware sampling strategy on its search image.

4.2.3 Region Proposal Networks

Fully convolutional region proposal network (RPN) are deep networks that take an input image and outputs a set of scored rectangular object proposals (or region proposals), each score designating its proposal's objectness. RPN are primarily class-agnostic but recent object detection [32, 31, 74, 95, 98, 85, 59, 36, 15] and video object tracking [52, 100, 51] frameworks employ those architectures for object localization and classification tasks. [32] popularized deep RPN-based object detection and consists of three modules; the first module generates class-independent region proposals; the second module extracts from each proposal a 4096-dimensional feature vector; the third module is a support vector machine used for object classification. Numerous extensions to this framework have been proposed [31, 74, 59, 36, 15]. For example, [31] builds on [32] and improves on accuracy and on training and testing speeds to achieve near real-time rates, and [74] merges two RPN architectures, one of

which uses the convolutional feature maps generated by [31] to generate new proposals, to construct a single unified network for object detection. FPN [59] exploits the multi-scale pyramidal architecture of the convolutional neural network and builds a feature pyramid using image pyramids to get high-level semantic features maps at all scales.

Anchors are default bounding boxes which serve as regression references in RPN architectures to predict proposals. Guided Anchoring (GA-RPN) [85] revisits anchoring schemes and learns sparse anchors with a wide range of scales and aspect ratios rather than uniformly sampling them, and improves the detection performance by generating higher quality proposals. [95] introduces meta-learning to anchor generation; instead of enumerating every possible bounding box, that is, predefining a number of anchor boxes with all kinds of positions and sizes, anchor functions are dynamically generated. [98] formulates the anchor matching as a maximum likelihood estimation and assigns a likelihood probability to each anchor for representing an object. The likelihood probability is formulated as a loss function; therefore, anchors which have a large classification error are classified as part of the background.

4.2.4 Saliency and Objectness Measures

Saliency and objectness information can be used to detect the presence or absence of objects in an image region (often rectangular). In the context of drift detection, a bounding box which has low saliency or low objectness is one that is more likely to not contain an object. This section looks into existing measures of saliency and objectness.

4.2.4.1 Saliency

Saliency is the quality of an object to stand out from its environment. As such, it can be extracted to discriminate between an object and its background. For example, [20]

takes both a bottom-up and a data-driven human visual attention approach: it generates region-based contrast (RC) saliency maps, as an improvement over histogram-based contrast (HC) maps, where saliency values are assigned to regions of a segmented input image. The algorithm outperforms on natural scenes; however, it does not perform as well on images with high texture. Similarly to [29], which is used in drift detection, [16] selectively picks seeds to discriminate foreground from background; however, it localizes salient objects using bit-mapping in compressed images to generate a salient object window and salient object map within the bounderies of the window. [27] also performs saliency detection in the compression domain; it extracts luminance, color, texture, and motion information of video frames at block level and combines a static saliency map of viewed frames with a motion saliency map of predicted frames to generate its final saliency map. These non-learning-based approaches have significantly reduced computational complexity compared to their deep-learning counterparts, at the cost of being less accurate.

The challenges that come with estimating an object's saliency are in the subjective nature of saliency. For example, [12, 11, 60, 56] are created through a combined effort of human participation and automated saliency computation to distinguish as objectively as possible the salient regions of images, but human intervention adds a restriction on the size of the training examples and on the number of object classes. In addition to that, only a few benchmarks look into saliency of dynamic targets in videos [96]. Saliency datasets mostly contain natural scenes or obviously separated foregrounds and, as such, lack examples of dynamic objects. For these reasons, we do not investigate deep-learning saliency models.

4.2.4.2 Objectness

Objectness measures can also be used to evaluate the likelihood of an image region to be an object. Evaluating the objectness of an image window can be performed

in several ways, for example by defining a well-closed boundary, by differentiating a foreground from its background, or by capturing salient information. [21] presents a simple and fast method by resizing image windows to 8×8 and using binarized normed gradient features to describe objects. The authors also incorporate a segmentation strategy to improve the object localization. [1] trains its objectness model to separate class-specific object windows from non-class specific object windows and background windows using characteristics of superpixel straddling in image segmentation. [101] generates bounding box proposals by operating on groups of edges rather than superpixels: candidate boxes are scored based on the number of contours that are fully enclosed within the box. [80] proposes its own objectness measure, called foreground connectivity, to determine whether a pixel belongs to a foreground, and uses that information to generate a saliency map of an image. [10] also uses saliency characteristics but combines them to contrast and motion information to highlight moving objects in complex video settings.

Among the presented measures of objectness, only [21, 10] have been experimentally tested on datasets containing video challenges. Nonetheless, all methods are computationally inexpensive and can, therefore, be applied on a frame-basis in the scope of detecting the occurrence of drift.

4.2.5 Summary of our Contributions

Regardless of a tracker's robustness, drift is always susceptible to occur, hence the importance of drift detection. Different to CNN-based trackers [24, 97, 23, 52, 100, 50, 81, 73] that either apply bounding box regression to reduce drift or re-detect a target object after the tracker has failed, our method automatically detects instances where drift is occurring. In addition, our drift detection is not class-specific and, therefore, applies to any kind of target drift. Similar to [29], our proposed system is tracker-independent. We also estimate the likelihood an object is present inside the

tracker output by computing the saliency of the output. However, we improve on the accuracy of our drift detection by combining our saliency measurements with objectness measurements. Finally, unlike segmentation-based drift correction methods [29, 69, 89], our drift correction runs a RPN that reinitializes the tracker's output whenever drift occurs.

We contribute an efficient method that automatically detects drifts using a combination of edge-based objectness and superpixel saliency measures on the bounding box image. Then, if drift is assumed, we initiate a region proposal network strategy to relocate the bounding box on the drifting target. Both drift detection and correction algorithms are independent of the tracker baseline design and are, therefore, modules that can be added to any tracking architecture.

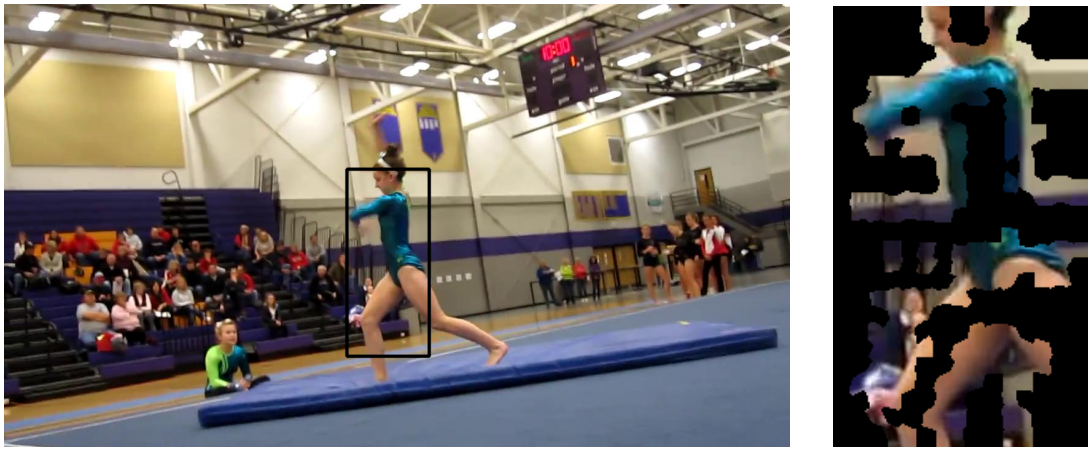
4.3 Proposed Method

Given the output bounding box B_t of a baseline object tracking algorithm, at the current frame F_t of a video sequence, our method applies two steps to improve the tracking accuracy of the baseline tracker: (1) drift detection using a combination of saliency and objectness measures, and (2) drift correction using a region proposal network.

4.3.1 Saliency and Objectness-Based Drift Detection

We propose a drift detection method which uses image color and edge information to detect the presence of objects in a bounding box. Measurements of saliency s are performed as per [29], where the image region in B_t is first partitioned into K segments r_k of saliency s_k, defined as follows

$$s_k = \sum_{j \neq k} n_j \cdot D_{lab}(r_k, r_j) \cdot e^{-E(r_k, r_i)/\sigma_s^2}, \tag{33}$$



(a) (b)

Figure 11: Partitioning of salient regions in sequence 'gymnastics2': (a) ground truth bounding box, and (b) salient segments of the bounding box image.

where n_j is the number of pixels in r_j, $D_{lab}(r_k, r_j)$ is the LAB color distance between r_k and r_j, and $E(r_k, r_j)$ is the Euclidean distance between those regions' center. All pixels within a segment r_k are defined as salient if s_k is larger than a given threshold. For example, the image mask from Figure 11b shows which pixels of Figure 11a are salient with respect to their neighbors. We can see that the segments which separate foreground from background are labeled as salient. While segments from the background are filtered out, we also notice in the foreground object that neighboring segments similar in color are not described as salient from each other. This is due to the the nature of (33) which relies on color and Euclidean distances between regions; therefore, contrasting regions are defined as more salient than neighboring regions that are homogeneous in color. Finally, the overall saliency s is calculated in [29] as the ratio between the number of salient pixels N_s and the total number of pixels N_t in the bounding box B_t,

$$s = \frac{N_s}{N_t}. \tag{34}$$

For objectness evaluation, we use the EdgeBoxes algorithm [101], which outputs a limited number of scored rectangular bounding box proposals {EB}; each score

(a) No drift (b) Drift

Figure 12: Objectness detection in sequence 'Deer' with Staple tracker [5]: EdgeBox proposals (green), Staple B_t (black), candidate EB which overlaps most with B_t (blue), and area A_{EB} (red).

represents the likelihood the score's corresponding bounding box contains an object based on image contours, in the form of a percentage score. Therefore, we assume that for a tracker output B_t to have high objectness o, it must have a high affinity with an EdgeBox proposal EB, that is, it must overlap well with an object proposal EB. At frame F_t, we measure o by applying [101] over a rectangular search region A_{EB}. The search region A_{EB} is centered around B_t and is of width $w^{B_t} + 2p_1$ and height $h^{B_t} + 2p_1$, where w^{B_t} and h^{B_t} are the width and height of B_t, respectively, and p_1 is the search region's pixel extension. Therefore, we measure objectness as follows

$$o = \max_{\{\text{EB}\}} \{\text{AOR}(B_t, \text{EB})\}, \tag{35}$$

where AOR is the average overlap ratio between B_t and a candidate proposal EB. Figure 12 shows how our objectness-based drift detection works. When B_t is well centered around the target object, there exists an object proposal EB in A_{EB} which overlaps well with B_t; however, when B_t is drifting away from its target, no proposal EB overlaps well with B_t.

Finally, to determine whether the target object inside B_t has drifted from its expected position, we compare parameters s in (34) and o in (35) to thresholds

(t_{s_1},t_{s_2}) and t_{o_1}, respectively,

$$\text{Drift} = \begin{cases} 1 & : (s \leq t_{s_1} \vee s \geq t_{s_2}) \wedge (o \leq t_{o_1}), \\ 0 & : \text{otherwise.} \end{cases} \tag{36}$$

Given the high frame rate of our non-learning-based approach, we can apply this saliency and objectness-based drift detection on a tracker output B_t at every frame F_t of a video sequence. Different from [29], our drift detection uses a combination of color and contour information to measure a window's likelihood of containing an object. Using only saliency or only objectness as a means to indicate the presence of drift yields more false positives than using a combination of the two. For a more efficient application of our RPN-based drift correction (see section 4.3.2), our drift detection looks into scenarios where B_t is on the edge of completely drifting (failure). In fact, our aim in using a region proposal for drift correction is not to refine the bounding box around the target object, but to relocate it when the target object is partially out of bounds. The thresholds (t_{s_1},t_{s_2}) and t_{o_1} are experimentally tested out with that aim in mind and are estimated using the OTB [91] benchmark (see section 4.4.1).

4.3.2 Drift Correction using Region Proposals

The drift correction module is triggered once drift is detected. For an image I of width W and height H, and for a tracker output B_t of size $w^{B_t} \times h^{B_t}$, let $\{\text{RP}\}$ be the proposals of a RPN (i.e., GA-RPN [85]) in a rectangular search region A_{RP}, where each candidate RP is of width w^{RP} and height h^{RP}. The search region A_{RPN} is centered around B_t and is of width $w^{B_t} + 2p_2$ and height $h^{B_t} + 2p_2$, where p_2 is the RPN search region's pixel extension. Although B_t is not a region proposal, we consider it to also be amidst the candidates. In fact, if the RPN does not provide good proposals to recover the target object from drift, B_t will remain the output of

the whole system.

For a proposal RP, we define the aspect ratio α_r^{RP} and the relative area α_a^{RP},

$$\alpha_r^{\text{RP}} = \frac{\min(w^{\text{RP}}, h^{\text{RP}})}{\max(w^{\text{RP}}, h^{\text{RP}})} \tag{37}$$

$$\alpha_a^{\text{RP}} = \frac{w^{\text{RP}} \times h^{\text{RP}}}{W \times H}, \tag{38}$$

where $(\alpha_r^{\text{RP}}, \alpha_a^{\text{RP}}) \in [0, \cdots, 1] \times [0, \cdots, 1]$. The aspect ratio α_r^{RP} is normalized as a ratio between the minimum and the maximum of a proposal's height and width in order to better compare the proportions of its corresponding proposal with that of other proposals. Similarly, the relative area α_a^{RP} is normalized by dividing the proposal's area by that of the entire frame I. Given the aspect ratio $\alpha_r^{B_t}$ and the relative area $\alpha_a^{B_t}$ of B_t, our goal is to select proposals $\{\text{RP}\}$ which fit both the size and the area of B_t. Therefore, we select region proposals such that $|\alpha_r^{\text{RP}} - \alpha_r^{B_t}| \leq c_r$ and $|\alpha_a^{\text{RP}} - \alpha_a^{B_t}| \leq c_a$, where c_r and c_a are thresholds. Theoretically, if a RP satisfies conditions $c_r \to 0$ and $c_a \to 0$, then both B_t and RP fit perfectly. However, we experimentally select values c_r and c_a that accept candidate bounding boxes of proportions slightly different to B_t. In fact, we assume that from frame $t-1$ to t and during drift, the proportions of B_t are susceptible to slightly change from those of B_{t-1}.

Let r_{k^*} be the partitioned superpixel regions of either B_{t-1} or B_1, r_{k_t} be the partitioned superpixel regions of either B_t or the selected candidate RP that overlaps most with B_t, and let r_{s^*} and r_{s_t} be their salient regions, respectively, with saliency calculated as per (33). To chose between the RPN candidate and B_t as the output at frame t, we measure the average color distances of their salient regions with B_{t-1} and B_1 and select the bounding box which solves the problem

$$\min_{r_{s_t}, r_{s^*}} \frac{1}{m} \sum_{j \in r_{s_t}} D_{lab}(r_{s^*}, r_j), \tag{39}$$

where m is the number of salient regions r_{s_t} and $D_{lab}(r_{s^*}, r_j)$ is the LAB color Bhattacharya distance between r_{s^*} and r_j. Hence, we choose the bounding box that has

salient features most similar color-wise to the salient features of either the tracker's previous output or the ground truth at frame $t = 1$.

Different from [29] which does not require any training, our drift correction uses a trained CNN-based region proposal network. Whilst the GrabCut segmentation-based drift correction works well in instances of a gradually drifting tracker [29], ours has a wider search area which allows to recover a target object from severe drift. In addition, the RPN is trained to detect objects with all their high level structures and generate a bounding box proposal around the entire object, unlike [29] which solely identifies the salient regions inside the tracker output. As such, both method are capable to recover from drift but ours performs more accurate reinitialization.

4.4 Results and Discussion

In this section, we provide the experimental setup and discuss the results of our drift detection and correction. We first present the setup for our work, then we discuss the evaluation protocol, and finally we discuss the results of our simulation and compare them to other methods.

4.4.1 Experimental Setup

We run our simulations on a virtual environment provided by ComputeCanada's Graham general purpose cluster. Trackers and drift detection and correction methods are run with P100 Pascal GPUs and using Python 3.7, CUDA 10, CUDNN 7.5, and OpenCV 3.4 modules. Trackers of choice for these simulations are DASIAMRPN [100] and SIAMRPN++ [51]. We test three variants of each tracker design: (1) the baseline tracker, (2) the baseline tracker to which we include our objectness-based drift detection and RPN-based drift correction, (3) the baseline tracker to which we include the saliency-based drift detection and GrabCut-based drift correction [29].

For our RPN-based drift correction, we use the state-of-the-art Guided Anchoring RPN (GA-RPN) [85] run with the ResNet-50 convolutional neural network [37] and FPN [59] as the backbone network. This architecture is provided by the mmdetection toolbox [19], an open source object detection toolbox which supports popular region proposal networks and object detection methods. For the segmentation-based drift correction [29], we use GrabCut segmentation from the OpenCV library.

We run every tracker variant over two datasets: VOT2018 [48] and TC128 [58]. VOT2018 is a widely used short-term tracking set of 60 video sequences released to the public for preparation to the yearly Visual Object Tracking Challenge. All frames are annotated and assigned an attribute: occlusion, illumination change, motion change, size change, camera motion, and unassigned. TC128 is a less experimentally used dataset consisting of 128 color sequences and 129 ground truths. While each ground truth is annotated at each frame, challenging attributes are provided on a per-sequence basis: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. Our evaluation protocol focuses primarily on analyzing variations in the failure rate FR, as it is a better estimator of drift than the average overlap area AOR. In fact, FR is better than AOR at representing the percentage of video frames in which a tracker either fails or tracks successfully.

We pick the widely used OTB dataset [91] for training in order to fine-tune our parameters. For drift detection, we set thresholds (t_{s_1}, t_{s_2}) and t_{o_1} in (36) to (0.375, 0.625) and 0.4, respectively. For EdgeBox [101], we set the maximim number of candidate boxes to 20, and the pixel extension of the image search area A_{EB} to $p_1 = 64$ pixels. Different values for A_{EB} were tested experimentally, and we chose a search area that is large enough for the EdgeBox object proposal to detect edge-based objects, but small enough for it to compute fast. For the Guided Anchoring RPN [85], we set both thresholds c_r and c_a to 0.125, and the pixel extension of the RPN search

area A_{RP} to $p_2 = 64$ pixels. We implement the Grabcut segmentation-based drift detection algorithm [29] with slight variations: (1) we measure histogram similarity using the Bhattacharya distance, (2) we call OpenCV's Fast keypoint detector instead of the SIFT keypoint detector, as SIFT is now patented as a non-free OpenCV 3 module, and (3) we define both an absolute background that is 24 pixels distant from the probable foreground, and a probable background that is in-between the probable foreground and the absolute background. For SLIC superpixel segmentation, we set the smoothing Gaussian kernel to 10 and the maximum number of superpixel segments to $100 + \frac{w^{B_t} \times h^{B_t}}{100}$ with a maximum cap of 400.

4.4.2 Comparison with Baseline Design

Table 25 displays the average FR and average AOR over VOT2018 and TC128 datasets of DASIAMRPN [100] and SIAMRPN++ [51] baseline designs as well as with our drift detection and correction implementation, denoted RPN DD/DC. We can see that our design improves the overall FR of each baseline tracker on the complete datasets. However, our drift detection and correction does not run on every sequence; which shows by the FR value being the same for both the base tracker and our implementation. For example, FR changes for DASIAMRPN in 37 of the 60 VOT2018 video sequences, and in only 17 of the 128 video sequences of TC128. For SIAMRPN++, FR changes in only 8 of the 60 VOT2018 video sequences, and in 15 of the 128 video sequences of TC128. Several factors may explain why FR does not always vary: (1) the baseline tracker was already successfully tracking its target object, (2) drift of failure was not detected, or (3) the drift detection algorithm was executed but the tracker's output was chosen over the RPN candidate boxes. In the first case, the failure rate remains unchanged whether drift is detected or not. In the second case, drift happened and remained undetected, or it was detected but was not successfully corrected. The last case depends more on the examples on which the

FR		Base Tracker	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	0.23565	0.20786	11.79%
	TC128	0.16484	0.15361	6.81%
SiamRPN++	VOT2018	0.17486	0.16123	7.79%
	TC128	0.1549	0.14951	3.47%

AOR		Base Tracker	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	0.40275	0.42436	5.37%
	TC128	0.5076	0.51299	1.06%
SiamRPN++	VOT2018	0.43424	0.44117	1.59%
	TC128	0.49145	0.49159	0.03%

Table 25: VOT2018 and TC128 datasets: Mean FR, AOR, and percentage improvement of the RPN-based drift detection and correction over the original base design of DASIAMRPN and SIAMRPN++.

region proposal network was trained, and therefore, on its ability to detect objects in challenging video images. To investigate how well our drift detection and correction worked on affected sequences, we measure FR across those sequences only, as shown in Table 26, and the overall improvement is much higher than when looking at all sequences altogether. Figures 13 to 22 display visual bounding box results of our implementation versus those of DASIAMRPN and SIAMRPN++ trackers on ten of the selected sequences from VOT2018 and TC128. Each frame is annotated with its frame number on the bottom left.

We also notice in Tables 25 and 26 that the AOR does not increase as much as FR, whether we are looking at the entire datasets VOT2018 and TC128 or at the selected videos in which FR change. In addition, that increase in FR does not compensate for a decrease in AOR. This means that our system is not badly affecting the AOR of the unselected videos, and therefore, our implemented system is, overall, more robust and accurate than the original base designs.

FR		# sequences	Base Tracker	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	37	0.3293	0.2843	13.68%
	TC128	17	0.44905	0.36387	18.97%
SiamRPN++	VOT2018	8	0.3293	0.2269	31.06%
	TC128	15	0.3679	0.3216	12.60%

AOR		# sequences	Base Tracker	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	37	0.3115	0.3471	11.43%
	TC128	17	0.3075	0.3537	15.03%
SiamRPN++	VOT2018	8	0.3119	0.3639	16.69%
	TC128	15	0.3787	0.3988	5.32%

Table 26: Selected videos from VOT2018 and TC128: Mean FR, AOR, and percentage improvement of the RPN-based drift detection and correction over the original base design of DASIAMRPN and SIAMRPN++.

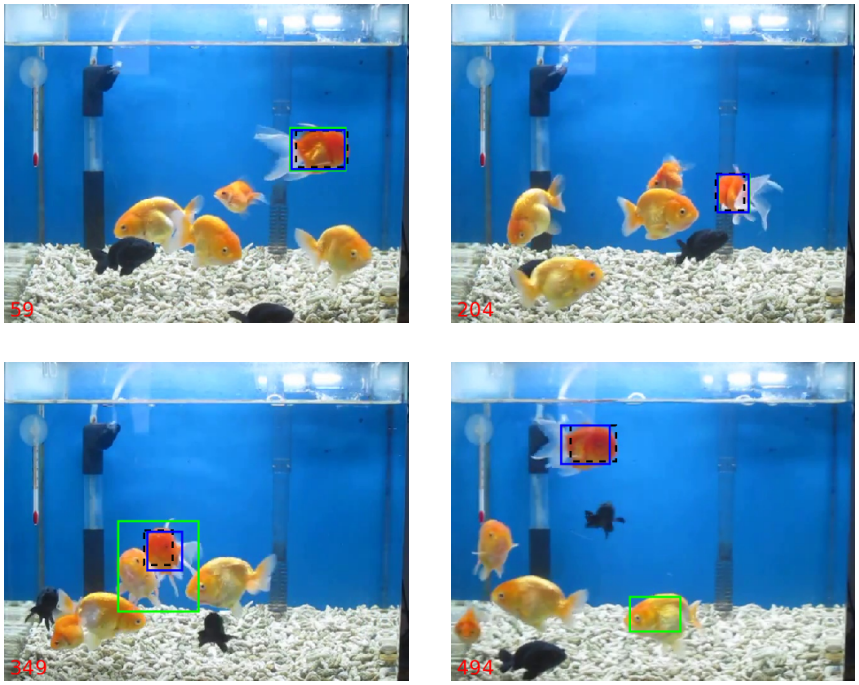


Figure 13: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'Fish2' sequence of TC128.

Figure 14: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'Badminton1' sequence of TC128.

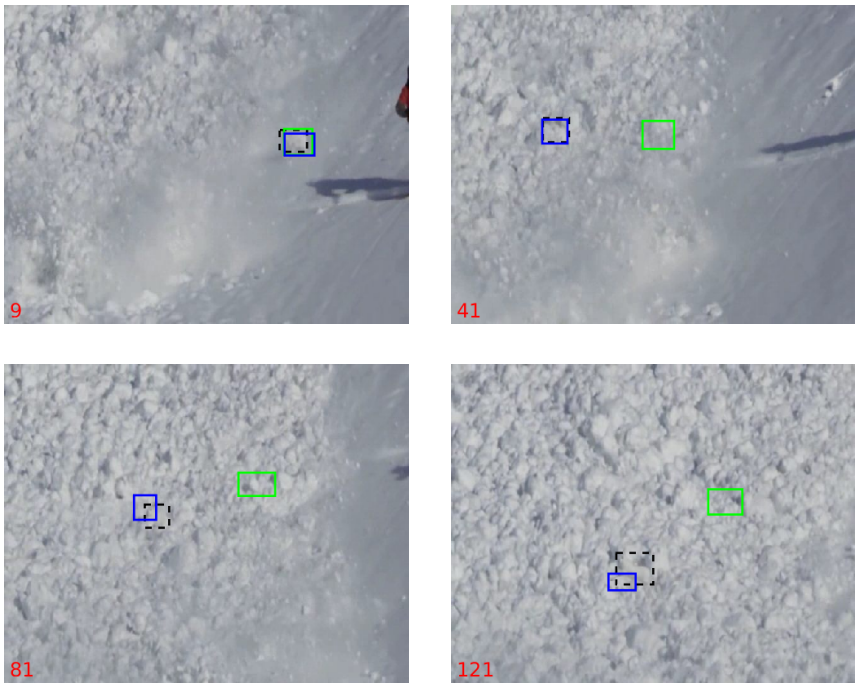


Figure 15: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'rabbit' sequence of VOT2018.

Figure 16: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'soccer2' sequence of VOT2018.

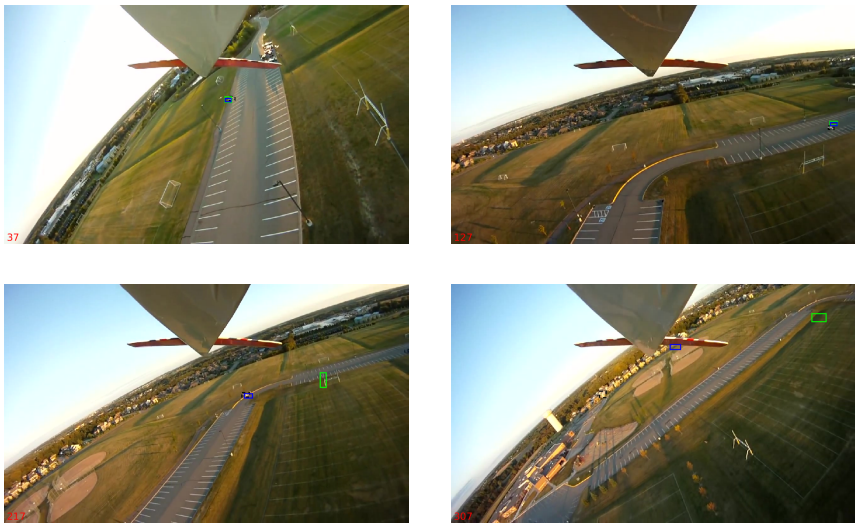


Figure 17: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'drone1' sequence of VOT2018.

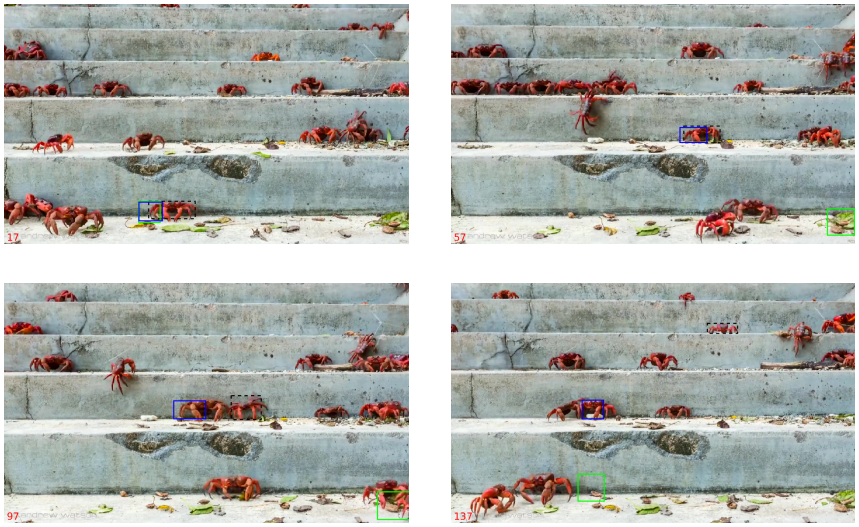


Figure 18: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'crabs1' sequence of VOT2018.

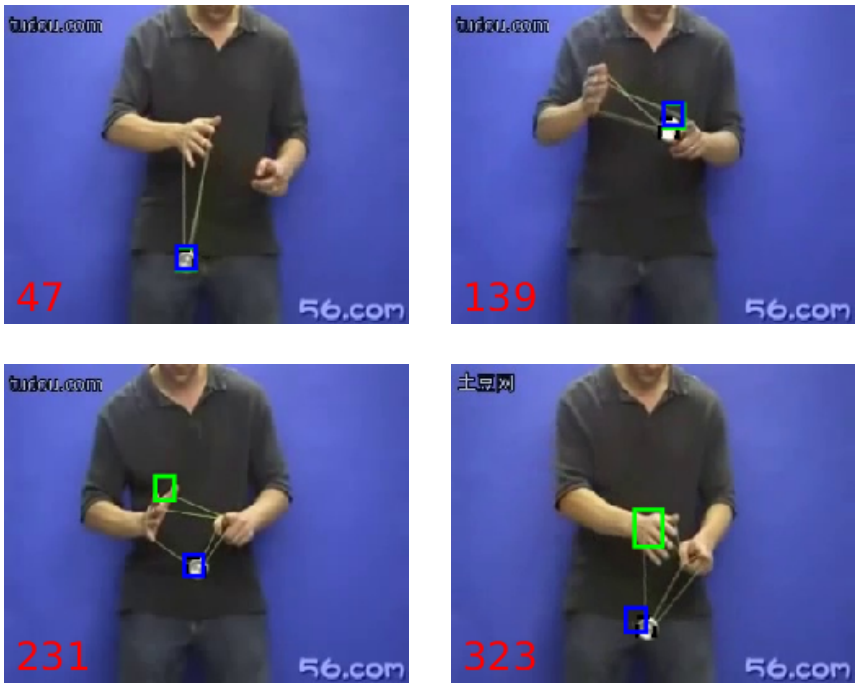


Figure 19: SIAMRPN++ (green), our implementation (blue), and ground truth (black) on 'Yo-yos2' sequence of TC128.

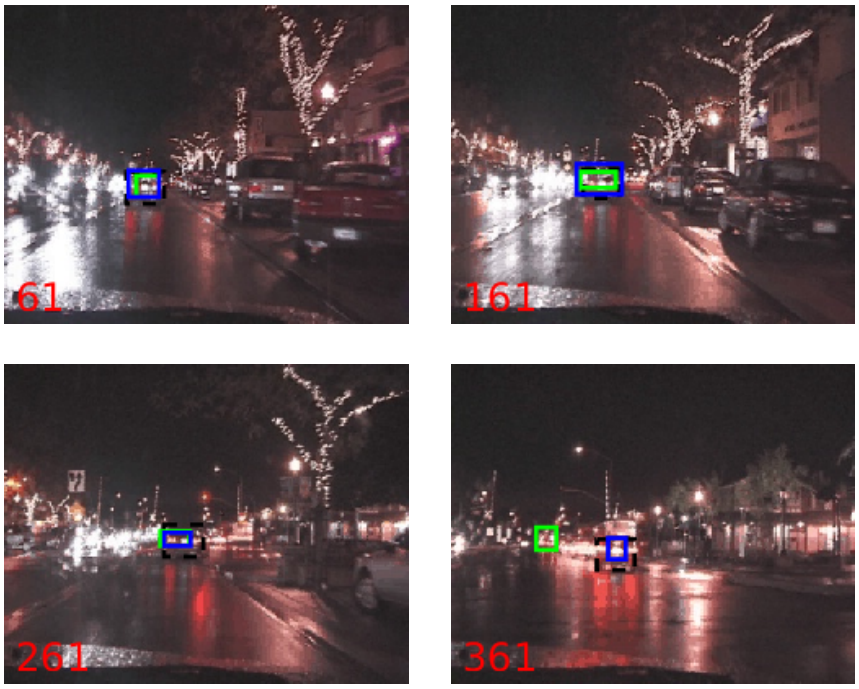


Figure 20: SIAMRPN++ (green), our implementation (blue), and ground truth (black) on 'CarDark' sequence of TC128.

Figure 21: SIAMRPN++ (green), our implementation (blue), and ground truth (black) on 'Hand2' sequence of TC128.

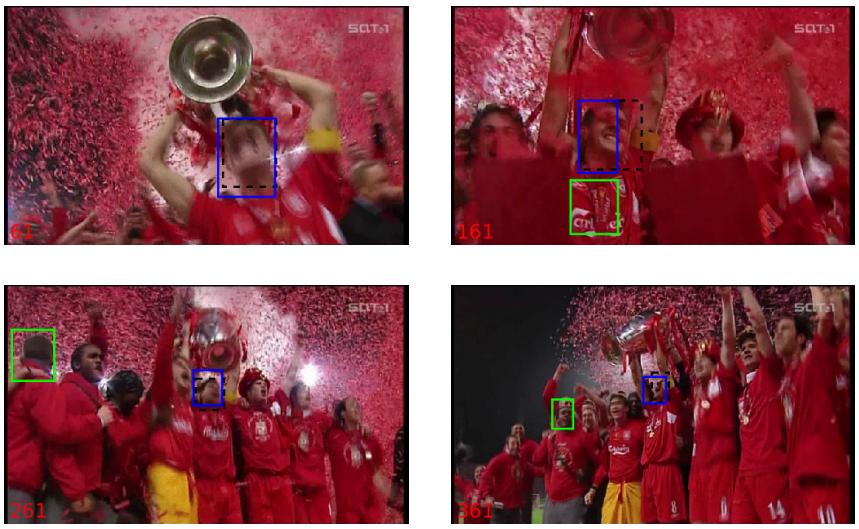


Figure 22: SIAMRPN++ (green), our implementation (blue), and ground truth (black) on 'soccer1' sequence of VOT2018.

4.4.3 Comparison with Related Work

To our knowledge, only [29] in the current literature addresses post-tracking drift detection and correction independently of tracker architecture. We therefore compare our system to our implementation of the GrabCut segmentation-based drift detection and correction proposed in [29]. Table 27 shows how both implementations perform on the entire VOT2018 and TC128 datasets and provides the percentage of improvement in FR measure over the GrabCut-based method. Our RPN-based drift detection and correction is denoted RPN DD/DC and the GrabCut-based drift detection and correction is denoted GrabCut DD/DC.

Stating that our drift detection and correction works better than that of [29] would be far-fetched, as it performs better in FR in only one case: TC128 with DASIAMRPN tracker. Table 27 actually shows that the GrabCut-based approach works better overall than our method. These results show the limitations of our RPN-based system, and they show the advantages of non-learning-based approaches such as GrabCut over deep-learning-based approaches. Whilst region proposal networks can outperform the state-of-the-art when trained on an extensive list of examples, the majority of those

FR		GrabCut DD/DC	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	0.19329	0.20786	-7.54%
	TC128	0.15983	0.15361	3.89%
SiamRPN++	VOT2018	0.1533	0.16123	-5.15%
	TC128	0.14156	0.14951	-5.61%

AOR		GrabCut DD/DC	RPN DD/DC	Improvement
DaSiamRPN	VOT2018	0.42412	0.42436	0.06%
	TC128	0.49729	0.51299	3.16%
SiamRPN++	VOT2018	0.43727	0.44117	0.89%
	TC128	0.49029	0.49159	0.27%

Table 27: VOT2018 and TC128 datasets: Mean FR, AOR, and percentage improvement of the RPN-based drift detection and correction over the GrabCut-based one of DASIAMRPN and SIAMRPN++.

examples are high definition images and contain commonly used labels. Applying a RPN on challenging video images with unknown target objects can pose a problem, as detecting a stationary object in an image is not similar to detecting an object that is undergoing challenges in motion, deformation, or illumination variation. We also notice that our method works better on the VOT2018 dataset, which is widely used in tracking evaluation, but performs more poorly on TC128, which is less known and does not contain as much high-definition content. In such scenarios, it could be favorable to use non-learning-based methods, such as GrabCut segmentation, to identify objectness characteristics, such as edges and salient regions, rather than objects.

Despite not performing as well in FR as the GrabCut-based drift correction [29] in FR, our method appears to be overall as accurate in terms of AOR. Table 27 shows that our drift correction yields a higher average AOR for each tracker and each dataset, although the percentage improvement is minor. This means that our method does not correct drift as frequently as [29], but when it does, it relocates the target object more effectively. In fact [29] only uses a saliency-based drift detection, which means it calls its drift correction on more false positives than our method, which relies

on both saliency and objectness measures. Thus, the GrabCut-based drift correction changes B_t more frequently, including instances where there is no drift or need for correction.

Besides our method and [29], we also tried combinations of both works, that is, we tested our drift detection with the drift correction from [29] and we tested the drift detection from [29] with our drift correction. However, both tests did not yield consistently better results. In fact, when we combined the drift detection from [29] with our RPN-based correction, the RPN was triggered too often in instances where the target object was not drifting. This combination is less efficient and does not take advantage of the RPN's search region to detect drifting objects. Quite contrarily, it could potentially lead to instances of wrongful object localization. In the case where we combined our drift detection with the GrabCut-based drift correction from [29], we also noticed that accuracy or robustness results were not better overall. The GrabCut segmentation-based drift correction required the target object to be consistently inside the tracker's bounding box. Since our drift detection executed correction when the bounding box most likely did not contain an object, those two processes did not combine well. Therefore, our drift detection is better suited for our RPN-based correction since it is better adapted for local re-detection tasks, whereas the drift detection and correction from [29] work better together as a bounding box regression task.

4.4.4 Limitations of Proposed Method

Although our drift detection and correction enhances a tracker's overall accuracy and robustness by 7.03% in terms of FR and 1.26% in terms of AOR across all VOT2018 and TC128 sequences combined, there are frames in some sequences where it fails to do so. We make the distinction between two cases of failure: (1) our system's overall performances do not change compared to the original tracker design, and (2)

our system under-performs compared to the original tracker design.

Case (1) where neither FR nor AOR performance measures are affected is less serious but it is still to be considered for matters of efficiency. It happens if drift is occurring but our method does not trigger drift correction. This means we falsely estimated the presence of drift. It also happens if drift correction is triggered but the bounding box is not relocated. This means either the RPN did not recognize any object in the vicinity of the search area A_{RPN}, or none of the candidate RPN proposals were chosen over B_t.

Case (2) where our method under-performs the baseline tracker happens if drift correction proposes a RPN candidate that overlaps less with the ground truth than B_t, that means when drift correction is successfully executed but is not needed.

Overall, looking at FR results of both DASIAMRPN and SIAMRPN++ trackers across all video sequences of VOT2018 and TC128 datasets, we identify a total of 13 sequences (3.44% of all test cases) where the percentage decrease in the average failure rate over all frames of our method compared to the baseline tracker is larger than 2.5%. One of these cases happens using SIAMRPN++ on TC128, one case happens using SIAMRPN++ on VOT2018, 4 cases happen using DASIAMRPN on TC128, and the 7 remaining cases happen using DASIAMRPN on VOT2018. We notice that our method performs less using DASIAMRPN than using SIAMRPN++. Figure 23 illustrates an example where the the drift correction localizes the wrong object after the target was occluded, and Figure 24 shows an instance where the drift correction fails at the start of the sequence. As in Figure 24, the target object can be recovered at later frames, but the recovery can be either tracker dependent (i.e., a tracker's object re-detection mechanism) or tracker independent (i.e., our drift correction).

One major reason our method can seriously fail is due to our saliency measures during drift detection. As it was shown on Figure 11 in section 4.3.1, our saliency-based drift detection does not entirely extract the foreground object and extracts the

Figure 23: Failure case: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'Kite1' sequence of TC128.

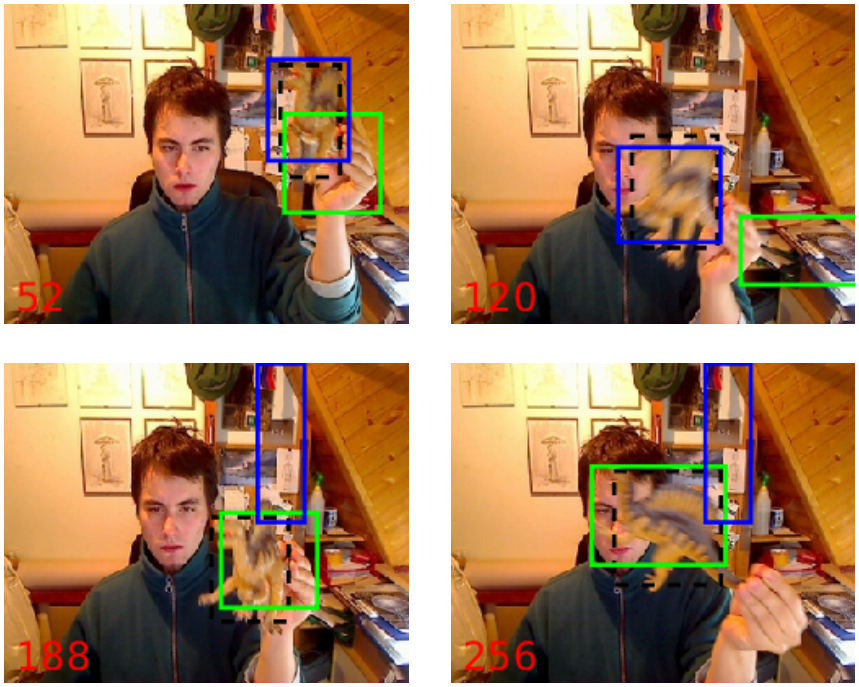


Figure 24: Failure case: DASIAMRPN (green), our implementation (blue), and ground truth (black) on 'dinosaur' sequence of VOT2018.

background at the contour of the target object. This affects (39) which has a decisive role in reinitializing the tracker output during drift correction by either keeping B_t as the system's output or replacing B_t with a RPN proposal. (39) uses the output of (33); thus, it is important that (33) well labels foreground superpixels as salient. Consequently, when (33) fails, the whole proposed method fails. To prevent scenarios similar to Figures 23 and 24 from happening, we need to better control the saliency estimation of superpixels, that is, reduce the number of background salient regions and increase the number of foreground salient regions.

4.5 Conclusion

This chapter investigated tracker drift. Even though target re-detection is being explored more in object tracking, most existing methods remain tracker-dependent. We looked into achieving drift detection and correction before drift happened. Drift detection was completed using a combination of saliency and objectness measures, and drift correction was then executed using a region proposal network. FR and AOR results of our implementation on DASIAMRPN and SIAMRPN++ baseline trackers showed that our method improved overall tracker robustness and accuracy measures by 7.03% in terms of FR and 1.26% in terms of AOR when tested on all sequences from VOT2018 and TC128 datasets.

Future contributions for drift correction may involve using deep-learning-based one shot video object segmentation [13] instead of a region proposal network. In this context, the target mask is collected at the first frame and would be the only data used as prior information.

In terms of drift detection, our saliency computations can be reworked in order to better extract the target object from the background. This means we need to improve the estimation of superpixel saliency (33). Note that (39) uses the output of (33); thus,

it is important that (33) well labels foreground superpixels as salient. Consequently, when (33) fails, the whole proposed method fails. Therefore, future work may include improving the selection of salient superpixels. This can be achieved first by only accepting superpixels that are within the boundaries of an absolute foreground and that contain salient feature points, in a similar way to [29]. To define the absolute foreground, we can compute the overall difference of pixel intensities between the foreground object and its background at a video's first frame. We then only pick the salient superpixels inside the absolute foreground that contain salient points, for example extracted SIFT feature points, and use those superpixels for (39). Using this strategy, we can (1) discard superpixels from the foreground and (2) capture a diverse set of superpixels that well describe the foreground object.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we explored two topics in video object tracking: performance evaluation of tracking techniques and tracker drift detection and correction.

When a tracker is newly introduced, its quality is compared with that of existing trackers based on its average performances. This method of evaluation does not use robust statistics or does not account for the presence of outliers. To facilitate the evaluation of object tracking techniques, we presented two robust scoring and ranking methods. Our first method effectively determined similarities between trackers and iteratively ranked them by using the median absolute deviation (MAD). Our second method used robust error norms, similarly to how they are employed in anisotropic diffusion for image denoising, to group trackers in the same piecewise uniform region of performance. Twenty trackers were scored and ranked according to their average overlap ratio and failure rate performance measures and across four different benchmarks.

Video object tracking is a sophisticated task which will inevitably drift or even fail as a result of various challenges such as illumination variation, object deformation,

motion blur, occlusion, and clutter, caused either by the environment or the target object. To tackle this issue, we first proposed a drift detection method which measures saliency and objectness features inside the tracker output at each frame. Then, we introduced a drift correction strategy that centers the output bounding box around the target object using a region proposal network. We embedded our implementation on two state-of-the-art trackers, tested the trackers on three benchmarks, and showed that our implementation improved the overall accuracy and robustness of the baseline tracker designs.

5.2 Future Work

In our scoring function in Chapter 3, the calculations of our scores were based on the mean average of a performance measure over all frames of a video sequence. In the future, we may include scoring to the frames themselves; then, we could calculate either the average score or apply another scoring function to these scores.

In terms of drift detection, our saliency computations can be reworked in order to better extract the target object from the background. This can be achieved first by only accepting superpixels that are within the boundaries of an absolute foreground and that contain salient feature points, in a similar way to [29]. To define the absolute foreground and distinguish it from the background, we can compute the overall difference of pixel intensities between the foreground object and its background at a video's first frame. We then only pick the salient superpixels inside the absolute foreground that contain salient points, for example SIFT features, and use those superpixels for equation (39).

Another contribution may involve using deep-learning-based one shot video object segmentation [13] instead of region proposal networks for drift correction. The only data that would be used as prior information is the target object's mask.

Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2189–2202, 2012.

[2] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2006.

[3] F. Bashir and F. Porikli. Imagery library for intelligent detection systems (i-lids); a standard for testing video based detection systems. In *IEEE International Carnahan Conference on Security Technology*, 2007.

[4] S. Bei, Z. Zhen, L. Wusheng, D. Liebo, and L. Qin. Visual object tracking challenges revisited: Vot vs. otb. *PLoS ONE*, 13(9), 2018.

[5] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV 2016 Workshops*, pages 850–865, 2016.

[7] M. J. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. In *IEEE Trans. Image Process.*, volume 7, pages 421–432, 1998.

[8] B. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[9] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *IEEE Int. Conf. Computer Vision*, pages 105–112, 2001.

[10] K. Brahim, R. Kalboussi, and M. Abdellaoui. Spatio-temporal saliency detection using objectness measure. In *SIViP*, pages 1055–1062, 2019.

[11] N. D. B. Bruce and J. K. Tsotsos. Saliency based on information maximization. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1–8. ACM, 2006.

[12] N. D. B. Bruce and J. K. Tsotsos. Saliency attention and visual search: An information theoretic approach. In *J. Vis*, volume 9, 2009.

[13] S. Caelles, K. K. Maninis, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] B. Cai, X. Xu, X. Xing, K. Jia, J. Miao, and D. Tao. Bit: Biologically inspired tracker. *IEEE Transactions on Image Processing*, 25(3):1327–1339, 2016.

[15] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018.

[16] B. Cao, X. Meng, S. Zhu, and B. Zengv. Salient object detection based on image bit-map. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2603–2607, 2020.

[17] L. Cehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited,. In *IEEE Int. Conf. Image Processing (ICIP)*, volume 25, pages 1261–1274, 2016.

[18] L. Cehovin, A. Lukezic, A. Leonardis, and M. Kristan. Beyond standard benchmarks: Parameterizing performance evaluation in visual object tracking. In *IEEE Int. Conf. Computer Vision*, 2017.

[19] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[20] M.-M. Cheng, N. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu. Global contrast based salient region detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 37, pages 569–582, 2015.

[21] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *IEEE CVPR*, 2014.

[22] B. D. Cullity. *The rating of chessplayers, past and present*. Ishi Press, Batsford London, 1978.

[23] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang. High-performance long-term tracking with meta-updater. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Atom: Accurate tracking by overlap maximization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4655–4664, 2019.

[25] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

[26] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[27] Y. Fang, W. Lin, Z. Chen, C.-M. Tsai, and C.-W. Lin. A video saliency detection model in compressed domain. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 24, 2014.

[28] R. Fisher, J. Santos-Victor, and J. Crowley. Caviar: Context aware vision using image-based active recognition, 2004.

[29] T. Ghoniemy and M. A. Amer. Drift detection and correction post-tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2333–2337, 2020.

[30] T. Ghoniemy, J. Valognes, and M. A. Amer. Robust scoring and ranking of object tracking techniques. In *IEEE International Conference on Image Processing (ICIP)*, pages 236–240, 2018.

[31] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[33] M. Glickoman. Parameter estimation in large dynamic paired comparison experiments. In *Journal of the Royal Statistical Society Series C Applied Statistics*, pages 377–394, 1999.

[34] N. Goyette, P.M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *IEEE CVPR Workshops*, pages 1–8, 2012.

[35] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York, 1986.

[36] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[38] Z. He, Y. Fan, J. Zhuang, Y. Dong, and H. Bai. Correlation filters with weighted convolution responses. *IEEE International Conference on Computer Vision*, 2017.

[39] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *Proceedings of the European Conference on Computer Vision*, pages 749–765, 2016.

[40] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI - IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[41] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[42] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. Need for speed: A benchmark for higher frame rate object tracking. *arXiv preprint arXiv:1703.05884*, 2017.

[43] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The visual object tracking vot2013 challenge results. In *Proc. IEEE Int. Conf. Computer Vision Workshops*, pages 98–111, 2013.

[44] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The visual object tracking vot2014 challenge results. In *Proc. IEEE European Conf. Computer Vision Workshops*, pages 191–217, 2014.

[45] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The visual object tracking vot2015 challenge results. In *Proc. IEEE Int. Conf. Computer Vision Workshops*, pages 564–586, 2015.

[46] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The visual object tracking vot2016 challenge results. In *Proc. IEEE European Conf. Computer Vision Workshops*, pages 777–823, 2016.

[47] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The visual object tracking vot2017 challenge results. In *Proc. IEEE Int. Conf. Computer Vision Workshops*, pages 1949–1972, 2017.

[48] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The sixth visual object tracking vot2018 challenge results. In *Proc. IEEE European Conf. Computer Vision Workshops*, 2018.

[49] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernandez. The sixth visual object tracking vot2018 challenge results. 2018.

[50] H. Lee, S. Choi, and Kim C. A memory model based on the siamese network for long-term tracking. In *The European Conference on Computer Vision (ECCV) Workshops*, 2018.

[51] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

[53] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, 2018.

[54] F. Li, Y. Yao, P. Li, D. Zhang, W. Zuo, and M.-H. Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ICCVW*, 2017.

[55] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. In *ACM Trans. Intell. Syst. Technol.*, volume 4, pages 1–48, 2013.

[56] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 280–287, 2014.

[57] Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, 2015.

[58] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: Algorithms and benchmark. In *IEEE Trans. on Image Processing*, 2015.

[59] T. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[60] T. Liu, J. Sun, N.-N. Zhen, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[61] A. Lukežič, T. Vojir, L. Zajc, J. Matas, and M. Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 2018.

[62] A. Lukezic, L.C. Zajc, T. Vojir, J. Matas, and M. Kristan. Now you see me: Evaluating performance in long-term visual tracking. *CoRR abs/1804.07056*, 2018.

[63] Kristan M., J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Zajc, O. Drbohlav, A. Lukezic, A. Berg, A. Eldesokey, J. Kapyla, and G. Fernandez. The seventh visual object tracking vot2019 challenge results. In *Proc. IEEE European Conf. Computer Vision Workshops*, 2019.

[64] A. Moudgil and V. Gandhi. Long-term visual object tracking benchmark. In *Asian Conference on Computer Vision*, pages 629–645. Springer, 2018.

[65] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[66] D. J. Olive. *Applied Robust Statistics*. PhD thesis, University of Minnesota, 1998.

[67] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford InfoLab*, 1999.

[68] Y. Pang and H. Ling. Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms. In *IEEE International Conference on Computer Vision*, 2013.

[69] K. E. Papoutsakis and A. Argyros. Integrating tracking with fine object segmentation. In *Image Vis. Comput.*, volume 31, pages 675–679, 2013.

[70] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. In *IEEE Trans. Pattern Anal. Machine Intell.*, volume 12, pages 629–639, 1990.

[71] P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1090–1104, 2000.

[72] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[73] L. Pu, X. Feng, Z. Hou, W. Yu, Y. Zha, and S. Ma. Deep correlation filter based real-time tracker. In *IEEE International Conference on Information Fusion*, 2019.

[74] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[75] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[76] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, New York, 2006.

[77] A. W. M. 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*, 2014.

[78] F. Solera, S. Calderara, and R. Cucchiara. Towards the evaluation of reproducible robustness in tracking-by-detection. In *Advanced Video and Signal Based Surveillance*, pages 1–6, 2015.

[79] Y. Song, C. Ma, L. Gong, J. Zhang, R. Lau, and M.-H. Yang. Crest: Convolutional residual learning for visual tracking. In *IEEE International Conference on Computer Vision*, pages 2555–2564, 2017.

[80] S. Srivatsa and V. Babu. Salient object detection via objectness measure. In *IEEE International Conference on Image Processing*, pages 4481–4485, 2015.

[81] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[82] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. Smeulders, P. Torr, and E. Gavves. Long-term tracking in the wild: A benchmark. In *ECCV*, 2018.

[83] L. Čehovin, M. Kristan, and L. Aleš. Is my new tracker really better than yours? In *IEEE Winter Conference on Applications of Computer Vision*, pages 540–547, 2014.

[84] P. T. Von Hippel. Mean, median, and skew: Correcting a textbook rule. *Journal of Statistics Education*, 13(2), 2005.

[85] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin. Region proposal by guided anchoring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2960–2969, 2019.

[86] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, 2018.

[87] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[88] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. 2019.

[89] L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. Jots: Joint online tracking and segmentation. In *IEEE Conf. Computer Vision Pattern Recognition*, pages 2226–2234, 2015.

[90] Y. Wu, J. Lim, and M. H. Yang. Object tracking: A benchmark. In *CVPR*, 2013.

[91] Y. Wu, J. Lim, and M. H. Yang. Object tracking benchmark. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37:1442–1468, 2015.

[92] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual tracking. *IEEE Transactions on Image Processing*, 28(11):5596–5609, 2019.

[93] L. Yang and Z. Jianke. A scale adaptive kernel correlation filter tracker with feature integration. In *Proc. European Conf. Computer Vision*, pages 254–265, 2014.

[94] L. Yang, R. Liu, D. Zhang, and L. Zhang. Deep location-specific tracking. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1309–1317. ACM, 2017.

[95] T. Yang, X. Zhang, Z. Li, and W. Zhang. Metaanchor: Learning to detect objects with customized anchors. In *Advances in Neural Information Processing Systems*, 2018.

[96] M. Zeeshan, M. Majid, I. F. Nizami, and S. Muhamma. A newly developed ground truth dataset for visual saliency in videos. In *Access*, volume 41, pages 20855–20867, 2018.

[97] L. Zhang, M. Danelljan, A. Gonzalez-Garcia, J. Van de Weijer, and F. Shahbaz Khan. Multi-modal fusion for end-to-end rgb-t tracking. In *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 2252–2261, 2019.

[98] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye. Freeanchor: Learning to match anchors for visual object detection. In *Neural Information Processing Systems*, 2019.

[99] Z. Zhang and H. Peng. Deeper and wider siamese networks for real-time visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4591–4600, 2019.

[100] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision*, 2018.

[101] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.