MUlti-Store Tracker (MUSTer): a Cognitive Psychology Inspired Approach to Object Tracking

Zhibin Hong[1], Zhe Chen[1], Chaohui Wang[2], Xue Mei[3], Danil Prokhorov[3], and Dacheng Tao[1]

[1]Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW 2007, Australia
[2]Laboratoire d'Informatique Gaspard Monge, UMR CNRS 8049, Université Paris-Est, 77454 Marne-la-Vallée, France
[3] Toyota Research Institute, North America, Ann Arbor, MI 48105, USA
{zhibin.hong@student., zhe.chen@student., dacheng.tao@}uts.edu.au, chaohui.wang@u-pem.fr, {xue.mei, danil.prokhorov}@tema.toyota.com

Abstract

Variations in the appearance of a tracked object, such as changes in geometry/photometry, camera viewpoint, illumination, or partial occlusion, pose a major challenge to object tracking. Here, we adopt cognitive psychology principles to design a flexible representation that can adapt to changes in object appearance during tracking. Inspired by the well-known Atkinson-Shiffrin Memory Model, we propose MUlti-Store Tracker (MUSTer), a dual-component approach consisting of short- and long-term memory stores to process target appearance memories. A powerful and efficient Integrated Correlation Filter (ICF) is employed in the short-term store for short-term tracking. The integrated long-term component, which is based on keypoint matching-tracking and RANSAC estimation, can interact with the long-term memory and provide additional information for output control. MUSTer was extensively evaluated on the CVPR2013 Online Object Tracking Benchmark (OOTB) and ALOV++ datasets. The experimental results demonstrated the superior performance of MUSTer in comparison with other state-of-art trackers.

1. Introduction

Object tracking is relatively easy for humans. Humans respond quickly to visual information by recognizing temporal consistency and memorizing useful visual features to recover from tracking failures when the target leaves field-of-view. Memory is one of the most powerful, but least well understood, functions of the human brain. With the sophisticated memory system, humans are capable of adapting to complex environments and behaving stably and consistently when facing temporal issues. We hypothesize that the principles of biological memory can be exploited to design solutions to tracking problems, which can be regarded as a time-series motion estimation problem.

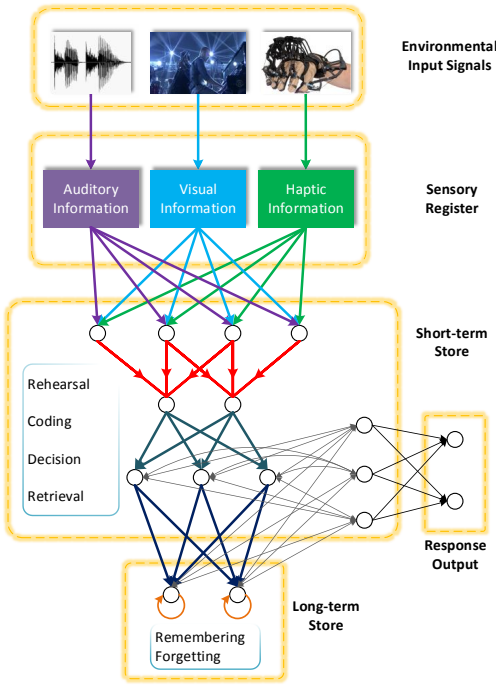

Figure 1. The Atkinson-Shiffrin memory model represented by an illustrative neural network. Nodes and their connections inside the short- and long-term stores represent the possible structure of the neural network inside the human brain.

Although how human memory works is still not fully understood, several memory models have been proposed. We focus on the well-known Atkinson-Shiffrin Memory Model (ASMM, also known as the multi-store model, outlined in Figure 1), which was proposed by Atkinson and Shiffrin in 1968 [1] to explain the basic structure and function of

memory. The ASMM proposes that memory exists as three separate stages: sensory memory, short-term memory, and long-term memory. First, external stimuli are delivered to the "sensory register", at which point the original input signals are transformed into chemical and physical signals for processing within the biological system. Acceptable input information is then sent to the *short-term store*, where information undergoes the processes of encoding, rehearsing, retrieving, and responding, after which the system can output a reasonable and appropriate response. However, the short-term store does not retain information for a long time. Long-term memorizing is actually performed by the *long-term store*; if a particular pattern is received repeatedly, the long-term store is activated and the pattern information is retained. Once memorized, the pattern is maintained for a certain period of time but forgotten if not reinforced. As a result, the information inside the long-term store is a stable and consistent representation of current event sequences.

By combining short-term processing and long-term maintenance, this memory model produces *sensitive* and *stable* responses to complex inputs. When the external environment is continuous and steady, short-term store processing is fast and produces immediate responses. On the other hand, if the model encounters a sudden change in input, information remembered by the long-term store is retrieved, which helps to stabilize the output. This cooperation allows humans to take reasonable actions in response to different and changing environments.

The online tracking research community have developed a number of trackers. Some [14, 16, 21, 28, 39] are highly sensitive and accurate in the short term, while others are relatively conservative but robust over the long term (e.g., [23, 29, 41, 44]). In other words, some trackers can be regarded as short-term systems while others can be regarded as long-term systems. The power of ASMM to track objects by co-operation between the long- and short-term memory stores has motivated us to design a tracker that integrates a short- and long-term system to boost tracking performance.

In this paper, we propose the MUlti-Store Tracker (MUSTer) based on the ASMM. MUSTer consists of one short-term store and one long-term store that collaboratively process the image input and track the target. An Integrated Correlation Filter (ICF), which stores short-term memory and depends on spatiotemporal consistency, is employed in the short-term store to perform short-term tracking via a two-stage filtering. In addition, we integrate a complementary component based on keypoint matching-tracking and RANSAC estimation that can interact with the keypoint feature database in the long-term store and control the final output and the short-term memory states. To maintain a reasonable keypoint feature database size, we adopt the well-known forgetting curve to model the remembering-forgetting loop and, in doing so, retain useful features.

2. Related Work

Online object tracking has long been a popular topic in computer vision. A large number of trackers have been proposed [30, 49], and the recent publication of benchmark datasets containing large numbers of sequences and standardized quantitative evaluation metrics is accelerating the pace of development in this field [24, 43, 48].

Various approaches that form the basis of existing trackers can be used to model the memory of the target appearance. In [42], Ross *et al.* proposed to incrementally learn a low-dimensional subspace of the target representation. Later, Mei *et al.* [37] introduced sparse representations for tracking, subsequently adopted in many trackers [19, 51], in which the memory of the target appearance is modeled using a small set of target instances. In contrast to the generative approaches used in [42] and [37], discriminative methods [2, 3, 4, 14, 17, 20] have been proposed that consider both foreground and background information. In particular, Struck [14] is one of the best performing trackers and has been highlighted in several recent studies [40, 43, 48]. In [14], Hare *et al.* introduced structured SVM for tracking and trained a classifier using samples with structured labels. The correlation filter-based trackers [5, 6, 7, 17, 16, 31] are becoming increasingly popular due to their promising performance and computational efficiency. However, most of these trackers depend on the spatiotemporal consistency of visual cues and adopt relatively risky update schemes; therefore, they can only handle short-term tracking.

Some long-term tracking approaches [8, 23, 29, 41, 44] have also been proposed with promising results. For instance, TLD [23] employs two experts to identify the false negatives and false positives to train a detector. The experts are independent, which ensures mutual compensation of their errors to alleviate "drifting". In [41], Pernici and Bimbo modeled the target appearance using oversampled local features. They used transitive matching to find the target keypoints and, in addition, performed occlusion detection to avoided updating errors. In [44], long-term tracking was conducted based on a self-paced learning scheme in which the target appearance was conservatively learned by selecting trustworthy frames.

Some systems have a similar architecture to ASMM. In 1997, Hochreiter and Schmidhuber introduced a special kind of artificial neural network called *Long Short Term Memory* [18]. In this neural network, memory blocks are made up of two kind of units which are called "constant error carousels" and "multiplicative gate units". The constant error carousels memorize information and the multiplicative gate units control the information flow. These memory blocks can be regarded as the long-term store of ASMM. Together with pure feed-forward networks, which play the short-term role, long short-term memory systems show the great potential for various memory-intensive tasks, such as

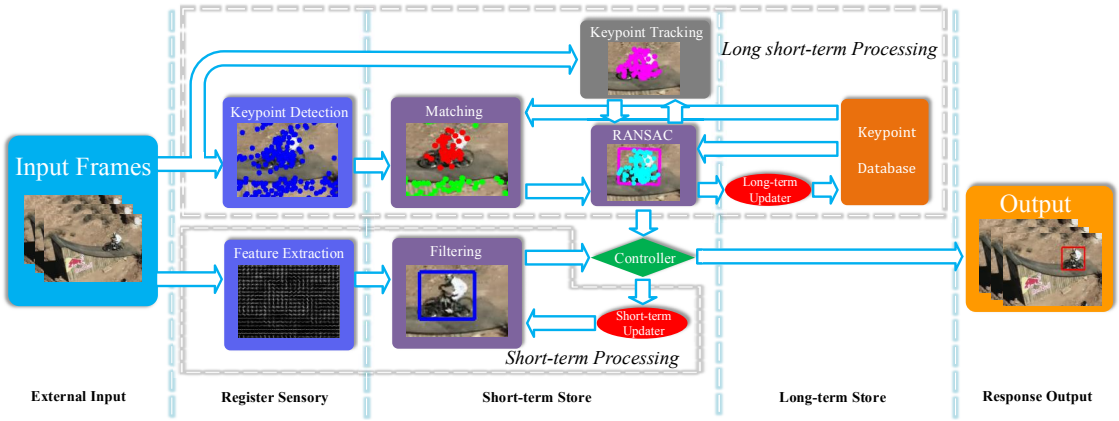

Figure 2. A system flowchart of the proposed tracker based on the Atkinson-Shiffrin Memory Model. The short-term processing in short-term store is conducted by an ICF via two-stage filtering. Another set of short-term procedures including keypoint matching, keypoint tracking and RANSAC estimation is conducted by a conservative long-term component in short-term store, and it is able to interact with the long-term memory located in the long-term store. Both the results of short-term processing and long short-term processing are obtained by a controller, which decides the final output and the ICF update.

speech recognition [13] and music composition [10]. This makes the use of memory models for object tracking more convincing.

3. The Proposed Tracker

The MUSTer framework is based on the ASMM (Figure 2), which consists of a short-term store, a long-term store, and the corresponding processing units. An Integrated Correlation Filter (ICF) is employed in the short-term store to perform short-term processing and track the target based on short-term memory and spatiotemporal consistency. This component generally works accurately and efficiently in relatively stable scenarios. In addition, another relatively conservative long-term component based on keypoint matching-tracking and RANSAC estimation is introduced to conduct the long short-term processing on the fly. This interacts with the short-term memory stored in an active set of keypoints using forward-backward tracking, and it also retrieves the long-term memory for matching and updates the long-term memory based on the RANSAC estimation results and the forgetting curve. During tracking, the outputs of both the short-term and long short-term processing are sent to a controller, which decides the final MUSTer output and the ICF update. Specifically, the short-term memory in ICF is reset when the short-term processing output is highly inconsistent with the long-term memory encoded by the output of the long short-term processing. This enables the recovery of the short-term tracking after dramatic appearance changes such as severe occlusion, the object leaving field-of-view, or rotation.

The following subsections detail successively all components in MUSTer: ICF short-term processing, the short-term processing of keypoints by the long-term component, the updating of long-term memory based on the forgetting curve, and finally, the output controller and ICF updater.

3.1. Short-term Integrated Correlation Filters

The short-term component is used to provide instant responses to the image input based on short-term memory. Recently, the robustness of correlation filter-based short-term trackers [6, 16, 31] has been recognized by [24]. For accurate and efficient short-term processing performance, we employ Integrated Correlation Filters (ICFs), which are based on the Kernelized Correlation Filters (KCFs) [16] and the Discriminative Scale Space Correlation Filter (DSSCF) [6]. The ICF is a two-stage filtering process that performs translation estimation and scale estimation, respectively, which is similar to the pipeline described in [6] and [31]. Here, ICF serves as the short-term component, where the short-term memory of ICF consists of the learned coefficients and templates for the filters.

In KCF, a classifier $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$ is trained on a $M \times N$ image patch $\mathbf{x}$ centered by the target bounding box $\mathbf{B}_\mathrm{T}$'s center and p times larger than $\mathbf{B}_\mathrm{T}$. Instead of using dense sliding windows to extract training samples, the classifier considers all the cyclic shift versions $\mathbf{x}_i$ for training, where $i \in \{0, ..., M-1\} \times \{0, ..., N-1\}$. Each example $\mathbf{x}_i$ is assigned with a score in $y_i \in [0, 1]$ generated by a Gaussian function in terms of the shifted distance, and the classifier is trained by minimizing the regression error:

$$\min_{\mathbf{w}} \sum_i (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - y_i)^2 + \lambda \|\mathbf{w}\|^2, \qquad (1)$$

where $\phi(\mathbf{x})$ is the mapping to a Hilbert space, and $\lambda \geq 0$ is the regularization parameter controlling the model sim-

plicity. Employing a kernel $\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, the classifier can be derived as $f(\mathbf{x}) = \sum_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$, where $\boldsymbol{\alpha}$ is the dual variable of $\mathbf{w}$. Let us denote the Discrete Fourier Transform (DFT) of a vector with a hat "$\hat{}$", e.g., $\hat{\boldsymbol{\alpha}} = \mathscr{F}(\boldsymbol{\alpha})$, and denote the complex conjugate with $\hat{\boldsymbol{\alpha}}^*$. According to [7] and [16], if the employed kernel is shift invariant, e.g., an RBF kernel, $\hat{\boldsymbol{\alpha}}^*$ can be obtained based on the favorable properties of circulant matrices:

$$\hat{\boldsymbol{\alpha}} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}, \tag{2}$$

where $\mathbf{k}^{\mathbf{xx}}$ is a vector whose ith element is $\kappa(\mathbf{x}_i, \mathbf{x})$. In particluar, for image data with C feature channels, a concatenation $\mathbf{x} = [\mathbf{x}^1; ...; \mathbf{x}^C]$ can be constructed, and the kernel correlation $\mathbf{k}^{\mathbf{xx}}$ based on a Gaussian kernel can be efficiently computed by element-wise products and simple summation over the feature channels in the Fourier domain:

$$\mathbf{k}^{\mathbf{xx}'} = \exp(-\frac{1}{\sigma^2}(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathscr{F}^{-1}(\sum_{c=1}^{C} \hat{\mathbf{x}^c} \odot (\hat{\mathbf{x}'^c})^*))), \tag{3}$$

where $\odot$ denotes the operator of element-wise products, and c is the index of the feature channels.

During the first-stage filtering for translation estimation, given a $M \times N$ candidate image patch $\mathbf{z}$ as the search space, all cyclic patches of $\mathbf{z}$ can be evaluated via

$$\mathbf{f}(\mathbf{z}) = \mathscr{F}^{-1}((\hat{\mathbf{k}}^{\mathbf{xz}}) \odot \hat{\boldsymbol{\alpha}}), \tag{4}$$

where $\mathbf{f}(\mathbf{z})$ is the filtering response for all the cyclic versions of $\mathbf{z}$, and the translation is estimated by finding the location with the highest response. In our tracker, the candidate image patch $\mathbf{z}$ is centered by using the tracking result $\mathbf{B}_o$ of the last frame. To adapt to the short-term appearance changes of the target, the filter coefficients $\boldsymbol{\alpha}$ and the target template $\mathbf{x}$ are updated in an interpolating manner with learning rate γ. In our implementation, we employ the 31-dimensional HOG descriptors [11], as in [16]. Moreover, for color image sequences, we further extract 10-dimensional color attributes [7, 46] as complementary features and combine them with the HOG descriptors to further boost performance.

To cope with scale changes, a one-dimensional DSSCF [6] is also trained and the second-stage filtering is performed for scale estimation. To evaluate the trained DSSCF, S image patches centered around the location found by the KCF filter are cropped from the image, each of size $a^s M_\mathrm{T} \times a^s N_\mathrm{T}$, where $M_\mathrm{T} \times N_\mathrm{T}$ is the current size of the target, a is the scale factor, and $s \in \{-\frac{S-1}{2}, ..., \frac{S-1}{2}\}$. All S image patches are then resized to the template size for the feature extraction. Finally, the final output $\mathbf{B}_\mathrm{s}$ from the short-term processing is given as the image patch with the highest filtering response. Similar to KCF, the model parameters are also updated in an interpolating manner with learning rate μ. We refer readers to [6] for more details and the implementation of DSSCF.

3.2. Short-term Processing of Keypoints

The goal of the long-term component is to conservatively learn the appearance of the target and to refresh the short-term memory when a mistake made by the short-term component is detected.

Local Scale-Invariant Features [33] are a powerful tool used in various computer vision tasks including recognition [33] and scene alignment [32]. In particular, some promising trackers [12, 38, 41] have been proposed to model the object appearance based on Local Scale-Invariant Features (i.e., keypoints). Therefore, we employ a long-term component based on keypoint matching-tracking and RANSAC estimation to take advantage of the flexibity and shape generalizability of the keypoint-based appearance models.

The long-term memory of the target appearance is modeled by a total feature database $\mathcal{M} = \mathcal{T} \cup \mathcal{B}$ that consists of a foreground (target) feature database $\mathcal{T}$ and a background feature database $\mathcal{B}$:

$$\mathcal{T} = \{(\mathbf{d}_i, \mathbf{p}_i^o)\}_{i=1}^{N_\mathcal{T}}, \;\; \mathcal{B} = \{\mathbf{d}_i\}_{i=1}^{N_\mathcal{B}} . \tag{5}$$

Here, $\mathbf{d}_i \in \mathcal{R}^{128}$ is the 128-dimensional Scale-invariant Feature Transform (SIFT) descriptors [34] of the keypoints. $N_\mathcal{T}$ and $N_\mathcal{B}$ are the respective numbers of descriptors. Each target descriptor $\mathbf{d}_i \in \mathcal{T}$ is also associated with the corresponding coordinates $\mathbf{p}_i^o \in \mathcal{R}^2$ that remember the keypoint location in the original target template, and can be used for estimating the transformation of target state. The background feature database is important to reduce erroneous matching of target keypoints and can help to detect the occlusions.

In addition to the filtering function of ICF, another set of short-term procedures conducted in the short-term store is to consecutively process the keypoints by retrieving the long-term memory stored in $\mathcal{M}$ and the short-term memory stored in an active set. In each frame, a SIFT detector based on the difference of Gaussians [34] is applied to an image search area to extract a set of keypoints with large responses. The set of detected keypoints associated with their SIFT descriptors is denoted as $\mathcal{P}_\mathrm{D} = \{(\mathbf{d}_k, \mathbf{p}_k)\}_{k=1}^{N_\mathrm{D}}$, where $\mathbf{p}_k \in \mathcal{R}^2$ is the coordinates of a keypoint.

Matching Keypoints. We search the total memory database $\mathcal{M}$ for the nearest neighbors of each $\mathbf{d}_k \in \mathcal{P}_\mathrm{D}$ based on the Euclidean distance. We define the matching confidence of $\mathbf{d}_k$ and its nearest neighbor[1] $\mathbf{d}_k^{1\mathcal{N}}$ as the cosine similarity $C(\mathbf{d}_k, \mathbf{d}_k^{1\mathcal{N}})$ between two descriptors. The candidate matching keypoint can be found if the matching confidence is larger than a predefined threshold, denoted as

[1] For simplicity, we denote the nearest and the second nearest neighbors of $\mathbf{d}_k$ in $\mathcal{M}$ by $\mathbf{d}_k^{1\mathcal{N}}$ and $\mathbf{d}_k^{2\mathcal{N}}$, respectively.

θ_{T} for the matching of target points and θ_{B} for that of background points. The different threshold settings are used to control the different recalls for the matching of foreground and background keypoints. In practice, the precision of the matching of target points is more important since they are used to estimate the current state of the target. Therefore, it is important to reject outliers during target point matching as discussed in [34, 41]. To further reject outliers, we employ the distinct non-parametric nearest neighbor classifier [34] by computing the ratio of the distances:

$$r(\mathbf{d}_k) = \frac{d(\mathbf{d}_k, \mathbf{d}_k^{1\mathcal{N}})}{d(\mathbf{d}_k, \mathbf{d}_k^{2\mathcal{N}})}, \tag{6}$$

where $\mathbf{d}_k^{2\mathcal{N}}$ is the second nearest neighbor of $\mathbf{d}_k$. The matched point $\mathbf{d}_k^{1\mathcal{N}}$ is classified as an inlier if $r(\mathbf{d}_k)$ is smaller than a threshold θ_r. Finally, the detected feature points $(\mathbf{d}_k, \mathbf{p}_k) \in \mathcal{P}_{\mathrm{D}}$ are classified into one of the three following sets: the matched target keypoints $\mathcal{P}_{\mathrm{T}}$, the matched background keypoints $\mathcal{P}_{\mathrm{B}}$ and unmatched keypoints $\mathcal{P}_{\mathrm{N}}$, according to the formula below:

$$\begin{cases} \mathbf{p}_k \in \mathcal{P}_{\mathrm{T}}, & \mathbf{d}_k^{1\mathcal{N}} \in \mathcal{T}, C(\mathbf{d}_k, \mathbf{d}_k^{1\mathcal{N}}) > \theta_{\mathrm{T}}, r(\mathbf{d}_k) < \theta_r \\ \mathbf{p}_k \in \mathcal{P}_{\mathrm{B}}, & \mathbf{d}_k^{1\mathcal{N}} \in \mathcal{B}, C(\mathbf{d}_k, \mathbf{d}_k^{1\mathcal{N}}) > \theta_{\mathrm{B}} \\ \mathbf{p}_k \in \mathcal{P}_{\mathrm{N}}, & \text{otherwise .} \end{cases} \tag{7}$$

Once the matched target keypoints $(\mathbf{d}_k, \mathbf{p}_k) \in \mathcal{P}_{\mathrm{T}}$ are determined, we further find their corresponding coordinates $\mathbf{p}_k^o$ in the original template, which are then added into $\mathcal{P}_{\mathrm{T}}$ so as to get the final complete version, *i.e.*, $\mathcal{P}_{\mathrm{T}} = \{(\mathbf{d}_k, \mathbf{p}_k^o, \mathbf{p}_k)\}_{k=1}^{N_m}$.

Forward-Backward Tracking. In addition to matching keypoints and inspired by [38], we also maintain an active set of keypoints $\mathcal{P}_{\mathrm{A}}^{t-1} = \{(\mathbf{p}_i^o, \mathbf{p}_i^{t-1})\}_{i=1}^{N_A}$, where $\mathbf{p}_i^{t-1}$ is the coordinates of the point in the $t-1$ frame and $\mathbf{p}_i^o$ is the corresponding coordinates of $\mathbf{p}_i^{t-1}$ in the original template. This active set can be also regarded as the short-term memory of MUSTer, and it provides additional information for long short-term processing. To obtain the coordinates of $\mathbf{p}_i^{t-1}$ in frame I_t, the optical flow can be computed using the Lucas-Kanade (LK) method [35]. To improve robustness, we employ a Forward-Backward (F-B) tracker [23] to obtain a set of keypoints with reliable tracking results. In the FB tracker, the forward optical flow from $\mathbf{p}_i^{t-1}$ to $\mathbf{p}_i^t$ and the backward optical flow from $\mathbf{p}_i^t$ to $\mathbf{p'}_i^{t-1}$ are computed using two consecutive frames: I_{t-1} and I_t. The displacement $d(\mathbf{p}_i^{t-1}, \mathbf{p'}_i^{t-1})$ between $\mathbf{p}_i^{t-1}$ and $\mathbf{p'}_i^{t-1}$ is then used to identify any tracking failures. Ideally, $d(\mathbf{p}_i^{t-1}, \mathbf{p'}_i^{t-1})$ should be small if the tracking is successful. Therefore, a threshold θ_{fb} is defined and a failure is detected if $d(\mathbf{p}_i^{t-1}, \mathbf{p'}_i^{t-1}) > \theta_{fb}$. Finally, we can obtain a set of remaining active keypoints $\mathcal{P}_{\mathrm{A}}^t = \{(\mathbf{p}_i^o, \mathbf{p}_i^t)\}_{i=1}^{N'_A}$ that contain the keypoints successfully tracked by the FB tracker.

RANSAC Estimation. Once the matched and tracked keypoints are obtained, a candidate set $\mathcal{P}_{\mathrm{C}}$ is formed consisting of the keypoints in $\mathcal{P}_{\mathrm{T}}$ and $\mathcal{P}_{\mathrm{A}}^t$, which is used to estimate the target state. Similar to [38, 41], we only consider the similarity transformation, which is more reliable than homography transformation for tracking generic objects, especially in cases where the planarity assumption does not hold [38]. In the case of the similarity transformation, the state of the target can be defined as $\mathbf{s}_t = \{x_t, y_t, s_t, \beta_t\}$, which are the parameters of translations, scale, and rotation angle, respectively. To predict the target state $\mathbf{s}_t$, a transformation $F_{\mathbf{s}_t}$ letting $F_{\mathbf{s}_t}(\mathbf{p}_i^o) \rightarrow \mathbf{p}_i$ can be estimated using $\mathcal{P}_{\mathrm{C}} = \{(\mathbf{p}_i^o, \mathbf{p}_i)\}_{i=1}^{N_C}$. However, mistakes in matching and tracking keypoints cannot always be avoided, especially when the background is cluttered or rapidly changing. Therefore, we employ the RANSAC estimator to compute $F_{\mathbf{s}_t}$ by randomly proposing putative solutions and identifying inliers and outliers. In our implementation, we use the robust MLESAC estimator [15, 45], which works by maximizing the likelihood rather than just the number of inliers.

Confidence of Matching. Once the resulting transformation $F_{\mathbf{s}_t}$ is estimated, a target bounding box $\mathbf{B}_l$ defined by the current state can be computed. Meanwhile, a set of inlier keypoints $\mathcal{P}_{\mathrm{I}} = \{\mathbf{p}_i\}_{i=1}^{N_I}$ can also be obtained, where the number of inliers N_I is an important evidence predicting tracking success; in general, the more inliers included in $F_{\mathbf{s}_t}$, the more confident the result. Therefore, we can define a binary variable G_C that indicates the tracking success and set it as:

$$G_C = \begin{cases} \text{True}, & N_I > \theta_I \\ \text{False}, & \text{otherwise} , \end{cases} \tag{8}$$

where θ_I is a predefined threshold controlling recall strictness.

Occlusion Handling. In the proposed framework, the long-term memory is progressively updated on the fly. Therefore, it is important to consider cases of occlusion, which have been discussed previously [25, 41]. In particular, we consider the set $\mathcal{P}(\mathbf{B}_l)$, which consists of the keypoints inside target bounding box $\mathbf{B}_l$, and the set $\mathcal{P}_{\mathrm{O}}$ of *occluding keypoints* is defined as the matched background keypoints inside $\mathbf{B}_l$, *i.e.*, $\mathcal{P}_{\mathrm{O}} = \mathcal{P}(\mathbf{B}_l) \cap \mathcal{P}_{\mathrm{B}}$. Intuitively, if there is no occlusion, the number of occluding points $N_O = |\mathcal{P}_{\mathrm{O}}|$ should be small and close to zero. In contrast, when the target is occluded, the number of occluding keypoints, $N_G = |\mathcal{P}_{\mathrm{G}}|$, is likely to be high and close to the number of keypoints belonging to the target. Therefore, we can consider the ratio of N_O and N_G and define a binary variable G_O that indicates occlusion occurrence:

$$G_O = \begin{cases} \text{True}, & N_O / N_G > \theta_o \\ \text{False}, & \text{otherwise} , \end{cases} \tag{9}$$

where $\mathcal{P}_{\mathrm{G}} = \mathcal{P}(\mathbf{B}_l) \cap \mathcal{P}_{\mathrm{T}}$ is the matched target keypoints

inside the region of $\mathbf{B}_l$. In our implementation, we empirically set $\theta_o = 0.5$.

Update the Active Set. The active set $\mathcal{P}_{\mathrm{A}}^t$ that stores the short-term memory is updated at each frame. However, we should not track the keypoints if an occlusion is detected since the keypoints may gradually lock onto occluding objects or the background. Therefore, we set $\mathcal{P}_{\mathrm{A}}^t = \varnothing$ if $G_O =$ True. Otherwise, we set $\mathcal{P}_{\mathrm{A}}^t = \mathcal{P}_{\mathrm{I}}$, which are the inliers found by the RANSAC estimator. However, when the target is stable or moving slowly, most of the keypoints in $\mathcal{P}_{\mathrm{A}}^t$ would be successfully tracked and identified as inliers, and meanwhile matched target keypoints would continually be added to $\mathcal{P}_{\mathrm{A}}^t$. This might lead to a very large $\mathcal{P}_{\mathrm{A}}^t$ and thus computational inefficiency. Therefore, we design a strategy to find the redundant points in $\mathcal{P}_{\mathrm{A}}^t$ and let the candidate set of RANSAC $\mathcal{P}_{\mathrm{C}} = \mathcal{P}_{\mathrm{T}} \cup (\mathcal{P}_{\mathrm{A}}^t \setminus \mathcal{P}_{\mathrm{R}})$, where $\mathcal{P}_{\mathrm{R}}$ denotes the set of redundant keypoints. This approach is based on the assumption that the matched keypoints are more reliable and, therefore, have higher priority. The redundant points are found using the quantization IDs. Specifically, a virtual grid is built upon the original target template, and each target keypoint can be assigned a quantization ID according to its corresponding coordinates $\mathbf{p}_i^o$ in the original template. Finally, the redundant points in $\mathcal{P}_{\mathrm{A}}$ are found by searching the repetitive quantization IDs in $\mathcal{P}_{\mathrm{T}}$.

3.3. Long-term Memory Updates

The use of keypoints as the appearance model allows natural handling of in-plane rotation. For instance, the keypoint database is not updated in [38] but performance is still reasonable. However, it is still crucial to update on the fly to generalize the appearance model to handle out-of-plane rotation, severe scale changes, and appearance variations of the target [41].

To maintain relatively reliable memory of the target appearance, the memory database $\mathcal{M}$ is updated conservatively only when the short-term processing is confident about the result ($G_C =$ True) and claims there is no occlusion ($G_O =$ False). Both the target keypoint database $\mathcal{T}$ and the background keypoint database $\mathcal{B}$ need to be updated. In particular, we consider the unmatched points that are important for capturing any changes in visual structure. During the update, we add the unmatched keypoints inside the $\mathbf{B}_l$ to $\mathcal{T}$, and those outside $\mathbf{B}_l$ to $\mathcal{B}$, as follows:

$$\begin{aligned} \mathcal{T} &= \mathcal{T} \cup (\mathcal{P}_{\mathrm{N}} \cap \mathcal{P}(\mathbf{B}_l)) \\ \mathcal{B} &= \mathcal{B} \cup (\mathcal{P}_{\mathrm{N}} \cap (\mathcal{P}_{\mathrm{D}} \setminus \mathcal{P}(\mathbf{B}_l)) \ . \end{aligned} \tag{10}$$

While the human brain is good at remembering, as well as forgetting. The remembering-forgetting interplay helps us effectively manage the valuable and finite memory capacity when handling massive quantities of input signals each day. To avoid the unbounded growth of the memory database $\mathcal{M}$, a certain capacity should be set for $\mathcal{T}$ and $\mathcal{B}$, and features should be forgotten over time. To model remembering-forgetting, we employ the famous *forgetting curve* [9] to maintain $\mathcal{M}$ and forget unimportant features according to the retention of features. The forgetting curve hypothesizes a decline in memory retention and shows how information is lost over time when there is no attempt to retain it. The memory retention r over time can typically be modeled using an exponential function:

$$r = \exp(-\frac{\tau}{\Gamma h}) \ , \tag{11}$$

where h is the relative strength of the memory, τ is the relative period of forgetting, and Γ is a constant controlling the scale of the timespan. In (11), the speed of decline in memory retention is decided by the relative strength h. For information with high relative strength h, the decline in memory retention becomes slow and it is more possible to be remembered over time. The relative strength of information can be increased using certain methods such as repetitive retrieving. In MUSTer, each feature in $\mathbf{d}_i \in \mathcal{M}$ is assigned a set of memory variables (r_i, τ_i, h_i), where the memory retention r_i of features can be updated according to their corresponding τ_i and h_i. To model the process of forgetting, during each update term, all relative periods τ_i of $\mathbf{d}_i \in \mathcal{M}$ are increased by 1. Moreover, for the retrieved background features $\mathbf{d}_k^{1\mathcal{N}} \in \mathcal{B}$ where $\mathbf{d}_k \in \mathcal{P}_{\mathrm{B}}$, and foreground features $\mathbf{d}_k^{1\mathcal{N}} \in \mathcal{T}$ where $\mathbf{d}_k \in (\mathcal{P}_{\mathrm{T}} \cap \mathcal{P}_{\mathrm{I}})$, the corresponding relative strength h_i is increased by 1 and the relative period τ_i is set to 0. In this way, the recalled features are renewed and strengthened in memory, while frequently recalled features obtain a high relative strength and become hard to forget. Once the numbers of features in the respective databases exceed a predefined memory capacity, $N_{\mathcal{T}} > \Theta_{\mathcal{T}}$ or $N_{\mathcal{B}} > \Theta_{\mathcal{B}}$, the features with low retention are removed and completely forgotten by the model.

3.4. Output and Short-term Memory Refreshing

Once the different processing procedures are performed in the short-term store, the results of filtering $\mathbf{B}_s$ and the result of the long short-term processing $\mathbf{B}_l$ together with the state variables G_C and G_O can be obtained by a controller. As mentioned earlier, short-term filtering is generally accurate in relatively stable scenarios. Therefore, if the results $\mathbf{B}_s$ and $\mathbf{B}_l$ are to some extent consistent (indicating that the long-term memory agrees with the output of the short-term tracker), or the long-term component is not confident about the result ($G_C =$ False), or an occlusion is detected ($G_C =$ True), the tracking result $\mathbf{B}_o$ is output as $\mathbf{B}_o = \mathbf{B}_s$, and the short-term filters are updated using $\mathbf{B}_o$ and the predefined learning rates $\gamma = \gamma_o$ and $\mu = \mu_o$. Otherwise, the tracking result is output as $\mathbf{B}_o = R(\mathbf{B}_l)$, where $R(\cdot)$ is a function to rotate a bounding box along its center and in the same orientation as $\mathbf{B}_s$. Moreover, the short-term memory

should be refreshed. We use $\mathbf{B}_o$ to update the short-term filters and set the learning rates γ and μ as 1 for the update, which cleans up all the previous short-term memory stored in the filters. In particular, the inconsistency of $\mathbf{B}_s$ and $\mathbf{B}_l$ is detected by the Intersection Over Union (IOU) metric $U(\mathbf{B}_s, \mathbf{B}_l) = \frac{|\mathbf{B}_s \cap \mathbf{B}_l|}{|\mathbf{B}_s \cup \mathbf{B}_l|}$ with a threshold θ_U.

4. Experiments

The proposed tracker[2] was implemented using Matlab & C++ with OpenCV library[3]. The average time cost on OOTB is 0.287s/frame on a cluster node (3.4GHz, 8 cores, 32GB RAM). The parameters mentioned in Section 3 are specified as follows: $\theta_U = 0$, the learning rate γ_o and μ_o for the ICF are set to 0.02 and 0.01 respectively, and the padding size p in KCF is set to 1. The thresholds for matching keypoints are set as $\theta_T = 0.8$, $\theta_B = 0.9$, $\theta_r = 0.85$, and $\theta_I = 8$. The threshold θ_{fb} for FB Tracker is set to 4. The Γ in the forgetting curve model is set to 10, while the memory capacities $\Theta_{\mathcal{T}}$ and $\Theta_{\mathcal{B}}$ of keypoint databases are both set to 2000. Note that the protocols proposed in [48] were strictly followed and all parameters were fixed for all video sequences in the following evaluations. For each sequences, we simply use the ground truth of the first frame given by the dataset for initialization.

4.1. Evaluation on CVPR2013 OOTB

In this section, we report the evalution on the CVPR2013 Online Object Tracking Benchmark (OOTB) [48] and compare MUSTer with a number of state-of-the-art trackers. OOTB is a popular comprehensive benchmark specifically designed for evaluating performance. OOTB contains 50 fully annotated sequences which extensively used by previous work. In [48], the evaluation for the robustness of trackers is based on two different metrics: the precision plot and success plot. The precision plot shows the percentage of successfully tracked frames on which the Center Location Error (CLE) of a tracker is within a given threshold T_C, and a representative precision score at $T_C = 20$ is used for ranking the trackers. The success plot also counts the percentage of successfully tracked frames, by measuring the Intersection Over Union (IOU) metrics for trackers on each frame. In success plot, the threshold of IOU is varied from 0 to 1, and the ranking of trackers is based on the Area Under Curve (AUC) score. For more details about OOTB and the adopted metrics, we refer readers to [48].

We run the One-Pass Evaluation (OPE) on the benchmark using the proposed MUSTer and use the online available software[4] provided by [48] to compute the evaluation plots. For the competitor trackers, we first consider those

[2]The code to reproduce the experiments is available on `https://sites.google.com/site/multistoretrackermuster/`

[3]`http://opencv.org/`

[4]`http://visual-tracking.net/`

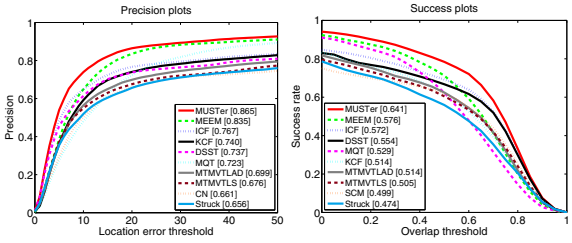

Figure 4. Quantitative comparison in CVPR2013 OOTB. The performance score for each tracker is shown in the legend. For each figure, only the top 10 trackers are presented.

29 popular approaches whose results are available in OOTB, such as Struck [14], TLD [23], SCM [52], VTD [26], VTS [27], and ASLA [22]. And on top of these, we include recent correlation filter-based trackers CN [7], KCF [16], DSST [6], which is the best performing tracker in [24], and further include very recent state-of-the-art trackers MEEM [50], MTMVTLAD [36] and the LGT [47] based on coupled-layer visual model. Last but not least, we also compare quantitatively with the short-term ICF presented in Section 3.1, so as to demonstrate the importance of the long-term component in our proposed tracker.

Figure 3 shows a qualitative comparison with selected trackers on several representative videos/frames. To quantitatively compare all 37 trackers, we show the precision plot and success plot in Figure 4, which indicates that MUSTer achieves overall the best performance using both the metrics and significantly outperforms the second best tracker MEEM with 11% performance gain using the metric of AUC score. As discussed in [48], the AUC score that measures the overall performance in success plot is more accurate than the precision score at one threshold of precision plot. Therefore, the experimental result clearly demonstrates the superior performance of the proposed tracker. In addition, MUSTer also significantly outperforms its baseline short-term component ICF, other correlation filter-based trackers DSST and KCF, and the well-known long-term tracker TLD, which validates the important role of the long-term component in MUSTer and the effectiveness of the proposed tracking framework based on ASMM.

4.2. Evaluation on ALOV++ Dataset

To further validate the robustness of MUSTer, we conducted the second evaluation on a larger dataset [43], namely ALOV++ (Amsterdam Library of Ordinary Videos), which is recently developed by Smeulders *et al.*. It consists of 14 challenge subsets, totally 315 sequences and focuses on systematically and experimentally evaluating trackers' robustnesses in a large variety of situations including light changes, low contrast, occlusion, etc. In [43], survival curves based on F-score were proposed to evaluate track-

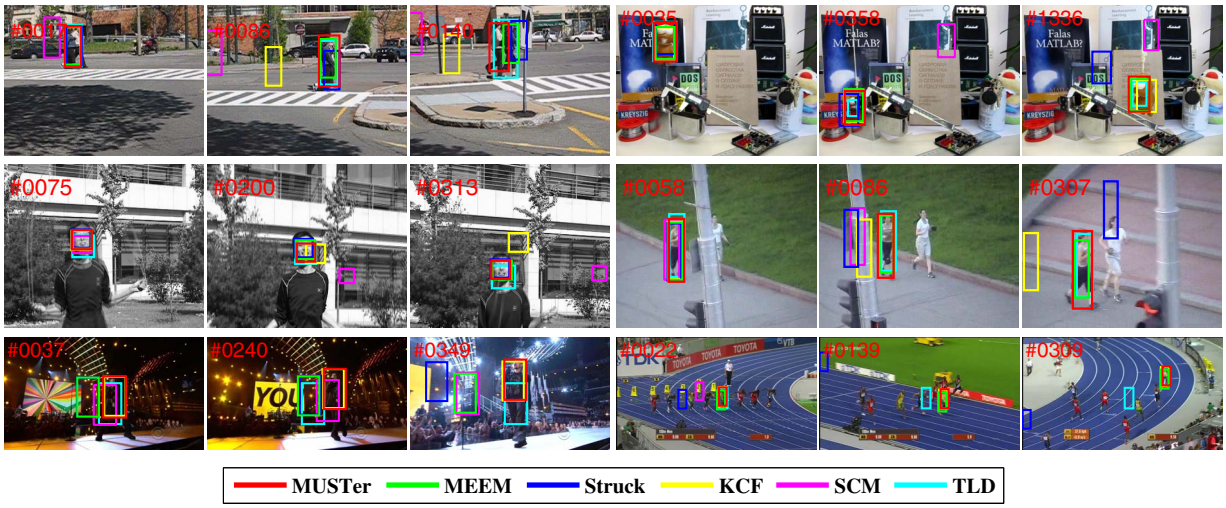

Figure 3. Tracking results of selected algorithms in representative frames. Frame indexes are shown in the top left of each figure. The showing examples are from sequences *Couple*, *Lemming*, *Jumping*, *Jogging*, *Singer2*, *Bolt*, respectively.

ers' robustnesses and demonstrated its effectiveness. To obtain the survival curve of a tracker, a F-score for each video is computed as $F = 2(\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$, where $\text{precision} = n_{tp}/(n_{tp} + n_{fp})$, $\text{recall} = n_{tp}/(n_{tp} + n_{fn})$, and n_{tp}, n_{fp}, n_{fn} respectively denote the number of true positives, false positives and false negatives in a video. A survival curve shows the performance of a tracker on all videos in the dataset. The videos are sorted according to the F-score. By sorting the videos, the graph gives a bird's eye view in cumulative rendition of the quality of the tracker on the whole dataset. We refer the reader to the original paper [43] and the author's website[5] for details about the dataset and the evaluation tools.

To evaluate MUSTer on ALOV++ dataset, we ran MUSTer on all the 315 sequences using the ground truth of the first frame as initialization and the same parameters as the previous evaluation shown in Section 4.1. We compare our tracker with 19 popular trackers[6] that were evaluated in [43]. In addition, we also ran MEEM, ICF, DSST, KCF on ALOV++, which rank on the top five trackers in the previous evaluation. The survival curves of the top ten trackers and the average F-scores over all sequences are shown in Figure 5, which demonstrates that MUSTer achieves the best overall performance over 24 compared trackers in this comparison.

5. Conclusion

In this paper, we propose the MUlti-Store Tracker (MUSTer) based on the Atkinson-Shiffrin Memory Model to handle tracking memory problems. MUSTer consists of two important but relatively independent components and exploits them to process the image input according to the short- and long short-term memories of the target being tracked. In the short-term store, an Integrated Correlation Filter (ICF), which stores the short-term memory and depends on spatiotemporal consistency, is employed to provide an instant response via two-stage filtering. In addition, a complementary component based on keypoint matching-tracking and RANSAC estimation is integrated, which is able to interact with the keypoint feature database in the long-term store and control the final output and the short-term memory states. To maintain a reasonable keypoint feature database size, the well-known forgetting curve is employed to model the remembering-forgetting loop and retain the most useful features. The experimental results on two large datasets demonstrate that the proposed tracker is capable of taking advantage of both the short-term and long-term systems and boosting the tracking performance.

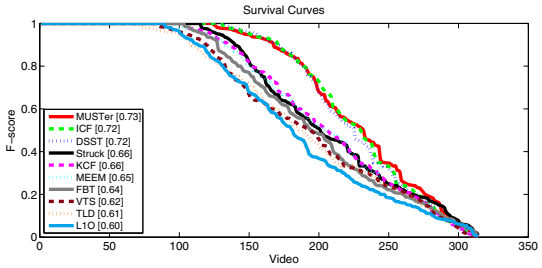

Figure 5. The survival curves for top ten trackers on AlOV++ dataset. The average F-scores over all sequences are specified in the legend.

[5] http://imagelab.ing.unimore.it/dsm/

[6] We refer the reader to [43] and its references for the details about the compared trackers. The evaluation results of these 19 trackers were obtained from the author of [43].

Acknowledgment

This project is supported by Australian Research Council Discovery Projects number DP-120103730, DP-140102164 and FT-130101457, and it is also supported by Toyota Research Institute NA collaborative project 2013001793.

References

[1] R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. *Psychology of learning and motivation*, 2:89–195, 1968.

[2] S. Avidan. Support vector tracking. *TPAMI*, 26(8):1064–1072, 2004.

[3] S. Avidan. Ensemble tracking. *TPAMI*, 29(2):261–271, 2007.

[4] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8):1619–1632, 2011.

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550, 2010.

[6] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[7] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, pages 1090–1097, 2014.

[8] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, pages 1177–1184, 2011.

[9] H. Ebbinghaus. *Memory: A contribution to experimental psychology*. Number 3. Teachers college, Columbia university, 1913.

[10] D. Eck and J. Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 2002.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010.

[12] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, pages 1285–1292, 2010.

[13] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649, 2013.

[14] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011.

[15] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[16] J. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, pages 583–596, 2015.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, pages 702–715. 2012.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Z. Hong, X. Mei, D. Prokhorov, and D. Tao. Tracking via robust multi-task multi-view joint sparse representation. In *ICCV*, pages 649–656, 2013.

[20] Z. Hong, X. Mei, and D. Tao. Dual-force metric learning for robust distracter-resistant tracker. In *ECCV*, pages 513–527, 2012.

[21] Z. Hong, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Tracking using multilevel quantizations. In *ECCV*, pages 155–171, 2014.

[22] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829, 2012.

[23] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 34(7):1409–1422, 2012.

[24] M. Kristan and R. Pflugfelder et al. The visual object tracking VOT2014 challenge results. In *ECCV Workshop*, pages 1–27, 2014.

[25] S. Kwak, W. Nam, B. Han, and J. H. Han. Learning occlusion with likelihoods for visual tracking. In *ICCV*, pages 1551–1558, 2011.

[26] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[27] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *ICCV*, pages 1195–1202, 2011.

[28] J. Kwon, J. Roh, K. M. Lee, and L. Van Gool. Robust visual tracking with double bounding box model. In *ECCV*, pages 377–392, 2014.

[29] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Long-term tracking through failure cases. In *ICCV Workshop*, pages 153–160, 2013.

[30] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *TIST*, 4(4):58, 2013.

[31] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshop*, 2014.

[32] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *TPAMI*, 33(5):978–994, 2011.

[33] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157, 1999.

[34] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[35] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[36] X. Mei, Z. Hong, D. Prokhorov, and D. Tao. Robust multitask multiview tracking in videos. *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

[37] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *TPAMI*, 33(11):2259–2272, 2011.

[38] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *WACV*, pages 862–869, 2014.

[39] S. Oron, A. Bar-Hille, and S. Avidan. Extended lucas-kanade tracking. In *ECCV*, pages 142–156, 2014.

[40] Y. Pang and H. Ling. Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In *ICCV*, pages 2784–2791, 2013.

[41] F. Pernici and A. Del Bimbo. Object tracking by oversampling local features. *TPAMI*, 36(12):2538–2551, 2014.

[42] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[43] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *TPAMI*, 36(7):1442–1468, 2014.

[44] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013.

[45] P. H. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *CVIU*, 78(1):138–156, 2000.

[46] J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning color names for real-world applications. *TIP*, 18(7):1512–1523, 2009.

[47] L. Čehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *TPAMI*, 35(4):941–953, 2013.

[48] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013.

[49] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[50] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203, 2014.

[51] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, pages 2042–2049, 2012.

[52] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845, 2012.