Object Tracking Benchmark

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · September 2015

DOI: 10.1109/TPAMI.2014.2388226

CITATIONS
967

READS
7,457

3 authors, including:

Jongwoo Lim
Hanyang University
57 PUBLICATIONS **7,369** CITATIONS

SEE PROFILE

Ming-Hsuan Yang
University of California, Merced
354 PUBLICATIONS **21,211** CITATIONS

SEE PROFILE

Object Tracking Benchmark

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang

Abstract—Object tracking has been one of the most important and active research areas in the field of computer vision. A large number of tracking algorithms have been proposed in recent years with demonstrated success. However, the set of sequences used for evaluation is often not sufficient or is sometimes biased for certain types of algorithms. Many datasets do not have common ground-truth object positions or extents, and this makes comparisons among the reported quantitative results difficult. In addition, the initial conditions or parameters of the evaluated tracking algorithms are not the same, and thus, the quantitative results reported in literature are incomparable or sometimes contradictory. To address these issues, we carry out an extensive evaluation of the state-of-the-art online object-tracking algorithms with various evaluation criteria to understand how these methods perform within the same framework. In this work, we first construct a large dataset with ground-truth object positions and extents for tracking and introduce the sequence attributes for the performance analysis. Second, we integrate most of the publicly available trackers into one code library with uniform input and output formats to facilitate large-scale performance evaluation. Third, we extensively evaluate the performance of 31 algorithms on 100 sequences with different initialization settings. By analyzing the quantitative results, we identify effective approaches for robust tracking and provide potential future research directions in this field.

Index Terms—Object tracking, benchmark dataset, performance evaluation

1 Introduction

Object tracking is one of the most important problems in the field of computer vision with applications ranging from surveillance and human-computer interactions to medical imaging [89], [13]. Given the initial state (e.g., position and extent) of a target object in the first image, the goal of tracking is to estimate the states of the target in the subsequent frames. Although object tracking has been studied for several decades and considerable progress has been made in recent years [36], [17], [65], [5], [54], [32], [35], it remains a challenging problem. Numerous factors affect the performance of a tracking algorithm, including illumination variation, occlusion, and background clutters, and there exists no single approach that successfully handles *all* scenarios. Therefore, it is crucial to thoroughly evaluate the state-of-the-art tracking algorithms to demonstrate their strength and weakness, thereby identifying future research directions in this field for more robust algorithms.

For a comprehensive performance evaluation, it is critical to collect a representative dataset. There exist several datasets for object tracking in surveillance scenarios, such as the VIVID [14], CAVIAR [24], and PETS [23] databases. However, in these surveillance sequences, the target objects are usually humans or small cars and the background is usually static. For generic scenes with various types of tracking targets, many of the available sequences do not provide the ground-truth target locations except a few datasets [65], [5], [43]. The reported tracking results on these unlabeled datasets are not directly comparable since different ground-truth annotations are used.

Recently, significant efforts have been made to make tracking code available for evaluation, e.g., OAB [27], IVT [65], MIL [5], L1 [53], and TLD [39] algorithms. These tracking algorithms accommodate different input formats (e.g., avi videos or raw image sequences) and motion models (e.g., 2D translation, similarity or affine transforms) with various output formats (e.g., position or extent). Therefore, to evaluate the performance of these algorithms on a large number of image sequences, it is necessary to integrate them in a library for evaluation on a common platform. In this work, we integrate most of the publicly available trackers in a code library with uniform input and output formats for a performance evaluation. In addition, we provide a large benchmark dataset with ground-truth annotations and attributes (e.g., occlusion, fast motion, or illumination variation) such that the performance of the evaluated tracking algorithms can be better analyzed.

One common issue in assessing tracking algorithms is that the reported results are often based on a few sequences with different initializations or parameters. Inaccurate localization of the target occurs frequently as an object detector may be used for locating the object in the first frame. In addition, an object detector may be used to recover from tracking failures by re-initializing the tracker. For a fair and comprehensive evaluation, we propose to perturb the initial object states spatially and temporally on the basis of the ground-truth target locations. With this evaluation methodology, the sensitivity of a tracking algorithm to initialization (i.e., robustness) can be better analyzed. While the robustness to initialization is a known problem in other fields, it has not been well addressed in the literature of object tracking. To the best of our knowledge, this is the first work to comprehensively address and analyze the initialization problem of object tracking.

The contributions of this work are three-fold:
Benchmark dataset. We construct a benchmark dataset

with 100 fully annotated sequences[1] to facilitate the performance evaluation.

Code library. We integrate most of the publicly available trackers in one code library with unified input and output formats to facilitate a large-scale performance evaluation.

Performance evaluation. We propose novel metrics to evaluate tracking algorithms where the initial object states are perturbed spatially and temporally for the robustness analysis. Each algorithm is extensively evaluated by analyzing more than millions of tracking outputs.

This work[2] mainly focuses on the performance evaluation of online[3] tracking algorithms for single targets. The code library, annotated dataset, and all the tracking results are available at http://pami.visual-tracking.net.

2 Brief Review of Object Tracking

Considerable progress in the field of object tracking has been made in the past few decades. In this section, we review some recent algorithms for object tracking in terms of target representation scheme, search mechanism, and model update. In addition, several methods that build upon combing some trackers or mining context information have been proposed. Here, we discuss the relevant performance evaluation work on object tracking and challenging factors in object tracking.

2.1 Representation Scheme

Object representation is one of the major components in any visual tracking algorithm, and numerous schemes have been proposed [46]. Since the early work of Lucas and Kanade (LK) [50], holistic templates (based on raw intensity values) have been widely used for tracking [52], [2]. The LK approaches [50], [7] do not take large appearance variability into account and thus, do not perform well when the visual properties of a target object change significantly. Matthews *et al.* [52] developed a template update method by exploiting the information of the first frame to correct drifts. To better account for appearance changes, subspace-based tracking approaches [30], [12], [65], [54] have been proposed. In [30], Hager and Belhumeur proposed an efficient LK algorithm and used low-dimensional representations for tracking under varying illumination conditions. For enhancing the tracking robustness, Black and Jepson [12] adopted a robust error norm and proposed an algorithm using a pre-trained view-based eigenbasis representation. In [65], a low-dimensional subspace representation was learned incrementally to account for target appearance variation for object tracking [36].

Recently, numerous tracking methods based on sparse representations have been proposed [53], [55], [94], [95], [38], [79]. Mei and Ling [53], [54] used a dictionary of holistic intensity templates composed of target and trivial templates, and determined the target location by solving multiple ℓ_1 minimization problems. To better handle occlusion and improve run-time performance, local sparse representations and collaborative representations have also been introduced for object tracking [94], [95], [38], [9]. For run-time efficiency, a minimal error bounding strategy was introduced [55] to reduce the number of ℓ_1 minimization problems to solve. Bao *et al.* [9] introduced the accelerated proximal gradient approach to efficiently solve ℓ_1 minimization problems. To enhance tracking robustness, a local sparse appearance model was proposed in [48] with the mean shift algorithm to locate objects. By assuming the representation of particles as jointly sparse, Zhang *et al.* [94] formulated object tracking as a multi-task sparse learning problem. Zhong *et al.* [95] proposed a collaborative tracking algorithm that combined a sparsity-based discriminative classifier and a sparsity-based generative model. In [38], sparse codes of local image patches with spatial layout in an object were used for modeling the object appearance for tracking. To deal with outliers for object tracking, Wang *et al.* [78] proposed a least soft-threshold squares algorithm by modeling image noise with the Gaussian-Laplacian distribution other than the trivial templates used in [53].

A number of tracking methods based on color histograms [17], [60] have been developed. Comaniciu *et al.* [17] applied the mean shift algorithm to object tracking on the basis of a color histogram. Collins [15] extended the mean shift tracking algorithm to deal with the scale variation of target objects. In [60], Perez *et al.* embedded a color histogram in a particle filter [36] for object tracking. Instead of relying on pixel-wise statistics, Birchfield and Rangarajan [11] proposed the spatiogram to capture both the statistical properties of pixels and their spatial relationships. A locality sensitive histogram [33] was developed by considering the contribution of local regions at each pixel to better describe the visual appearance for object tracking. To exploit local directional edge information, histograms of oriented gradients (HOGs) [18] have been adopted for tracking [73] with the integral histogram [62]. To fuse different types of features, representations based on covariance region descriptors [74] were introduced for object tracking. In covariance descriptors, the spatial and statistical properties as well as their correlations are characterized within the same representation. In addition, the local binary patterns (LBP) [57] and Haar-like features [75] have also been utilized to model the object appearance for tracking [27], [5], [92], [32].

Recently, discriminative models have been developed in the field of object tracking [3], [4], where a binary classifier is learned online to separate the target from the background. Numerous classifiers have been adapted for object tracking, such as support vector machine (SVM) [3], structured output SVM [32], ranking SVM [6], boosting [27], semi-boosting [28], and online multi-instance boosting [5]. To handle appearance changes, Avidan [3] integrated a trained SVM classifier in an optical flow framework for tracking. In [16], the most discriminative feature combination was

1. Each sequence denotes a trajectory of one target object. If one video contains two target objects, it is considered two sequences.

2. Preliminary results of this work are presented in [83].

3. For *online* tracking algorithms, only the information of the previous few frames is used for inference at any time instance.

learned online to build a confidence map in each frame for separating a target object from the background. In [4], an ensemble of online learned weak classifiers was used to determine whether a pixel belonged to the target region or the background. Grabner *et al.* [27] proposed an online boosting method to select discriminative features for the separation of a foreground object and the background. To balance between tracking adaptivity and drifting, Stalder *et al.* [71] combined multiple supervised and semi-supervised classifiers for tracking. In [6], object tracking is posed as a weakly supervised ranking problem by capturing the relative proximity relationships between samples toward the true target samples. To alleviate the drifting problem, a semi-online boosting algorithm has been developed for tracking [28]. In [39], Kalal *et al.* presented a learning method guided by positive and negative constraints to distinguish a target object from the background. Multiple instance learning (MIL) has also been applied to tracking [5], where all ambiguous positive and negative samples are put into bags to learn a discriminative model. Hare *et al.* [32] designed a tracking algorithm based on a kernelized structured SVM, which exploits the constraints of the predicted outputs.

To account for an appearance change caused by a large pose variation and heavy occlusion, an object can be represented by parts with descriptors or histograms. In [1], local histograms were used for representing a target object in a defined grid structure. Kwon and Lee [42] have proposed a tracking method that updates the topology of local patches to handle large pose changes.

Several approaches based on multiple representation schemes have been developed [72], [43] to better handle appearance variations. Stenger *et al.* [72] fused multiple observation models online in a parallel or cascaded manner. Recently, Kwon and Lee [43] have developed an object-tracking decomposition algorithm that used multiple observation and motion models to account for a relatively large appearance variation caused by drastic lighting changes and fast motion. This approach has been further extended to search for appropriate trackers by Markov Chain Monte Carlo sampling [44].

2.2 Search Mechanism

Deterministic and stochastic search methods have been developed to estimate the object states. When the tracking problem is posed within an optimization framework with an objective function differentiable with respect to motion parameters, gradient descent methods can be used for locating the target efficiently [50], [17], [22], [69]. In [50], the first-order Taylor expansion is used to linearize the nonlinear cost function, and the motion parameters are estimated iteratively. Further, a mean shift estimation is used for searching the target locally by using the Bhattacharyya coefficient as the similarity metric for kernel-regularized color histograms [17]. In [22], Fan *et al.* adopted a discriminative approach to identify spatial attentional regions with a gradient-based formulation to locate objects. In [69], Lara and Miller proposed a tracking algorithm based on distribution fields, which allow smoothing the objective function without blurring the image, and the target is located by searching for the local minimum by using a coarse-to-fine strategy.

However, objective functions for object tracking are usually nonlinear with many local minima. To alleviate this problem, dense sampling methods have been adopted [27], [5], [32] at the expense of a high computational load. On the other hand, stochastic search algorithms such as particle filters [36], [60] have been widely used since they are relatively insensitive to the local minimum and are computationally efficient. Recent methods based on particle filters have been developed using effective observation models [65], [53], [38] with demonstrated success.

2.3 Model Update

It has been shown that online update of target representation to account for appearance variations plays an important role for robust object tracking [37], [27], [65], [32], [38]. Matthews *et al.* [52] addressed the template update problem for the LK algorithm [50], where the template was updated with the combination of the fixed reference template extracted from the first frame and the result from the most recent frame. Effective update algorithms have also been proposed in the form of the online mixture model [37], online boosting [27], and incremental subspace update [65].

For the discriminative model, recently, considerable attention has been paid to draw samples effective for training online classifiers [28], [5], [39], [32]. In contrast to supervised discriminative object tracking, Grabner *et al.* [28] formulated the update problem as a semi-supervised task where the classifier was updated with both labeled and unlabeled data. To handle ambiguously labeled positive and negative samples obtained online, Babenko *et al.* [5] focused on the tracking problem within the multiple instance learning (MIL) framework and developed an online algorithm. To exploit the underlying structure of the unlabeled data, Kalal *et al.* [39] developed a tracking algorithm within the semi-supervised learning framework to select positive and negative samples for model update. In [32], the proposed tracking algorithm directly predicts the target location change between frames on the basis of structured learning. Yu *et al.* [91] presented a tracking method based on co-training to combine generative and discriminative models. While considerable progress has been made, it is still difficult to develop an adaptive appearance model without drifts.

2.4 Context and Fusion of Trackers

Context information facilitates object tracking by providing distinct visual properties of the immediate surroundings of the targets. Numerous approaches have been developed by mining auxiliary or local visual information surrounding the target objects to assist tracking [88], [29], [20]. In [88], auxiliary objects were automatically discovered and tracked for the verification of the target. Dinh *et al.* [20] exploited

distractors and supporters around a target object by using a sequential randomized forest. The context information is useful when the target objects are fully occluded or out of the camera views [29].

To improve the tracking performance, some fusion methods have been developed. Santner *et al.* [68] proposed an approach that combines static, moderately and highly adaptive trackers to account for appearance changes. In [44], tracking modules sampled from a predefined tracker space interact with each other to deal with appearance variations caused by lighting changes and fast motion. Multiple feature sets [90] were maintained and selected in a Bayesian framework to account for the appearance changes.

2.5 Performance Evaluation

Evaluation of tracking approaches is of critical importance, and efforts have been made to assess the performance of different tracking algorithms [80]. In [81], Wu *et al.* proposed an approach using a time-reversed Markov chain to evaluate tracking algorithms in the absence of annotations. Further, a number of tracking approaches based on sparse representation were compared in [93]. To evaluate the appearance model of tracking algorithms, Salti *et al.* [67] introduced a unified conceptual framework and presented an experimental analysis. However, the number of evaluated tracking algorithms and image sequences in the above-mentioned work are quite limited.

Most recently, Smeulders *et al.* [70] have evaluated 19 trackers on 315 videos. Although it has been noted that poor initialization of a tracker significantly decreases the tracking accuracy, further analysis based on comprehensive experimental evaluations is necessary and important to better understand the state-of-the-art algorithms. In [59], Pang and Ling used a ranking approach to analyze the reported results of object tracking methods. The main shortcoming of this approach is that it is not appropriate to rank the recently published trackers due to a lack of sufficient experimental evaluations in terms of trackers, sequences, and metrics. In [40], 27 trackers were evaluated on 16 sequences, where the ranking of trackers were obtained by averaging the performance on image sequences using different metrics. The failure rate of a tracking method was computed by counting the number of frames in which a method fails to follow a target object. A tracker was re-initialized several frames after a failure occurs (when the overlap ratio is zero). If a different overlap threshold is used, each tracker needs to be re-evaluated on the entire dataset. In this work, we propose a virtual run to compute failure rates with different overlap thresholds.

2.6 Challenging Factors

We discuss some main tracking challenges in this section.

Occlusion. Numerous approaches including part-based local representations have been proposed in recent years to handle the appearance changes caused by heavy occlusion. By adopting a sparse representation of the target and trivial templates, Mei *et al.* [55] developed a method to detect object occlusion by analyzing coefficients for appropriate template update in order to reduce the tracking drift. In [39], Kalal *et al.* developed a re-initialization module to detect whether a target object is completely occluded or missing for robust tracking.

Deformation. To model non-rigid appearance changes, histogram representations of color and intensity have been used for object tracking [17], [56], [31], [42], [26]. In [56], Nejhum *et al.* modeled the target appearance by using a small number of rectangular blocks from which histograms were extracted. The positions of these blocks within an object were adaptively determined for object tracking. In [42], a target object was represented by a patch-based appearance model and the topology between local patches was updated online. Another effective approach to address the deformation problem is based on segmentation techniques to describe object shape. In [26], Godec *et al.* presented a tracking-by-detection approach based on a generalized Hough transform and used segmentation based on the GrabCut method [66] to better describe the foreground objects.

Scale Variation. To estimate the scale of a target object, one common approach is to search at multiple scales and use the one with the maximum likelihood for tracking [17]. Collins [15] used the scale space theory [47] to improve the mean-shift tracking method [17]. Another approach is to include object scale as one state in the motion model (e.g., similarity or affine transformation). In the tracking methods based on particle filters, object states are often estimated by the average of a few particles with large weights (likelihoods) [36], [65].

Fast Motion. Most tracking methods operate on the implicit assumption that objects move without abrupt or sudden movements. In [63], Porikli and Tuzel extended the mean-shift tracking method by using multiple kernels centered around fast motion areas. In [41], Kwon and Lee introduced the Wang-Landau Monte Carlo sampling method to handle fast motion by alleviating motion smoothness constraints with both the likelihood functions and the density of states. To cope with abrupt motion and large appearance changes, multiple trackers with different motion and appearance models were used where the best one was selected using Markov Chain Monte Carlo sampling [43], [44].

3 Evaluated Tracking Algorithms

We evaluated 31 representative tracking algorithms in this work (See Table 1). As all implementations inevitably involve technical details and specific parameter settings, we included the algorithms only if the original source or binary code was publicly available[4]. While some algorithms are not included due to accessibility issues, the set of evaluated trackers is representative for an accurate assessment of the

4. In addition, we implemented the tracking algorithms [60], [17]. We evaluated the trackers in the VIVID dataset [14], including the mean shift (MS), template matching (TM), ratio shift (RS), and peak difference (PD) methods.

TABLE 1: Evaluated tracking algorithms.

	Representation										Search				Code			
	Local	Template	Color	Histogram	Subspace	Sparse	Binary or Haar	Discriminative	Generative	Model Update	Particle Filter	MCMC	Local Optimum	Dense Sampling	C	Matlab	FPS	Publication
ASLA [38]	✓	✓			✓	✓			✓	✓	✓				✓	✓	8.5	'12
BSBT [71]							H	✓		✓				✓	✓		7.0	'09
CPF [60]	✓		✓	✓					✓		✓						109	'02
CSK [34]		✓						✓		✓				✓		✓	362	'12
CT [92]							H	✓		✓				✓	✓	✓	64.4	'12
CXT [20]							B	✓		✓				✓	✓		15.3	'11
DFT [69]	✓	✓							✓	✓			✓			✓	13.2	'12
FOT [76]	✓	✓							✓	✓			✓		✓		244	'11
FRAG [1]	✓			✓					✓					✓	✓		6.3	'06
IVT [65]		✓			✓				✓	✓	✓				✓	✓	33.4	'08
KMS [17]			✓	✓					✓				✓	✓	✓		3,159	'03
L1APG [9]		✓			✓	✓			✓	✓	✓				✓	✓	2.0	'12
LOT [58]	✓		✓						✓	✓	✓					✓	0.7	'12
LSHT [33]	✓		✓	✓			H		✓	✓				✓	✓		20	'13
LSK [48]	✓	✓			✓	✓			✓				✓			✓	5.5	'11
LSS [78]	✓	✓			✓	✓			✓	✓	✓					✓	15	'13
MIL [5]							H	✓		✓				✓	✓		38.1	'09
MTT [94]		✓			✓	✓			✓	✓	✓				✓		1.0	'12
OAB [27]							H	✓		✓				✓	✓		22.4	'06
ORIA [85]		✓			✓		H	✓		✓				✓	✓		20.2	'11
PCOM [77]		✓			✓	✓			✓	✓	✓					✓	20	'14
SCM [95]	✓	✓			✓	✓		✓	✓	✓	✓			✓	✓	✓	0.51	'12
SMS [15]			✓	✓					✓				✓		✓		19.2	'03
SBT [28]							H	✓		✓				✓	✓		11.2	'08
STRUCK [32]							H	✓		✓				✓	✓		20.2	'11
TLD [39]	✓						B	✓		✓				✓	✓	✓	28.1	'10
VR [16]			✓					✓		✓			✓		✓		109	'05
VTD [43]		✓	✓		✓				✓	✓		✓			✓	✓	5.7	'10
VTS [44]	✓	✓	✓		✓				✓	✓		✓			✓	✓	5.7	'11

state-of-the-art in object tracking. For large-scale evaluations, we built a tracker library by modifying the interface of the source code with unified input and output formats. This code library provides a platform for evaluating the state-of-the-art trackers.

For fair comparisons, the parameters of each tracker were fixed for all the considered sequences. In most cases, we used the default parameters provided in the source code[5]. It was difficult, if not impossible, to tune the parameters of each tracker in this large-scale evaluation. The evaluation results in this work could be viewed as the average or lower bound of the tracking performance. While numerous diverse trackers have been proposed, the best performing approaches share similar components. Table 1 lists the main characteristics of the evaluated tracking algorithms.

4 Datasets

Recently, numerous benchmark datasets have been developed for various vision problems, such as the Berkeley segmentation [51], FERET face recognition [61], and optical flow dataset [8]. There exist a few benchmark datasets for tracking in the surveillance scenarios, such as the VIVID [14] and CAVIAR [24] databases. For generic object tracking, several sequences have been used for the evaluation [65], [5], [43]. However, most image sequences are not appropriately provided with the ground-truth annotations, and thus, the reported quantitative results in the literature are different or inconsistent since the trackers are not initialized and evaluated on the same platform. To facilitate a fair performance evaluation, we collected and annotated most of the commonly used tracking sequences. This work expands the sequences that we collected in [83] to include 100 target objects in the tracking benchmark TB-100 dataset[6]. Since some of the target objects are similar or less challenging, we also selected 50 difficult and representative ones in the TB-50 dataset for an in-depth analysis. Note that as humans are the most important target objects in practice, the TB-100 dataset contains more sequences of this category (36 body and 26 face/head videos) than of the other types.

Attributes of a test sequence. An evaluation of tracking algorithms is challenging as many factors affect the experimental results. For a better analysis of the strength and the weakness of the tracking algorithms, we categorized the sequences according to the 11 attributes defined in Table 2. Each attribute represents a specific challenging factor in object tracking. One sequence may be annotated with many attributes, and some attributes occur more frequently than others.

In addition to the performance evaluation on the TB-100 dataset, we report tracking results of sequences with specific attributes. The characteristics of tracking algorithms can be better understood from the sequences with the same attributes. For example, to evaluate how well the tracker handles occlusion, one may use 49 sequences (29 in TB-50) annotated with the OCC attribute. Figure 1 shows the

5. Some trackers (e.g., IVT) provide different parameters for different sequences in the source code. In such a case, one default parameter setting was selected for all the experiments.

6. The number of videos is less than 100 since a few sequences have multiple targets in them.

TABLE 2: Annotated sequence attributes with the threshold values in the performance evaluation.

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

TABLE 3: Attribute distribution. The diagonal corresponds to the distribution over the entire dataset, and each row or column presents the distribution for the attribute subset.

(a) Attribute distribution for TB-100.

TB-100	IV	OPR	SV	OCC	DEF	MB	FM	IPR	OV	BC	LR
IV	**38**	24	24	20	15	12	12	17	5	17	2
OPR	24	**63**	45	38	29	16	24	42	11	19	7
SV	24	45	**64**	33	29	21	28	35	11	17	9
OCC	20	38	33	**49**	25	14	19	25	12	14	5
DEF	15	29	29	25	**44**	10	15	17	5	12	3
MB	12	16	21	14	10	**29**	24	16	8	8	1
FM	12	24	28	19	15	24	**39**	22	11	10	2
IPR	17	42	35	25	17	16	22	**51**	8	14	6
OV	5	11	11	12	5	8	11	8	**14**	6	2
BC	17	19	17	14	12	8	10	14	6	**31**	1
LR	2	7	9	5	3	1	2	6	2	1	**9**

(b) Attribute distribution for TB-50.

TB-50	IV	OPR	SV	OCC	DEF	MB	FM	IPR	OV	BC	LR
IV	**22**	15	16	11	10	9	8	12	4	14	2
OPR	15	**32**	26	25	16	10	15	22	9	14	6
SV	16	26	**38**	25	17	15	20	22	9	14	8
OCC	11	25	25	**29**	15	11	14	17	10	11	5
DEF	10	16	17	15	**23**	6	10	10	5	7	3
MB	9	10	15	11	6	**19**	16	13	7	7	1
FM	8	15	20	14	10	16	**25**	14	9	9	2
IPR	12	22	22	17	10	13	14	**29**	7	12	5
OV	4	9	9	10	5	7	9	7	**11**	5	2
BC	14	14	14	11	7	7	9	12	5	**20**	1
LR	2	6	8	5	3	1	2	5	2	1	**8**

first frames of all 100 targets with ground-truth bounding boxes and attributes where the target objects of the TB-50 dataset are marked with green rectangles. The attribute distribution of the TB-100 dataset is shown in Table 3.

5 Evaluation Methodology

It is a challenging task to assess the performance of a tracking algorithm with quantitative metrics. Many factors such as position accuracy, robustness over a certain type of appearance changes, tracking speed, memory requirement, and ease of use can be considered. Even in one frame with the tracking output and ground-truth object state, there can be several metrics to measure accuracy. When an algorithm loses track of the target object, it may resume to track after failure if there exists a re-detection module, or if it fortuitously locates the target object again as the object reappears at the position where the tracking bounding box is. If we simply average the evaluated values of all frames in an image sequence, the evaluation may not be fair since a tracker may have lost the target in the beginning but could have tracked the target successfully if it were initialized in a object state or frame. We first discuss a few performance measures that are commonly used and then, propose new metrics to evaluate whether tracking algorithms perform robustly under different conditions.

Precision plot. One widely used evaluation metric for object tracking is the center location error, which computes the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground-truth positions of all the frames. When a tracking algorithm loses track of a target object, the output location can be random, and thus, the average error value does not measure the tracking performance correctly [5]. Instead, the percentage of frames in which the estimated locations are within a given threshold distance of the ground-truth positions is a better metric to measure tracking performance [5], [34].

However, the center location error only measures the pixel difference and does not reflect the size and scale of the target object. For completeness, in the supplementary document, we present the representative precision plots of the trackers averaged over all sequences using the threshold of 20 pixels [5] despite the above-mentioned issues.

Success plot. Another commonly used evaluation metric is the overlap score [21]. Given a tracked bounding box r_t and the ground-truth bounding extent r_0 of a target object, the overlap score is defined as $S = \frac{|\, r_t \bigcap r_0 \,|}{|\, r_t \bigcup r_0 \,|}$, where $\bigcap$ and $\bigcup$ represent the intersection and union operators, respectively, and $|\cdot|$ denotes the number of pixels in a region. The average overlap score (AOS) can be used as the performance measure. In addition, the overlap scores can be used for determining whether an algorithm successfully tracks a target object in one frame, by testing whether S is larger than a certain threshold t_o (e.g., t_o=0.5). As the threshold varies between 0 and 1, the success rate changes and the resultant curve is presented in this work. In addition, the average success rate with a fixed threshold $t_o = 0.5$, is often used for the performance evaluation. Another measure for ranking trackers is the area under curve (AUC) of each success plot, which is the average of the success rates corresponding to the sampled overlap thresholds.

As shown in the supplementary document, with sufficient uniformly sampled thresholds, the AUC score of one sequence is equal to the AOS across the sequence. This can be understood as the average of the frame ratios whose overlap scores are larger than the thresholds is the same as the average of the overlap scores, if there are infinitely many thresholds in the success plot. In the following subsections, we interchangeably use the AOS or AUC score to summarize the performance of the tracking methods on the same sequence.

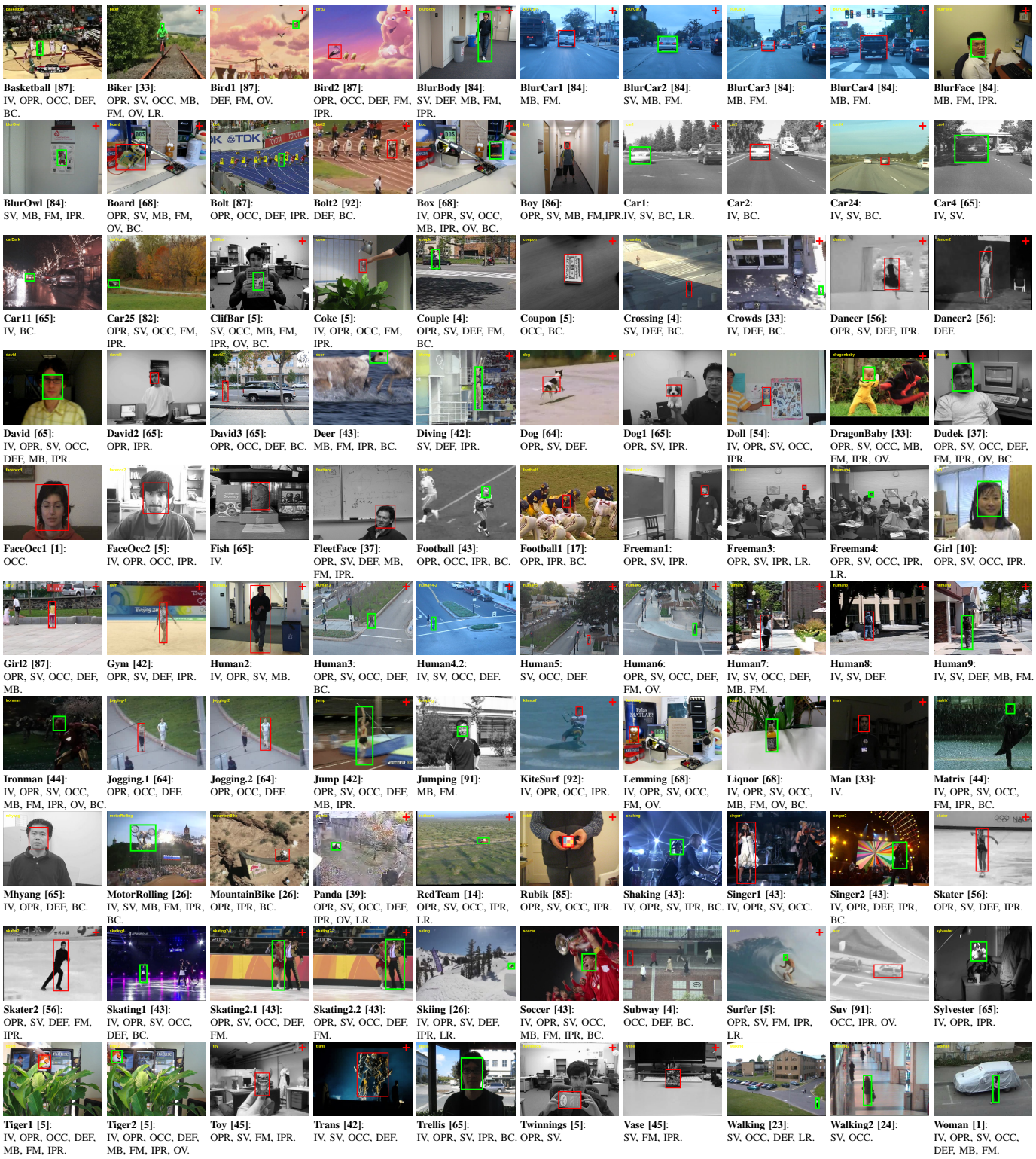


Fig. 1: Annotated image sequences for performance evaluation. The first frame of each sequence is shown with the initial bounding box of the target object. The 50 targets marked with green bounding boxes are selected for extensive evaluations. The newly added sequences compared to [83] are denoted by a red cross at the upper right corner of each image. Some frames are cropped for better illustration.

5.1 Robustness Evaluation

The most common evaluation method is to initialize an algorithm with the ground-truth object state in the first frame and report the average precision or success rate of all the results. This straightforward approach is referred to as a one-pass evaluation (**OPE**). While it is simple, this metric has two major drawbacks. First, a tracking algorithm may be sensitive to initialization in the first frame, and its performance with different initial states or frames may vary significantly. Second, most algorithms do not have re-initialization mechanisms and the tracking results after tracking failures do not provide meaningful information.

We propose two metrics to analyze whether a tracking algorithm is robust to different object states by perturb-

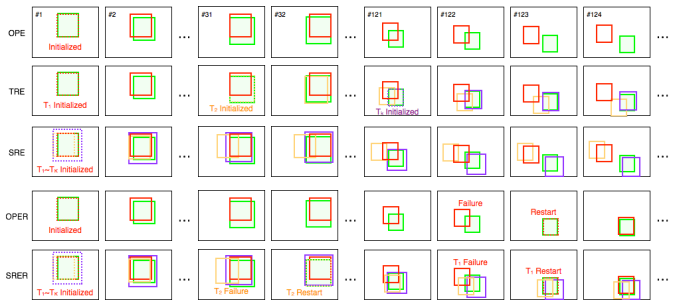


Fig. 2: Evaluation methods for trackers. In each image, the green box represents the ground-truth target location, and the other colors denote the tracker outputs. Dotted boxes represent the initialization of the tracker. OPE is the simplest - initialize the tracker in the first frame and let it track the target until the end of the sequence. In TRE, a tracker starts at different starting frames initialized with the ground-truth bounding box. In SRE, each tracker runs several times with spatially shifted and scaled initializations. The OPER and SRER metrics are used for restarting a tracker for the rest of the sequence if it fails (based on an overlap threshold) at some frame.

ing them temporally (i.e., starting at different frames) or spatially (i.e., starting with different bounding boxes), as illustrated in Figure 2. These evaluation metrics are referred as temporal robustness evaluation (**TRE**) and spatial robustness evaluation (**SRE**) in this work.

Temporal Robustness Evaluation. Each tracking algorithm is evaluated numerous times from different starting frames across an image sequence. In each test, an algorithm is evaluated from a particular starting frame, with the initialization of the corresponding ground-truth object state, until the end of an image sequence. The tracking results of all the tests are averaged to generate the TRE score. Unlike OPE where the earlier part of a sequence is more important since the results from the frames after one tracking failure are not informative, TRE addresses this issue.

Spatial Robustness Evaluation. Accurate initialization of a target object is often important for tracking algorithms, but in practice, it is difficult to achieve this due to errors caused by the detectors or by manual labeling. To evaluate whether a tracking method is sensitive to initialization errors, we generate the object states by slightly shifting or scaling the ground-truth bounding box of a target object. In this work, we use 8 spatial shifts (4 center shifts and 4 corner shifts), and 4 scale variations (See Figure 2). The amount for shift is 10% of the target size, and the scale ratio varies from 80% to 120% of the ground truth at the increment of 10%. The SRE score is the average of these 12 evaluations.

5.2 Robustness Evaluation with Restart

For challenging sequences, a tracking algorithm may fail and lose track of the target when the appearance changes drastically or some distractors appear in the scenes. Once a method fails, it is unlikely to recover and track the target without any external input (e.g., re-detection by an object detector or manual re-initialization). While the TRE score is designed to mitigate this effect, different metrics are necessary to better measure the tracking performance.

One-Pass Evaluation with Restart (OPER). The OPER metric constantly measures how a tracking method performs and re-initializes it at the next frame with the corresponding ground-truth position throughout an image sequence. The average overlap score and the total number of failures show the accuracy and the stability of a tracking algorithm. A tracking failure is defined as when the average overlap score of the frames in a moving window is lower than the given threshold, as illustrated in Figure 2. The averaging window size ω controls the sensitivity to instantaneous failure, and its effect is shown in Figure 6.

Spatial Robustness Evaluation with Restart (SRER). As in the case of OPER, we evaluate whether a tracking method is sensitive to spatial perturbation with restarts such that the tracking performance in challenging sequences can be better analyzed (See Figure 2 for an example).

Approximation using Virtual Runs. Ideally, a tracking method should be restarted at the frame when a failure occurs. However, a few potential issues need to be considered. First, to analyze the behavior of a tracker, we vary the overlap threshold; consequently, the tracking failures occur at different frames. However, it is impractical to evaluate all possible scenarios with different thresholds and parameters (and spatial perturbations in SRER) for every image sequence of the TB-50 or TB-100 benchmark datasets. Second, numerous tracking algorithms are distributed with binary code, and it is not possible to detect a failure and restart a tracker at some particular frames. As such, we use virtual runs to approximate specific parameter settings generated from a set of actual experiments. For each spatial perturbation δ, each tracker is evaluated from every τ-th frame to the end of a sequence with N frames, and the set of results is $\{\mathbf{r}_k^{\boldsymbol{\delta}}\}_{k=0}^{\lfloor N/\tau \rfloor}$, where each run $\mathbf{r}_k^{\boldsymbol{\delta}}$ is a sequence of tracking outputs from frame $(\tau k + 1)$ to N.

A virtual run $\hat{\mathbf{r}}_{\omega,\theta}^{\boldsymbol{\delta}}$ for the perturbation δ, averaging window size ω, and failure threshold θ can be generated from $\{\mathbf{r}_k^{\boldsymbol{\delta}}\}$, as shown in Algorithm 1 in the supplementary file and Figure 3. Basically, when a failure is detected at frame t, there exists a run $\mathbf{r}_{\delta,\lfloor t/\tau \rfloor}$, which is initialized within ω frames from t. Thus, we can approximate the restart by replacing the frames from $t+1$ with the corresponding frames in this run.

In this work, each SRER consists of seven spatial perturbations (1 ground-truth, 4 center shifts, and 2 scale variations: 0.9 and 1.1) and $\omega = 90$ frames. We only evaluate on the TB-50 dataset with SRER due to the large number of experiments: for a sequence of 600 frames, one tracker requires 140 runs (7 variations $\times$ 20 runs), processing 44,100 $(7 \times (600 + 570 + \ldots + 30))$ frames.

6 Evaluation Results

For each tracking algorithm, the default parameters with the source or binary code are used in all evaluations. Table 1 lists the average frame per second (FPS) of each method in OPE on a desktop computer with Intel i7 3770 CPU (3.4 GHz). For a tracking method with a re-detection module (e.g., TLD), no tracking results are returned after

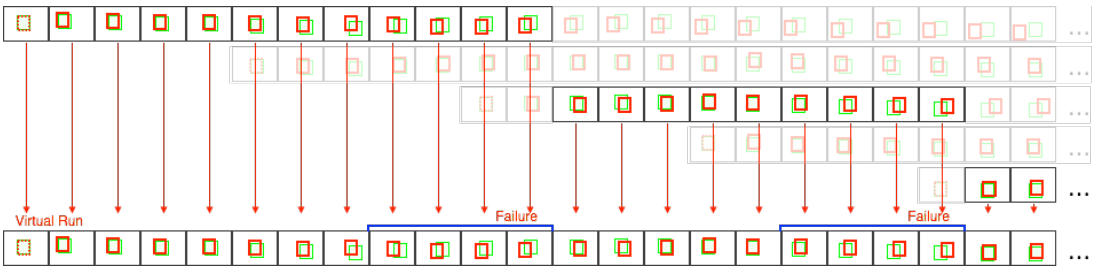


Fig. 3: An OPER virtual run is created from a set of TRE results. From the first frame, it takes parts of the first run in TRE until the average overlap in the test window is less than a given threshold. If a failure is detected, the last run containing the next frame is used, and this process is repeated until the end. It can also be easily extended to SRER virtual runs. Refer to the text and Algorithm 1 in the supplementary material for more details. In the illustrated example, the window size ω is 4 and the temporal initialization sampling interval τ is 5.

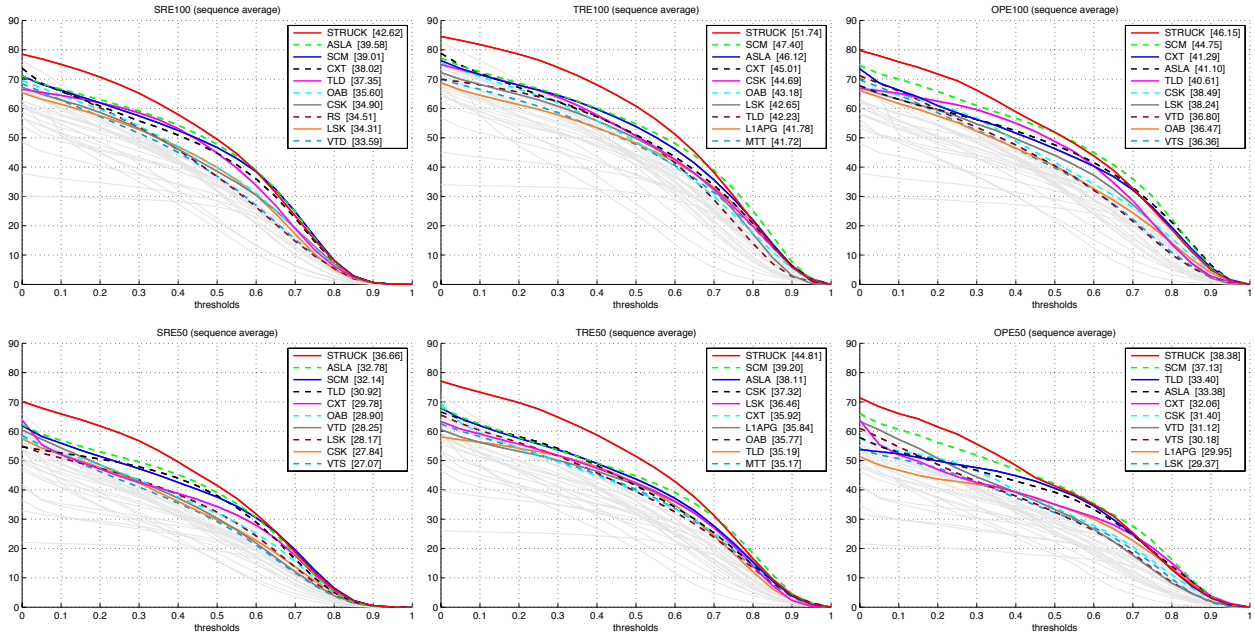


Fig. 4: Plots of OPE, SRE, and TRE on TB-100 (first row) and TB-50 (second row). The score for each tracker is shown in the legend. The top 10 trackers are presented for the sake of clarity, and the rest are shown as gray dashed curves.

the algorithm determines that it loses track of a target object. In such cases, the last tracked position is used for the evaluation. Note that in this benchmark, we annotate each target object in every frame even when it is fully occluded. In the following, we use SRE and SRER as the most representative metrics for the performance evaluation.

6.1 Overall Performance

Each tracker is evaluated on 58,897 frames of the entire TB-100 dataset for OPE. For SRE, each tracking algorithm is evaluated on each sequence with 12 initial object states, where more than 700,000 tracking results per tracker are generated. For TRE, each sequence is partitioned into 20 segments and each method is tested with more than 610,000 frames per tracker. In terms of SRER, more than 80 million tracking results from the TB-50 sequences are generated. To the best of our knowledge, this is the largest-scale performance evaluation of object tracking. We report the most important findings in this manuscript, and further details can be found at http://pami.visual-tracking.net.

The experimental results of OPE, SRE, and TRE on the TB-100 and TB-50 datasets are shown in Figure 4. The scores from the TB-50 dataset are lower than those from the TB-100 set as the TB-50 set consists of more challenging sequences. For the sake of presentation clarity, the plots of the top-10 performing trackers, ordered by the AUC scores, are shown in color (plots of the other trackers are shown in gray).

The average TRE scores from both datasets are higher than those of OPE in that the number of frames decreases from the first to the last segment of TRE. As tracking algorithms tend to perform well in shorter sequences, the average scores of all the results in TRE tend to be higher. On the other hand, the average SRE scores are lower than those of OPE. As a result of imprecise initial appearance models induced in SRE, tracking methods tend to drift away from target objects at a faster pace than those in OPE.

The evaluation results show that OPE is not the best performance indicator as it is one trial of SRE or TRE and does not take the initialization noise into account. The OPE, SRE, and TRE results are mostly consistent in the sense that the top few performing tracking methods according to one criterion also perform well in the other evaluations. However, these tracking methods are effective in different

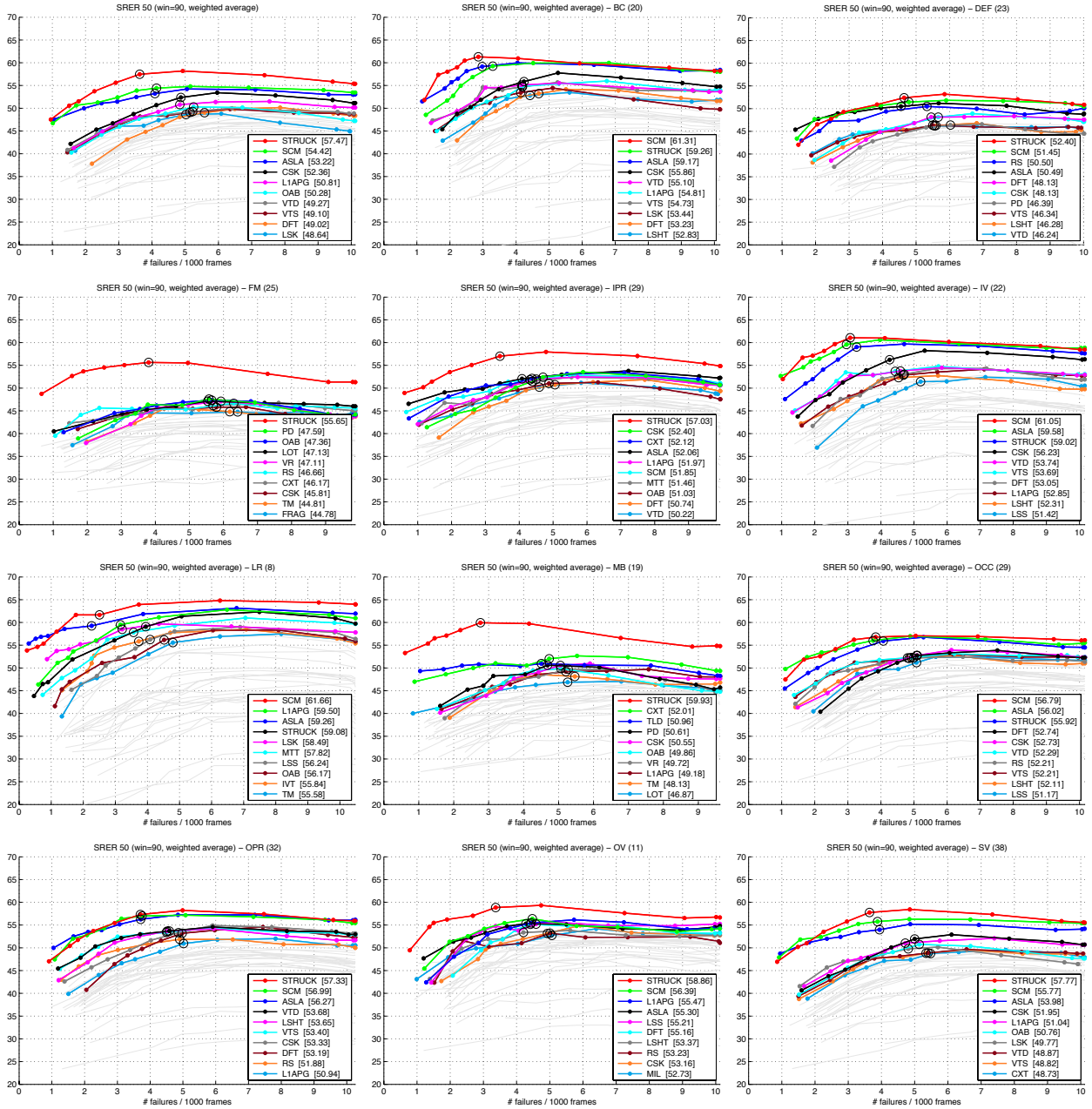


Fig. 5: SRER plots of the overall performance (upper left) and sequences with different attributes. A curve in an SRER plot shows the average overlap scores (y-axis) and the average number of failures (x-axis) for the overlap thresholds (dots on the curve), which are varied from 0 to 1. The black circle represents the score when the overlap threshold is 0.5. The values in the legend are the average overlap scores at the threshold of 0.5.

aspects. In the success plots for evaluations based on the TB-100 dataset, the SCM method in OPE outperforms the ASLA approach (by 3.65%) but is slightly worse in SRE (by 0.57%), which suggests that one of the considered algorithms is more robust to the spatial perturbation of the initial object states. The ranking of the TLD method in TRE is lower than that in OPE and SRE. This can be attributed to the fact that the TLD algorithm performs well in long sequences with a re-detection module while there are numerous short segments in TRE. The success plots of the Struck method in TRE and SRE show that its success rate is higher than those of the SCM and ALSA methods when the overlap threshold is low, but lower when the overlap threshold is higher. This is because the Struck method only estimates the object location but not scale or rotation.

Sparse representations are used in the SCM, ASLA, LSK, MTT, and L1APG methods. These tracking approaches perform well in SRE and TRE, which suggests that sparse representations are effective models to account for appearance changes (e.g., occlusion). We note that the SCM, ASLA, and LSK algorithms outperform the MTT and L1APG methods. The results suggest that local sparse representations are more effective than the ones with holistic sparse templates. The AUC score of the ASLA method decreases slowly as compared to that of the other top 5 trackers in SRE (when compared with OPE), which indicates that the used pooling method is more robust to alignment errors and background clutters.

Among the top 10 trackers, the CSK method achieves the

TABLE 4: SRER evaluation results on the TB-50 dataset. Each entry contains the average overlap in percentage and the average number of failures in 1000 frames at the overlap threshold of 0.5. The trackers are ordered by the average overlap scores, and the top 5 methods in each attribute are denoted by different colors: red, green, blue, cyan, and magenta.

	All	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
STRUCK	**57.5 / 3.6**	**59.3 / 3.3**	**52.4 / 4.6**	**55.6 / 3.8**	**57.0 / 3.4**	**59.0 / 3.3**	**59.1 / 3.9**	**59.9 / 2.8**	**55.9 / 4.1**	**57.3 / 3.7**	**58.9 / 3.4**	**57.8 / 3.6**
SCM	**54.4 / 4.1**	**61.3 / 2.9**	**51.5 / 4.8**	42.8 / 6.5	51.8 / 4.3	**61.1 / 3.1**	**61.7 / 2.5**	45.2 / 5.9	**56.8 / 3.8**	**57.0 / 3.8**	**56.4 / 4.5**	**55.8 / 3.9**
ASLA	**53.2 / 4.1**	**59.2 / 3.0**	**50.5 / 4.5**	42.0 / 6.5	**52.1 / 4.1**	**59.6 / 3.0**	**59.3 / 2.3**	44.6 / 5.9	**56.0 / 3.8**	**56.3 / 3.7**	**55.3 / 4.3**	**54.0 / 3.9**
CSK	**52.4 / 4.9**	**55.9 / 4.2**	48.1 / 5.7	45.8 / 5.8	**52.4 / 4.7**	**56.2 / 4.3**	55.0 / 4.8	**50.6 / 5.1**	**52.7 / 5.1**	53.3 / 4.9	53.2 / 4.9	**52.0 / 5.0**
L1APG	**50.8 / 4.8**	54.8 / 4.2	44.6 / 5.8	44.6 / 6.0	**52.0 / 4.5**	52.9 / 4.7	**59.5 / 3.2**	49.2 / 5.2	50.9 / 4.8	50.9 / 5.0	**55.5 / 4.4**	**51.0 / 4.7**
OAB	50.3 / 5.3	50.1 / 5.1	46.0 / 6.0	**47.4 / 5.7**	51.0 / 4.9	48.3 / 5.8	56.2 / 4.5	49.9 / 5.1	49.2 / 5.6	49.2 / 5.6	50.0 / 5.6	50.8 / 5.1
VTD	49.3 / 5.2	**55.1 / 4.2**	46.2 / 5.5	41.7 / 6.8	50.2 / 4.6	**53.7 / 4.6**	47.1 / 5.6	43.5 / 6.3	52.3 / 4.9	**53.7 / 4.6**	51.5 / 5.4	48.9 / 5.4
VTS	49.1 / 5.1	54.7 / 4.2	46.3 / 5.5	40.6 / 6.8	50.0 / 4.5	53.7 / 4.4	47.1 / 5.6	42.6 / 6.2	52.2 / 4.8	53.4 / 4.5	51.2 / 5.3	48.8 / 5.3
DFT	49.0 / 5.6	53.2 / 4.7	**48.1 / 5.5**	41.7 / 6.3	50.7 / 5.1	53.0 / 4.7	47.7 / 6.3	46.7 / 5.4	**52.7 / 5.1**	53.2 / 5.0	55.2 / 4.4	47.9 / 5.9
LSK	48.6 / 5.0	53.4 / 4.1	45.1 / 5.5	38.9 / 6.7	48.1 / 4.8	50.2 / 4.8	**58.5 / 3.2**	39.8 / 6.5	50.6 / 4.8	50.0 / 4.8	47.5 / 5.5	49.8 / 4.8
RS	48.4 / 5.6	48.6 / 5.1	**50.5 / 5.3**	46.7 / 6.3	47.0 / 5.8	47.6 / 5.4	44.5 / 5.8	46.4 / 6.0	52.2 / 5.0	51.9 / 4.9	53.2 / 5.0	47.9 / 5.7
MTT	48.3 / 5.2	50.7 / 4.9	40.1 / 6.6	40.6 / 6.8	51.5 / 4.4	50.2 / 5.0	57.8 / 3.6	45.5 / 6.2	47.3 / 5.5	48.2 / 5.3	50.4 / 5.6	48.2 / 5.3
LSHT	48.3 / 5.3	52.8 / 4.4	46.3 / 5.6	40.6 / 6.4	48.0 / 4.9	52.3 / 4.5	49.0 / 5.6	42.9 / 5.6	52.1 / 4.9	**53.7 / 4.5**	53.4 / 4.2	48.6 / 5.4
CXT	48.2 / 5.4	49.4 / 5.5	37.0 / 7.2	46.2 / 5.7	**52.1 / 4.4**	48.5 / 5.4	53.7 / 5.0	**52.0 / 4.7**	45.7 / 6.0	47.2 / 5.7	51.0 / 5.3	48.7 / 5.5
LSS	47.6 / 5.6	52.8 / 4.7	42.7 / 6.3	39.5 / 6.6	47.1 / 5.5	51.4 / 5.2	56.2 / 4.1	42.7 / 6.1	51.2 / 5.1	50.2 / 5.4	**55.2 / 4.6**	48.5 / 5.3
TLD	46.8 / 5.5	48.3 / 5.4	37.4 / 7.2	44.6 / 5.5	48.9 / 4.9	46.7 / 5.6	53.3 / 4.9	**51.0 / 4.5**	45.2 / 5.9	46.0 / 5.7	50.2 / 5.0	47.1 / 5.6
TM	46.7 / 6.0	49.4 / 5.4	40.2 / 6.6	44.8 / 6.2	47.2 / 5.7	45.7 / 6.2	55.6 / 4.8	48.1 / 5.5	46.8 / 5.8	46.3 / 6.1	52.6 / 5.1	47.6 / 5.8
PD	46.6 / 5.8	45.1 / 6.1	46.4 / 6.0	**47.6 / 5.6**	48.5 / 5.2	45.1 / 6.0	43.2 / 6.1	**50.6 / 4.7**	49.2 / 5.6	48.9 / 5.6	52.4 / 5.2	46.3 / 5.9
IVT	46.4 / 5.5	51.6 / 4.6	40.5 / 6.4	37.3 / 6.9	46.4 / 5.3	51.2 / 4.8	55.8 / 3.7	41.3 / 6.2	49.3 / 5.1	49.0 / 5.3	52.3 / 4.9	47.1 / 5.3
VR	45.9 / 6.0	44.8 / 6.2	45.3 / 6.2	**47.1 / 5.9**	47.9 / 5.5	43.7 / 6.4	42.4 / 6.0	49.7 / 5.3	48.1 / 5.9	48.2 / 5.9	50.8 / 5.5	45.8 / 6.2
MIL	45.9 / 6.1	48.6 / 5.3	45.7 / 5.9	44.1 / 6.5	45.7 / 5.9	47.1 / 5.6	43.5 / 6.8	43.7 / 6.3	47.6 / 5.8	48.9 / 5.7	52.7 / 5.1	44.5 / 6.5
LOT	45.5 / 5.9	48.6 / 5.2	42.4 / 6.5	**47.1 / 5.5**	43.2 / 6.3	45.4 / 5.8	38.1 / 6.1	46.9 / 5.3	49.5 / 5.2	48.5 / 5.5	49.2 / 4.8	45.6 / 5.8
FOT	44.3 / 6.6	51.6 / 5.3	38.8 / 7.2	40.8 / 6.7	46.1 / 5.8	50.8 / 5.6	42.1 / 8.0	44.9 / 6.2	44.7 / 6.6	47.0 / 6.1	49.0 / 5.5	43.8 / 6.7
FRAG	44.2 / 6.6	46.1 / 5.9	41.8 / 6.8	44.8 / 6.4	43.3 / 6.7	42.6 / 6.6	42.6 / 7.3	46.1 / 6.1	46.6 / 6.2	46.1 / 6.4	50.1 / 5.5	44.2 / 6.8
PCOM	44.0 / 6.0	49.2 / 5.0	39.0 / 6.4	37.6 / 6.8	43.7 / 5.9	47.0 / 5.8	49.4 / 5.3	41.7 / 6.3	47.4 / 5.5	46.3 / 5.9	50.4 / 5.2	44.8 / 5.7
KMS	43.8 / 6.6	44.1 / 6.2	44.8 / 6.4	44.6 / 6.3	42.7 / 6.8	41.5 / 6.8	39.4 / 7.3	43.9 / 6.1	46.6 / 6.2	45.8 / 6.5	45.8 / 6.2	44.0 / 6.6
CPF	43.5 / 6.6	42.8 / 6.4	44.2 / 6.8	42.9 / 6.9	43.1 / 6.4	41.0 / 6.9	43.6 / 5.3	39.8 / 7.1	47.9 / 5.8	47.6 / 5.9	46.4 / 6.0	44.1 / 6.6
ORIA	42.4 / 6.4	47.6 / 5.9	34.8 / 7.8	32.8 / 7.6	44.9 / 5.5	46.9 / 5.9	50.8 / 4.4	36.8 / 7.3	44.4 / 6.3	44.9 / 6.1	46.9 / 6.2	43.4 / 6.3
CT	40.4 / 6.6	43.2 / 5.9	39.4 / 6.8	35.7 / 7.5	42.8 / 6.0	42.5 / 6.2	40.0 / 6.8	37.1 / 7.2	43.7 / 6.4	44.9 / 6.1	47.3 / 5.9	40.2 / 6.8
SBT	37.3 / 7.4	37.3 / 7.4	28.0 / 8.5	36.7 / 7.3	35.5 / 7.5	34.7 / 7.7	48.0 / 6.3	38.3 / 6.8	36.8 / 7.4	35.8 / 7.7	41.0 / 7.2	38.7 / 7.2
MS	35.6 / 7.9	36.7 / 7.4	32.8 / 8.0	40.5 / 7.1	36.8 / 7.8	34.6 / 7.8	28.4 / 9.2	41.2 / 6.8	37.4 / 7.7	37.3 / 7.9	41.0 / 6.9	36.0 / 7.9
BSBT	31.4 / 7.8	30.8 / 7.8	20.6 / 8.7	30.8 / 7.6	31.2 / 7.6	28.9 / 8.0	42.4 / 6.8	32.3 / 7.4	32.0 / 7.7	30.4 / 7.9	36.3 / 7.2	32.5 / 7.7
SMS	29.0 / 8.2	24.2 / 8.4	29.7 / 8.5	31.6 / 7.7	30.4 / 8.1	26.7 / 8.3	31.7 / 8.0	33.0 / 7.2	33.3 / 7.9	31.5 / 8.2	31.0 / 8.2	29.9 / 8.3

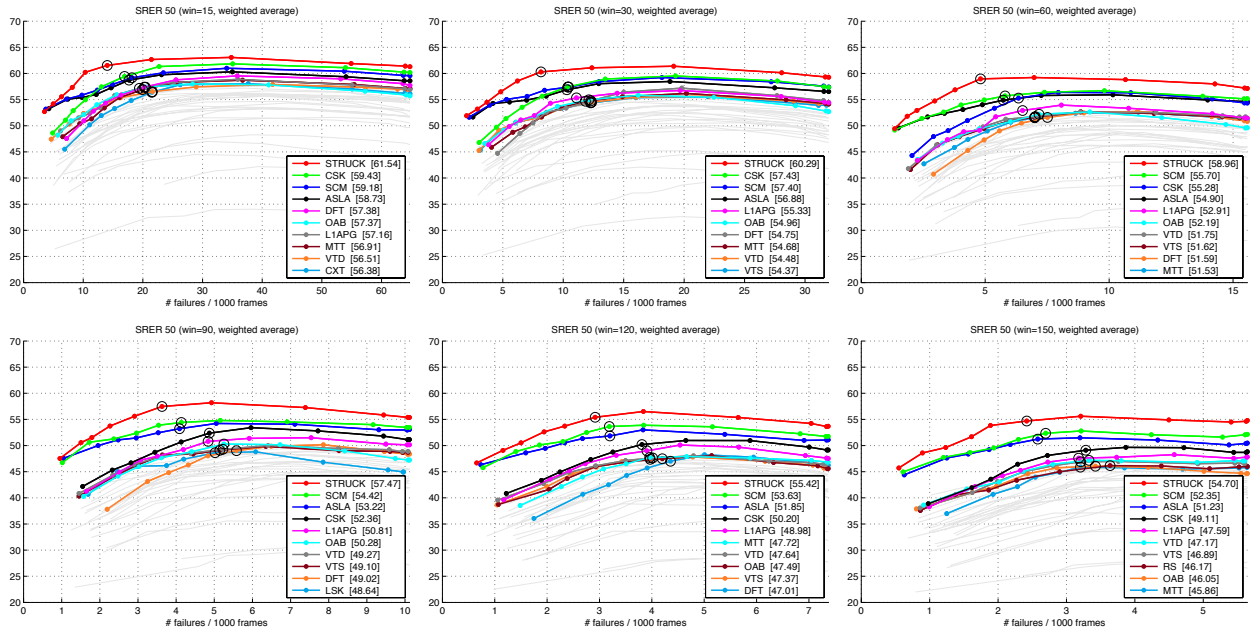


Fig. 6: SRER plots with different averaging window sizes for failure detection. As the window size for failure detection increases, the average overlap scores and the number of failures decrease. However, the trends and ranks of the trackers remain similar.

highest frame rate where the formulation based on a circulant structure plays a key role. The VTD and VTS methods use multiple models to account for appearance changes and fast motion. Compared with the top-performing tracking methods, the performance bottleneck of these two methods may be the representations based on the sparse principal component analysis of holistic templates.

6.2 Performance of SRER

To evaluate how each tracking algorithm performs under spatial and temporal perturbations of initial object states, we carry out experiments using the spatial robustness evaluation with restart (SRER) metric.

In the SRER plot (shown in Figure 5), the y-axis denotes the average success rate of all frames, and the x-axis represents the average number of failures (the moment at

which the tracker restarts) in 1000 frames. Each tracker is evaluated with 11 different overlap thresholds from 0.0 to 1.0, and the success rates and the number of failures are shown as solid dots in the plot. The plots can be interpreted in a way similar to the receiver operator curve (ROC). As the overlap threshold increases, the number of failures increases and the success rate increases at first and then decreases slightly. When the overlap threshold is low, a tracker is not restarted even when it actually loses track of a target object and the model of a tracker is likely to be incorrect. The average success rate increases when the threshold values are low since re-initialization significantly helps the object tracking. However, an SRER plot saturates around the threshold value of 0.5 or 0.6 and decreases in higher thresholds where a tracker is restarted too frequently and fails to learn effective representations to account for appearance changes. Ideal trackers should achieve high success rates and a low number of restarts, which corresponds to the top left corner of the plot. To rank the tracking performance, we use the average success rate at the overlap threshold of 0.5 (denoted by a black circle), and the values are also shown in the legend. The upper left plot of Figure 5 shows the overall performance of the considered trackers on the TB-50 dataset. The top-five tracking methods in SRER perform significantly better than the others in terms of both the average success rate and the average number of failures.

Table 4 shows the average success rates and the average number of failures for all the trackers with respect to all attributes at the overlap threshold of 0.5. The tracking algorithms are sorted by the average success rates, and the top-five methods denoted by different colors. We present the effect of window size that determines the interval for testing the tracking failure in Figure 6. The original runs are generated at every 30 frames from the beginning, and the virtual runs are constructed for the given test windows and thresholds. As the test window size increases, both the average success rate and the number of failures decrease as momentary failures have less effect in tests with longer averaging windows. Overall, the performance of the evaluated trackers is consistent over large variations of the time window, and the window size of 90 is used for all other SRER results.

6.3 Performance Analysis by Attributes

By annotating each sequence, we construct subsets with different dominant attributes that facilitate the analysis of the performance of trackers for each challenging factor. Figure 5 shows the SRER plots of 11 attributes on the TB-50 dataset. These plots show the different performance characteristics of the tracking algorithms.

When an object moves fast, dense sampling based trackers (e.g., Struck) perform much better than the others. This can be attributed to the fact that methods based on optimal prediction or relatively sparse samples do not perform well as the true object state is not included in the search region. However, stochastic search-based trackers with a high overall performance (e.g., SCM and ASLA) do not perform well in this subset due to the simple motion models and small search regions. If the search region is large, more particles need to be drawn. These trackers can be further improved with more effective motion models and particle filters.

On the OCC subset, the Struck, SCM, and ASLA methods outperform the others. These results suggest that structured learning and local sparse representations are effective in dealing with occlusion. On the SV subset, the ASLA, SCM, and Struck methods perform well. The results show that trackers with affine motion models (e.g., ASLA and SCM) often handle scale variation better than the others that are designed to account for only translational motion with a few exceptions such as the Struck method.

6.4 Tracking Speed

The speed of tracking methods is affected by different factors despite the different implementation platforms. These factors include the bounding box size, representation scheme, state space, number of particles, number of features, and number of iterations. For example, the L1 tracking method requires solving one ℓ_1 minimization problem for each drawn particle. The MIL tracking method has a lower frame rate and more robust results when the number of Haar-like features is larger. The ASLA and SCM methods solve several ℓ_1 minimization problems with small image patches. On the other hand, the time for Markov chains to converge in the VTD and VTS algorithms is significant. Table 1 lists the average speed of each method in OPE[7]. More detailed speed statistics, such as the minimum and the maximum values, are available on the above-mentioned web page.

7 Conclusions

In this work, we performed large-scale experiments to evaluate the performance of recent object-tracking algorithms. Based on the benchmark experiments, we highlighted some tracking components that are essential for improving the tracking performance. First, background information is important for effective tracking, which can be exploited in discriminative approaches implicitly (e.g., Struck) or used as the tracking context explicitly (e.g., CXT). Second, local models are effective for object tracking as shown by the performance improvement of methods based on the local sparse representation (e.g., ASLA and SCM) than those based on holistic sparse approaches (e.g., MTT and L1APG). These models are particularly useful when the target appearance changes under partial occlusion or deformation. Third, motion models are crucial for object tracking, particularly when the target movement is large or abrupt. Effective state prediction models significantly reduce the search range and thus improve the tracking efficiency and robustness. It is of great interest to develop

7. The average FPS of the VIVID trackers, MS, TM, RS, and PD is 125, 106, 130, and 109, respectively.

tracking methods to take these factors into account. The evaluation results show that significant progress in the field of object tracking has been made in the last decade. We propose and demonstrate evaluation metrics for an in-depth analysis of object-tracking algorithms from several perspectives.

We note that considerable progress has recently been made [19], [49], [25] to improve the state-of-the-art. In [19], Danelljan *et al.* extended the CSK method [34] by using many color attributes/features and 41 color sequences were selected from the benchmark dataset [83] for the evaluation. On the other hand, the CSK method was extended by using multi-channel features [35]. A hierarchical and compositional and-or graph representation was used for simultaneously tracking, learning, and parsing objects [49]. A Gaussian process regressor was used for estimating the probability of target appearance, and two types of labeled samples were used for improving the tracking performance [25], and some promising results have been demonstrated on the benchmark dataset [83] These trackers will be added to the library and made available on the evaluation website.

In this work, the large-scale performance evaluation facilitated a better understanding of the state-of-the-art object tracking approaches, and provided a platform for gauging new algorithms. Our future work focuses on extending the datasets and code library to include more fully annotated sequences and trackers.

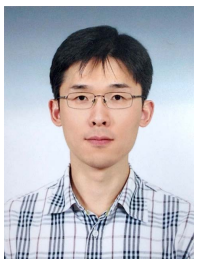
Acknowledgement We thank the reviewers for valuable comments and suggestions. Y. Wu is supported partly by NSFC under Grants 61005027 and 61370036. J. Lim is supported partly by the ICT R&D programs of MSIP/IITP (No. 10047078 and No. 14-824-09-006) and MSIP/NIPA (CITRC program No. NIPA-2014-H0401-14-1001). M.-H. Yang is supported in part by the National Science Foundation CAREER Grant 1149783 and IIS Grant 1152576.

REFERENCES

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust Fragments-based Tracking using the Integral Histogram. In *CVPR*, 2006.
[2] N. Alt, S. Hinterstoisser, and N. Navab. Rapid Selection of Reliable Templates for Visual Tracking. In *CVPR*, 2010.
[3] S. Avidan. Support Vector Tracking. *PAMI*, 26(8):1064–1072, 2004.
[4] S. Avidan. Ensemble Tracking. *PAMI*, 29(2):261–271, 2008.
[5] B. Babenko, M.-H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *PAMI*, 33(7):1619–1632, 2011.
[6] Y. Bai and M. Tang. Robust Tracking via Weakly Supervised Ranking SVM. In *CVPR*, 2012.
[7] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *IJCV*, 56(3):221–255, 2004.
[8] S. Baker, S. Roth, D. Scharstein, M. J. Black, J. Lewis, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. In *ICCV*, 2007.
[9] C. Bao, Y. Wu, H. Ling, and H. Ji. Real Time Robust L1 Tracker Using Accelerated Proximal Gradient Approach. In *CVPR*, 2012.
[10] S. Birchfield. Elliptical Head Tracking Using Intensity Gradients and Color Histograms. In *CVPR*, 1998.
[11] S. T. Birchfield and S. Rangarajan. Spatiograms Versus Histograms for Region-based Tracking. In *CVPR*, 2005.
[12] M. J. Black and A. D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *IJCV*, 26(1):63–84, 1998.
[13] K. Cannons. A Review of Visual Tracking. Technical Report CSE-2008-07, York University, Canada, 2008.
[14] R. Collins, X. Zhou, and S. K. Teh. An Open Source Tracking Testbed and Evaluation Web Site. In *PETS*, 2005.
[15] R. T. Collins. Mean-shift Blob Tracking through Scale Space. In *CVPR*, 2003.
[16] R. T. Collins, Y. Liu, and M. Leordeanu. Online Selection of Discriminative Tracking Features. *PAMI*, 27(10):1631–1643, 2005.
[17] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-Based Object Tracking. *PAMI*, 25(5):564–577, 2003.
[18] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.
[19] M. Danelljan, F. S. Khan, M. Felsberg, and J. V. D. Weijer. Adaptive Color Attributes for Real-Time Visual Tracking. In *CVPR*, 2014.
[20] T. B. Dinh, N. Vo, and G. Medioni. Context Tracker: Exploring Supporters and Distracters in Unconstrained Environments. In *CVPR*, 2011.
[21] M. Everingham, L. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.
[22] J. Fan, Y. Wu, and S. Dai. Discriminative Spatial Attention for Robust Tracking. In *ECCV*, 2010.
[23] J. Ferryman and A. Shahrokni. Pets 2009: Dataset and challenge. In *PETS*, 2009.
[24] R. B. Fisher. The PETS04 Surveillance Ground-Truth Data Sets. In *PETS*, 2004.
[25] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer Learning Based Visual Tracking with Gaussian Processes Regression. In *ECCV*, 2014.
[26] M. Godec, P. M. Roth, and H. Bischof. Hough-based Tracking of Non-Rigid Objects. In *ICCV*, 2011.
[27] H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. In *BMVC*, 2006.
[28] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-Line Boosting for Robust Tracking. In *ECCV*, 2008.
[29] H. Grabner, J. Matas, L. V. Gool, and P. Cattin. Tracking the Invisible: Learning Where the Object Might be. In *CVPR*, 2010.
[30] G. D. Hager and P. N. Belhumeur. Efficient Region Tracking With Parametric Models of Geometry and Illumination. *PAMI*, 20(10):1025–1039, 1998.
[31] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *PAMI*, 30(7):1186–1197, 2008.
[32] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured Output Tracking with Kernels. In *ICCV*, 2011.
[33] S. He, Q. Yang, R. W. H. Lau, J. Wang, and M.-H. Yang. Visual Tracking via Locality Sensitive Histograms. In *CVPR*, 2013.
[34] J. a. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the Circulant Structure of Tracking-by-Detection with Kernels. In *ECCV*, 2012.
[35] J. a. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *PAMI*, 2015.
[36] M. Isard and A. Blake. CONDENSATION–Conditional Density Propagation for Visual Tracking. *IJCV*, 29(1):5–28, 1998.
[37] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust Online Appearance Models for Visual Tracking. *PAMI*, 25(10):1296–1311, 2003.
[38] X. Jia, H. Lu, and M.-H. Yang. Visual Tracking via Adaptive Structural Local Sparse Appearance Model. In *CVPR*, 2012.
[39] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In *CVPR*, 2010.
[40] M. Kristan, R. Pflugfelder, and et.al. The Visual Object Tracking VOT2013 challenge results. In *ICCV Workshops*, 2013.
[41] J. Kwon and K. Lee. Tracking of abrupt motion using Wang-Landau Monte Carlo estimation. In *ECCV*, 2008.
[42] J. Kwon and K. M. Lee. Tracking of a Non-Rigid Object via Patch-based Dynamic Appearance Modeling and Adaptive Basin Hopping Monte Carlo Sampling. In *CVPR*, 2009.
[43] J. Kwon and K. M. Lee. Visual Tracking Decomposition. In *CVPR*, 2010.
[44] J. Kwon and K. M. Lee. Tracking by Sampling Trackers. In *ICCV*, 2011.
[45] J. Kwon, K. M. Lee, and F. C. Park. Visual Tracking via Geometric Particle Filtering on the Affine Group with Optimal Importance Functions. In *CVPR*, 2009.
[46] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Hengel. A Survey of Appearance Models in Visual Object Tracking. *ACM Transactions on Intelligent Systems and Technology*, 4(4):58, 2013.
[47] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Springer, 1993.
[48] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. Robust Tracking using

Local Sparse Appearance Model and K-Selection. In *CVPR*, 2011.
[49] Y. Lu, T. Wu, and Z. Song-Chun. Online Object Tracking, Learning and Parsing with And-Or Graphs. In *CVPR*, 2014.
[50] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with An Application to Stereo Vision. In *IJCAI*, 1981.
[51] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *PAMI*, 26(5):530–49, 2004.
[52] I. Matthews, T. Ishikawa, and S. Baker. The Template Update Problem. *PAMI*, 26(6):810–815, 2004.
[53] X. Mei and H. Ling. Robust Visual Tracking using L1 Minimization. In *ICCV*, 2009.
[54] X. Mei and H. Ling. Robust Visual Tracking and Vehicle Classification via Sparse Representation. *PAMI*, 33(11):2259–2272, 2011.
[55] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum Error Bounded Efficient L1 Tracker with Occlusion Detection. In *CVPR*, 2011.
[56] S. M. S. Nejhum, J. Ho, and M.-H. Yang. Visual Tracking with Histograms and Articulating Blocks. In *CVPR*, 2008.
[57] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns. *PAMI*, 24(7):971–987, 2002.
[58] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan. Locally Orderless Tracking. In *CVPR*, 2012.
[59] Y. Pang and H. Ling. Finding the Best from the Second Bests - Inhibiting Subjective Bias in Evaluation of Visual Tracking Algorithms. In *ICCV*, 2013.
[60] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. In *ECCV*, 2002.
[61] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. The FERET Evaluation Methodology for Face-Recognition Algorithms. *PAMI*, 22(10):1090–1104, 2000.
[62] F. Porikli. Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces. In *CVPR*, 2005.
[63] F. Porikli and O. Tuzel. Object tracking in low-frame-rate video. In *SPIE Image and Video Communications and Processing*, 2005.
[64] F. Porikli, O. Tuzel, and P. Meer. Covariance Tracking using Model Update based on Lie Algebra. In *CVPR*, 2006.
[65] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. *IJCV*, 77(1):125–141, 2008.
[66] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.
[67] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive Appearance Modeling for Video Tracking: Survey and Evaluation. *T-IP*, 21(10):4334–4348, 2012.
[68] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel Robust Online Simple Tracking. In *CVPR*, 2010.
[69] L. Sevilla-Lara and E. Learned-Miller. Distribution Fields for Tracking. In *CVPR*, 2012.
[70] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual Tracking: An Experimental Survey. *PAMI*, 2013, in press.
[71] S. Stalder, H. Grabner, and L. van Gool. Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition. In *ICCV Workshop*, 2009.
[72] B. Stenger, T. Woodley, and R. Cipolla. Learning to Track with Multiple Observers. In *CVPR*, 2009.
[73] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-Tracking Using Semi-Supervised Support Vector Machines. In *CVPR*, 2007.
[74] O. Tuzel, F. Porikli, and P. Meer. Region Covariance: A Fast Descriptor for Detection and Classification. In *ECCV*, 2006.
[75] P. Viola and M. J. Jones. Robust Real-Time Face Detection. *IJCV*, 57(2):137–154, 2004.
[76] T. Vojir and J. Matas. Robustifying the Flock of Trackers. In *Computer Vision Winter Workshop*, 2011.
[77] D. Wang and H. Lu. Visual Tracking via Probability Continuous Outlier Model. In *CVPR*, 2014.
[78] D. Wang, H. Lu, and M.-H. Yang. Least Soft-thresold Squares Tracking. In *CVPR*, 2013.
[79] D. Wang, H. Lu, and M.-H. Yang. Online Object Tracking with Sparse Prototypes. *T-IP*, 22(1):314–325, 2013.
[80] Q. Wang, F. Chen, W. Xu, and M.-H. Yang. An Experimental Comparison of Online Object Tracking Algorithms. In *Proceedings of SPIE: Image and Signal Processing*, 2011.
[81] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. Online Empirical Evaluation of Tracking Algorithms. *PAMI*, 32(8):1443–1458, 2010.
[82] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, and L. Bai. Real-time Probabilistic Covariance Tracking with Efficient Model Update. *T-IP*, 21(5):2824–2837, 2012.
[83] Y. Wu, J. Lim, and M.-H. Yang. Online Object Tracking: A Benchmark. In *CVPR*, 2013.
[84] Y. Wu, H. Ling, J. Yu, F. Li, X. Mei, and E. Cheng. Blurred Target Tracking by Blur-driven Tracker. In *ICCV*, 2011.
[85] Y. Wu, B. Shen, and H. Ling. Online Robust Image Alignment via Iterative Convex Optimization. In *CVPR*, 2012.
[86] Y. Wu, B. Shen, and H. Ling. Visual Tracking via Online Nonnegative Matrix Factorization. *T-CSVT*, 24(3):374–383, 2014.
[87] F. Yang, H. Lu, and M.-H. Yang. Robust Superpixel Tracking. *T-IP*, 23(4):1639–1651, 2014.
[88] M. Yang, Y. Wu, and G. Hua. Context-Aware Visual Tracking. *PAMI*, 31(7):1195–1209, 2008.
[89] A. Yilmaz, O. Javed, and M. Shah. Object Tracking: A Survey. *ACM Computing Surveys*, 38(4):1–45, 2006.
[90] J. H. Yoon, D. Y. Kim, and K.-J. Yoon. Visual Tracking via Adaptive Tracker Selection with Multiple Features. In *ECCV*, 2012.
[91] Q. Yu, T. B. Dinh, and G. Medioni. Online Tracking and Reacquisition Using Co-trained Generative and Discriminative Trackers. In *ECCV*, 2008.
[92] K. Zhang, L. Zhang, and M.-H. Yang. Real-time Compressive Tracking. In *ECCV*, 2012.
[93] S. Zhang, H. Yao, X. Sun, and X. Lu. Sparse coding based visual tracking: Review and experimental comparison. *Pattern Recognition*, 46(7):1772–1788, 2013.
[94] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust Visual Tracking via Multi-task Sparse Learning. In *CVPR*, 2012.
[95] W. Zhong, H. Lu, and M.-H. Yang. Robust Object Tracking via Sparse Collaborative Appearance Model. *T-IP*, 23(5):2356–2368, 2014.

Yi Wu received his PhD degree from Institute of Automation, Chinese Academy of Sciences, China, in Pattern Recognition and Intelligent Systems in 2009. Since Fall 2009, he has been an assistant professor at Nanjing University of Information Science and Technology. From May 2010 to June 2012, he was a postdoctoral fellow at Temple University, USA. From July 2012 to April 2014, he was working as a postdoctoral fellow at University of California, Merced. His research interests include computer vision, multimedia analysis, and machine learning.



Jongwoo Lim graduated from Seoul National University, Seoul, Korea, in 1997, and received his MS degree in 2003 and PhD degree in 2005 from University of Illinois at Urbana-Champaign (UIUC), USA. He worked at Honda Research Institute USA Inc., Mountain View, CA, USA, as a senior scientist from 2005 to 2011, and at Google Inc., Mountain View, CA, USA, as a software engineer from 2011 to 2012. Currently, he is an assistant professor in Division of Computer Science & Engineering at Hanyang University. His research interests include computer vision, robotics, and machine learning.

Ming-Hsuan Yang is an associate professor in Electrical Engineering and Computer Science at University of California, Merced. He received his PhD in Computer Science from the University of Illinois at Urbana-Champaign in 2000. Yang served as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence from 2007 to 2011, and is an associate editor of the International Journal of Computer Vision, Image and Vision Computing and Journal of Artificial Intelligence Research. He received the NSF CAREER award in 2012, the Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He is a senior member of the IEEE and the ACM.