Visual Tracking Using Attention-Modulated Disintegration and Integration

Jongwon Choi[1] Hyung Jin Chang[2] Jiyeoup Jeong[1] Yiannis Demiris[2] Jin Young Choi[1]
[1]ASRI, Dept. of Electrical and Computer Eng., Seoul National University, South Korea
[2]Dept. of Electrical and Electronic Eng., Imperial College London, UK
jwchoi.pil@gmail.com {jy.jeong, jychoi}@snu.ac.kr {hj.chang, y.demiris}@imperial.ac.uk

Abstract

In this paper, we present a novel attention-modulated visual tracking algorithm that decomposes an object into multiple cognitive units, and trains multiple elementary trackers in order to modulate the distribution of attention according to various feature and kernel types. In the integration stage it recombines the units to memorize and recognize the target object effectively. With respect to the elementary trackers, we present a novel attentional feature-based correlation filter (AtCF) that focuses on distinctive attentional features. The effectiveness of the proposed algorithm is validated through experimental comparison with state-of-the-art methods on widely-used tracking benchmark datasets.

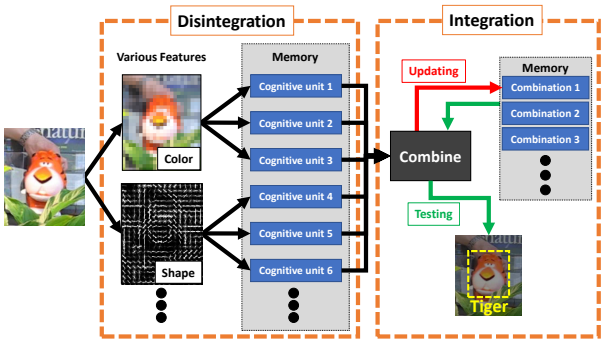

Figure 1. **Structuralist cognitive model for tracking (SCT).** Our SCT model works with two stages: disintegration and integration. By cooperating between the two stages, the target can be trained and found from the input image patch.

1. Introduction

When tracking objects, humans readily find useful features to distinguish the tracking target from background clutter. The ability to find useful features is one of the most powerful aspects of the human visual system. With this robust visual system, humans are capable of adapting stably and rapidly to complex environments, even when encountering background clutter or drastic changes of the target.

Over the last 10 years, remarkable advances have been achieved in visual tracking research, as surveyed in [28,29]. However, even with the advances in performance, the technical level of the current visual tracking cannot achieve the performance of the human visual system. Hence, current visual tracking algorithms have much room for improvement via a human-mimetic approach.

In this paper, we propose a structuralist cognitive model for visual tracking (SCT). As shown in Fig. 1, the SCT is composed of two separate stages: disintegration and integration. In the disintegration stage, the target is divided into a number of small cognitive structural units, which are memorized separately. Each unit includes a specific color or a distinguishable target shape, and is trained by elementary trackers with different types of kernel.

In the integration stage, an adequate combination of the structural units is created and memorized to express the target's appearance. When encountering a target with changing appearance in diverse environments, the SCT utilizes all the responses from the cognitive units memorized in the disintegration stage and then recognizes the target through the best combination of cognitive units, referring to the memorized combinations.

Through disintegration and integration synergies, the SCT produces robust responses to complex inputs. When background clutter appears, the disintegration stage trains the cognitive units to effectively distinguish the target from the clutter. In addition, the best combination of the cognitive units is learned and memorized in the integration stage, which enlarges the discriminability between the clutter and target. If the target is deformed into a sudden change of appearance, the integration reduces the influence of the previously influential cognitive units and enlarges the influence of new appropriate units, which helps the SCT to adapt itself rapidly to the change.

With respect to the elementary trackers, we propose an attentional feature-based correlation filter (AtCF). The AtCF focuses on the attentional features discriminated from the background, which is motivated by the concentration cognition model of human [46]. Each AtCF consists of an

attentional weight estimator and a kernelized correlation filter (KCF) [1]. As a result of the AtCF, the SCT can avoid the problems caused by background features misplaced in the bounding box, such as drift and a missing target. In the disintegration stage, multiple AtCFs are updated using various feature and kernel types. The integration stage combines the responses of AtCFs by ordering the AtCFs following their performance.

2. Related Research

Recent papers provide benchmark datasets containing a large number of videos with ground truth to accelerate the improvement of trackers [27, 42]. Generative trackers [8, 9, 36, 45] generate the features that are robust to the changes of the target's appearance without extracting background clutter. In [8,9,45], Kwon and Lee proposed trackers that generate robust features with PCA-based sparse representation. Gauglitz *et al*. [36] tried to track a target by registering and matching the features obtained from the target. Distinctive trackers [1–7, 22, 25] exploit background clutter to extract distinctive features of the target from the background. Kalal *et al*. [3] presented a tracker by simultaneously updating a target tracker and a negative estimator. Hare *et al*. [4] introduced a structured SVM for tracking, which was trained by structured labels. MEEM [7], which is one of the state-of-the-art trackers, trains multiple linear SVM-based trackers from which one major expert was selected. Gall *et al*. [22] developed a tracker utilizing a random forest and Henriques *et al*. [1] used the ridge regression to consider the samples neighboring the targets.

Recently, correlation filter-based trackers have become increasingly popular [1, 2, 5, 6, 25]. The correlation filter can be trained quickly based on the property of the circulant matrix in the Fourier domain, so the trackers show high performance with low computational load [2]. In particular, Henriques *et al*. [1] improved the performance of the correlation filter-based tracker by extending it to multi-channel input and kernel-based training. Because of the high performance of correlation filter-based trackers, many recent trackers [5,6,25] have utilized correlation filter-based trackers as a baseline. Danellijan *et al*. [6] developed a correlation filter covering the scale change of the target, and Ma *et al*. [25] and Hong *et al*. [5] used the correlation filter as a short-term tracker with an additional long-term memory system. However, the previous correlation filter-based trackers have utilized one fixed type of feature and kernel, so there has been limitations in performance. In this paper, multiple correlation filters using various types of feature and kernel are updated in the disintegration stage, which improves the distinctiveness of target.

Among previous trackers, some approaches [13, 23] also built one strong tracker by combining many trackers as the proposed algorithm. Grabner *et al*. [13] built one strong tracker combined from many equivalent trackers using a boosting algorithm. Yang *et al*. [23] used many trackers trained by various types of feature and kernel, which were combined by bootstrap learning. Unlike the trackers based on boosting and bootstrap learning, elementary trackers of the proposed algorithm are updated independently using samples only from the current frame. The independent update enables the use of a large number of negative samples to improve the performance. Contrary to the previous trackers based on the multiple weak trackers [7, 47, 49], the proposed algorithm uses a fast correlation filter as weak trackers to work in real-time. In addition, against the hybrid trackers fusing the trackers of different structure [5, 48, 50], the proposed framework can be extended easily by supplementing additional feature or kernel type.

The scheme to estimate the attentional features in the AtCF is similar to saliency detection. Research in saliency detection can be categorized into top-down and bottom-up approaches. Top-down saliency [14, 17, 18, 44] is obtained by a classifier pre-trained by the dataset of salient objects and bottom-up saliency [12, 15, 16, 19, 20, 30, 31] is estimated by utilizing only the input image without pre-training. Bottom-up saliency has been applied to foreground detection [30], activity recognition [31], classification and segmentation problems [21, 32–34] and saliency has been applied to a tracker [35]. Contrary to Hong *et al*. [35], the proposed framework does not find the target location directly from saliency map but use the saliency map as a weight map for correlation filters. In addition, the attentional scheme of the proposed algorithm includes both top-down (in the tracking phase) and bottom-up (in the updating phase) factors, while working much faster than [35].

3. Proposed Tracker

3.1. Overview

Our proposed algorithm consists of two stages: disintegration and integration. The scheme of the algorithm is depicted in Fig. 2. In the disintegration stage, multiple AtCFs are generated and updated to cover the various properties of the target. Each AtCF utilizes one of various feature types (HOG, color, etc.) and one of various kernel types (Gaussian, linear, etc.) so that the types of the used feature and kernel are different from those of other AtCFs. When F feature types and K kernel types are used, FK AtCFs are generated in the disintegration stage. In the integration stage, an integrated tracking response is estimated by combining the responses of AtCFs. The combination is updated in every frame, based on priority and reliability measures evaluated by the response of each AtCF.

In the tracking phase, a candidate patch in the new frame is located at the same position, with the previous tracking bounding box under the assumption that the new position

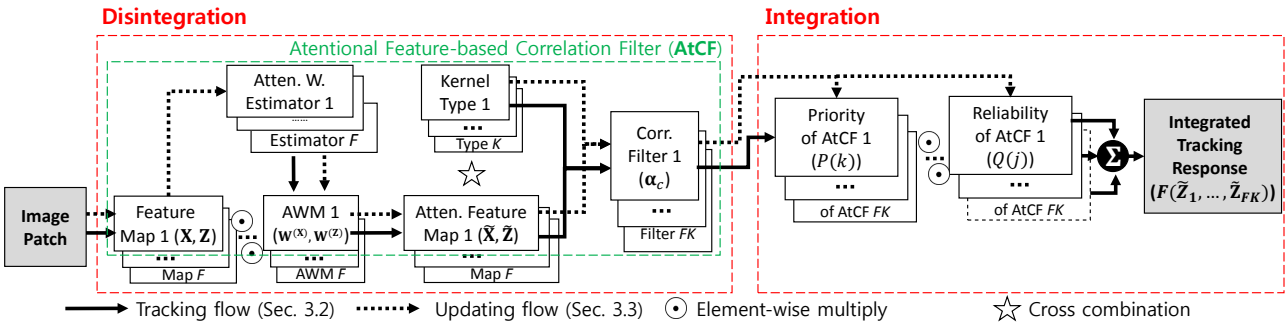

Figure 2. **Proposed Algorithm Scheme.** Our proposed tracker is divided into disintegration and integration stages, which cooperate to maximize the ability to discriminate the target from background clutter. The disintegration consists of multiple AtCFs with various types of feature and kernel, which are combined in the integration to get an integrated tracking response according to the priority and reliability of each AtCF.

of the target is not far from the previous one. However, to cover the movement of the target, the size of the candidate patch is enlarged with s_p times width and height of the bounding box at the same center. The candidate patch is divided into $s_g \times s_g$ pixel-wise grids, and a feature vector is extracted from each grid following the used feature type $f \in \{1, ..., F\}$, which constitutes a feature map $\mathbf{Z}_f$. Because every AtCF works in the same manner, we omit the index of the used feature in the disintegration stage and embed it to the index of each AtCF implicitly in the integration stage: each AtCF is indexed in order by embedding the feature index and kernel index. Letting C be the dimension of feature vector, the grid features from a channel $c \in \{1, ..., C\}$ forms a layer of a 2-dimensional channel feature map $\mathbf{Z}^c$. Then we obtain a cubic-type map $\mathbf{Z}$ stacked with the layers of all $\mathbf{Z}^c, c \in \{1, ..., C\}$. From $\mathbf{Z}$, the attentional weight map (AWM) $\mathbf{W}^{(\mathbf{Z})}$ is calculated by the attentional weight estimator (AWE) updated in the previous frame. The AWE estimates the grid-wise weights from the feature vector of each grid by multiple decision trees, and it assigns a higher weight to the feature vector distinct from the background. Then, the attentional feature map $\widetilde{\mathbf{Z}}$ for tracking is obtained by the element-wise product between $\mathbf{W}^{(\mathbf{Z})}$ and $\mathbf{Z}$. The responses of AtCFs are evaluated by applying their own correlation filters to $\widetilde{\mathbf{Z}}$. The KCF [1] is used for the correlation filter of the AtCF. The responses are combined into the integrated tracking response on the basis of the priorities and reliabilities updated at the previous frame. Finally, the target is determined to be centered at the maximum point of the integrated response. The calculation of the AWM is described at length in section 3.2.1, the calculation procedures for AtCF responses in section 3.2.2, and the integrated tracking response in section 3.2.3.

In the updating phase, each AtCF in the disintegration stage and the priority and reliability for each AtCF are updated. The updating patch is centered at the location of the target determined by the tracking phase. At the first frame, the input target box is centered at the initial target box. The size of the updating patch is set to the same size of the candidate patch in tracking. The updating feature map extracted from the updating patch is denoted by $\mathbf{X}$. The updating feature map $\mathbf{X}$ has the same structure with the tracking feature map $\mathbf{Z}$ and is constructed in the same way with $\mathbf{Z}$. The update of the AtCF includes the update of both the AWE and its correlation filter. To efficiently update the AWE, a partially growing decision tree (PGDT) [26] is used. The updated AWE produces the AWM $\mathbf{W}^{(\mathbf{X})}$ for $\mathbf{X}$, and a smoothed AWM $\mathbf{W}^o$ is obtained by temporally interpolating $\mathbf{W}^{(\mathbf{X})}$ for stable updating. The attentional feature map $\widetilde{\mathbf{X}}$ for updating is obtained by element-wise multiplication of $\mathbf{W}^o$ and $\mathbf{X}$. Then, the correlation filter of the AtCF is updated by using $\widetilde{\mathbf{X}}$. In the integration stage, the priority and reliability of each AtCF are updated from the discrimination ability of the AtCF. The update of the AWE is described in detail in section 3.3.1, the update of the correlation filter in section 3.3.2, and the update for integration in section 3.3.3.

3.2. Tracking

3.2.1 Attentional Weight Map Calculation

AWM $\mathbf{W}$ is obtained by weighted-sum of a strong Attentional Weight Map (sAWM) $\mathbf{W}_s$ and a weak Attentional Weight Map (wAWM) $\mathbf{W}_w$. $\mathbf{W}_s$ gives high weights to the grids inferred as the target and low weights to the grids with similar features to the background. The examples of $\mathbf{W}_s$ are shown in Fig. 3. $\mathbf{W}_w$ is evaluated by the prior knowledge that the target is generally located at the center of an input image patch.

Strong Attentional Weight Map: To obtain $\mathbf{W}_s$, we use AWE trained continuously in every previous frame. AWE is an ensemble of $N_\mathcal{T}$ decision trees. An attentional weight for a feature vector from a grid is allocated at a final leaf of every decision tree in the ensemble. This weight is assigned to an instant attentional weight of a grid whose feature arrives at the leaf. The strong attentional weight of the ith grid in $\mathbf{W}_s$ is estimated by averaging the instant attentional weights obtained by applying $\mathbf{x}_i$ into all the decision trees.

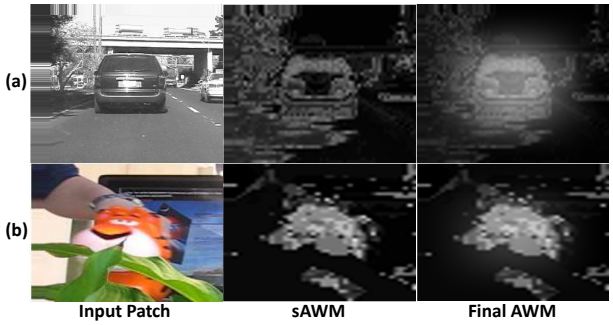

Figure 3. **Attentional Weight Map.** (a) shows AWM obtained by color features from *'Car4'*, which is influenced by wAWM due to the trustless sAWM. (b) represents AWM acquired by color features from *'Tiger2'*, which is determined mainly by sAWM because the sAWM is reliable. (b) also shows that AWM makes the tracker robust to partial occlusion by reducing the weights for the occluding backgrounds like a leaf.

The method to train AWE is explained in Section 3.3.1.

Weak Attentional Weight Map: Following the centered prior of $\mathbf{W}_w$, a cosine window is used as follows:

$$\begin{gathered}\mathbf{W}_w(x,y) = \left(1-cos(2\pi\tfrac{x}{W})\right)\left(1-cos(2\pi\tfrac{y}{H})\right) \\ x \in \{1,...,W\}, y \in \{1,...,H\},\end{gathered} \tag{1}$$

where (x, y) has its origin at the left-top corner and enlarges as going to right-bottom, and W and H indicate the width and height of $\mathbf{Z}$. By applying the cosine window, high weights are allocated to the grids near the center, while low weights are assigned to the grids close to the boundary.

Final Attentional Weight Map: $\mathbf{W}$ is obtained by the weighted sum of $\mathbf{W}_s$ and $\mathbf{W}_w$ as

$$\mathbf{W} \quad = (1-w^f)\mathbf{W}_w + w^f\mathbf{W}_s, \tag{2}$$

where w^f is the dependency on $\mathbf{W}_s$ which controls the influence of $\mathbf{W}_s$ based on the credibility of $\mathbf{W}_s$. The dependency is determined in updating phase at the previous frame, and Fig. 3 shows the effect resulted by varying w^f.

3.2.2 Response of the AtCF

Let $\mathbf{W}^{(\mathbf{Z})}$ be AWM obtained by applying the feature map $\mathbf{Z}$ to AWE and (2). Then, the AWM $\widetilde{\mathbf{Z}}$ is obtained by the element-wise multiplication of $\mathbf{W}^{(\mathbf{Z})}$ and $\mathbf{Z}$ for every channel. Thus, $\widetilde{\mathbf{Z}}^c$ for the channel c is calculated by

$$\widetilde{\mathbf{Z}}^c = \mathbf{Z}^c \odot \mathbf{W}^{(\mathbf{Z})}, c \in \{1,...,C\}, \tag{3}$$

where $\odot$ represents the element-wise multiplication. By imputing $\widetilde{\mathbf{Z}}$ to KCF [1], the response of each AtCF to $\widetilde{\mathbf{Z}}$ is obtained by

$$\begin{aligned}\mathbf{f}(\widetilde{\mathbf{Z}}) \quad &= \mathcal{F}^{-1}\left(\sum_{c=1}^{C}\hat{\mathbf{k}}^{(\bar{\mathbf{M}}_c,\widetilde{\mathbf{Z}}^c)} \odot \hat{\boldsymbol{\alpha}}_c\right) \\ \mathbf{k}_i^{(\mathbf{X}_1,\mathbf{X}_2)} &= \kappa(vec(\mathbf{X}_2), P^{i-1}vec(\mathbf{X}_1)),\end{aligned} \tag{4}$$

where $\mathcal{F}^{-1}(\bullet)$ is the inverse discrete fourier transformation (DFT) operator, $\hat{v}$ represents the vector transformed by DFT from v, and $vec(\bullet)$ is a function vectorizing the input map. $\mathbf{k}_i^{(\mathbf{X}_1,\mathbf{X}_2)}$ is ith value of $\mathbf{k}^{(\mathbf{X}_1,\mathbf{X}_2)}$, κ is a function estimating a distance on the kernel type used in a current AtCF, and P represents a unit cyclic permutation matrix. $\boldsymbol{\alpha}_c$ and $\bar{\mathbf{M}}_c$ are respectively a filter vector and a model feature map for channel $c \in \{1,...,C\}$, which are updated at the previous frame. For more detail, KCF [1] can be referred.

3.2.3 Integrated Tracking Response

The position of target is determined by the integrated tracking response. The integrated tracking response is obtained by combining the responses of AtCFs according to the relative performance (priority) and the absolute performance (reliability). The priority is given to each AtCF based on the order of the performance among the entire AtCFs. The reliability is introduced to additionally control the importance of AtCF on the basis of their own performance.

The response of jth AtCF at the tracking phase is

$$\mathbf{T}^j = \mathbf{f}^j(\widetilde{\mathbf{Z}}_{S(j)}),\ j \in \{1,...,FK\}, \tag{5}$$

where $S(j)$ is a mapping function which returns the index f of the feature type used by jth AtCF. Then, the integrated tracking response $\mathbf{F}$ is obtained by

$$\mathbf{F}(\mathbf{T}^1,...,\mathbf{T}^{FK}) = \sum_{j=1}^{FK}\left(\frac{1}{P_j}\right)^{\lambda_P} Q_j\mathbf{T}^j, \tag{6}$$

where P_j is a priority of jth AtCF and Q_j is a reliability for jth AtCF. Both P_j and Q_j is estimated in the updating phase at the previous frame. In the equation, the first weight term assigns a large weight to the front order of priority, while the second weight term additionally controls the integration based on the reliability. A predefined parameter λ_P determines the degree of influence by the priority order of AtCFs. When λ_P is close to zero, every elementary tracker has the same influence in the integration regardless of the priority order. On the contrary, when λ_P is large enough, the elementary tracker at the first order of priority is used greedily. Finally, the tracking bounding box is determined to be centered at the maximum point of the integrated tracking response.

3.3. Updating

3.3.1 Update of Attentional Weight Estimator (AWE)

AWE is the ensemble of $N_\mathcal{T}$ decision trees, and one tree per each frame is newly created by training and stacked into the ensemble by first-in-first-out strategy. By utilizing the ensemble of decision trees trained from the consecutive multiple frames, AWE can avoid being overfitted by

one frame and biased by a wrong tracking result. The decision tree is trained in non-parametric way without assuming the distribution model of training samples, and it can be tested rapidly. Therefore, the decision tree is proper for the tracking problem where the algorithm should adapt itself to various environments and work fast. However, it takes too much time to be trained at every frame in real-time. To reduce the computational load, we utilize a PGDT structure [26]. PGDT is composed of one main tree and multiple subtrees, where the main tree is trained at the first frame and, in the following frames, only the subtrees are trained and connected to ambiguous leaves of the main tree. The ambiguous leaves are the leaves requiring further decision procedure, in which the similar number of the differently labeled features are left. In a general tracking problem, the large portion of features in the target are kept even in changing appearance. Hence a decision tree trained in the first frame can be shared in other frames. By fixing the main tree, we can reduce the training time because a majority of the duplicated split parameters are reused.

Training of PGDT: At the first frame, only the main tree is trained. For the afterward frames, all the grid features of $\mathbf{X}$ are tested by the main tree to find ambiguous leaves. When ambiguous leaves occur at the main tree, we create subtrees connected to the ambiguous leaves to divide the features left at the leaves. To prevent a memory overflow, all the subtrees are removed and newly created at every frame. The training sample for the decision tree is a grid feature $\mathbf{x}_i$ with a label α_i. For $\mathbf{x}_i$ located in the bounding box, α_i is set to 1, while the others are labeled by $\alpha_i = 0$. At the leaves of the entire tree, the attentional weight is assigned by the ratio of the labels of the samples remaining at the leaves.

The main tree and subtrees of PGDT follow a structure of classification decision tree [24]. The samples of a node are divided to its child nodes by a linear split function as

$$\begin{aligned} \mathcal{N}_L^{(c,\tau)} &= \{\mathbf{x}_i | x_i^c \leq \tau, \mathbf{x}_i \in \mathcal{N}_p\} \\ \mathcal{N}_R^{(c,\tau)} &= \{\mathbf{x}_i | x_i^c > \tau, \mathbf{x}_i \in \mathcal{N}_p\}, \end{aligned} \tag{7}$$

where $\mathcal{N}_p$ is the sample set of a parent node to be divided, and $\mathcal{N}_L^{(c,\tau)}$ and $\mathcal{N}_R^{(c,\tau)}$ are the sample sets of a left child node and a right child node, respectively, when the split parameter of $\mathcal{N}_p$ is set to (c, τ). The best split parameters (c, τ) are determined by the split parameters minimizing a gini-diversity index (gdi) given by

$$\begin{aligned} gdi(c,\tau) &= \sum_{l \in \{\mathcal{N}_L^{(c,\tau)}, \mathcal{N}_R^{(c,\tau)}\}} \{r(l) \times (1 - r(l))\} \\ r(l) &= n_l^{\alpha=1} / (n_l^{\alpha=1} + n_l^{\alpha=0}), \end{aligned} \tag{8}$$

where $n_l^{\alpha=1}$ and $n_l^{\alpha=0}$ are the number of the features with $\alpha_i = 1$ and $\alpha_i = 0$ in a node l, respectively. The split sequence is repeated until one of the following stop criteria is satisfied.

- All remaining features have same label α_i.
- Remaining features cannot be divided anymore.
- The number of remaining features is under 10.

The ambiguous leaves L^a of the main tree are determined by

$$L^a = \{l | \varepsilon \leq r(l) \leq 1 - \varepsilon, l \in L^m\}, \tag{9}$$

where ϵ is a predefined threshold and L^m represents a set containing the entire leaves of the main tree.

Finally, the attentional weight at leaf is assigned by

$$f(l) = \frac{n_l^{\alpha=1}}{n_l^{\alpha=1} + n_l^{\alpha=0}}. \tag{10}$$

Therefore, a small attentional weight is assigned for the feature vector similar to the background features because similar features gather at same leaf.

Update of dependency on sAWM: The dependency on sAWM w^f, which is needed for aggregating sAWM and wAWM in (2), is estimated by

$$w^f = exp(-\beta_w \left\| \mathbf{W}_s - \mathbf{M}^o \right\|^2), \tag{11}$$

where $\mathbf{M}^o$ is the initial teaching label map, where the grids in bounding box are assigned by 1 and the others are by 0. β_w is a predefined parameter to control the value of the dependency on sAWM. When β_w is very large, AWM becomes similar to a cosine window of KCF as sAWM is ignored. When β_w is too small, the tracker can be failed easily by a wrong sAWM caused by distractors.

Full Occlusion Handling: When the target is fully occluded, AWE is trained wrongly by the background occluding the target. To prevent the problem, we detect the full occlusion of the target by

$$\begin{aligned} \tfrac{1}{WH} \sum_{(x,y) \in \mathbf{U}} (\mathbf{1} - \mathbf{W}(x,y)) &> \epsilon_{fo} \\ \mathbf{U} = \{(x,y) | \mathbf{M}^o(x,y) = 1\}&, \end{aligned} \tag{12}$$

where ϵ_{fo} is a pre-defined threshold. In other words, when the entire sum of the attentional weights inside the bounding box becomes too small, the algorithm determines that the full occlusion happens. When the full occlusion is detected, we set the dependency on sAWM w to 0 and stop the updating of AWE during N_{fo} frames.

3.3.2 Update of correlation filter

The correlation filter is updated intensively on the attentional features obtained by AWM. The AWM $\mathbf{W}^{(\mathbf{X})}$ for updating is obtained through the newly updated AWE and the weighted sum as (2). However, when the original $\mathbf{W}^{(\mathbf{X})}$ is used to update the correlation filter, the filter can be fluctuated by drastically changing AWM, causing its unstable

update. To prevent the problem, we estimate a smoothed AWM $\mathbf{W}^o$ by temporally smoothing $\mathbf{W}^{(\mathbf{X})}$.

$$\mathbf{W}^o = \mathbf{W}^o_{(t)} = \gamma \mathbf{W}^o_{(t-1)} + (1-\gamma)\mathbf{W}^{(\mathbf{X})}, \ \mathbf{W}^o_{(0)} = \mathbf{W}_w, \quad (13)$$

where γ is a predefined learning rate. Then, the AFM $\widetilde{\mathbf{X}}$ for updating is obtained by

$$\widetilde{\mathbf{X}}^c = \mathbf{X}^c \odot \mathbf{W}^o, c \in \{1, ..., C\} \quad (14)$$

where $\odot$ represents the element-wise multiplication.

We train the filter $\boldsymbol{\alpha}^o_c = [\boldsymbol{\alpha}^o_{1,c}, ..., \boldsymbol{\alpha}^o_{N,c}]$ of KCF for every channel $c \in \{1, ..., C\}$ as

$$\begin{aligned} \hat{\boldsymbol{\alpha}}^o_{i,c} &= \frac{\hat{\mathbf{y}}_i}{\hat{\mathbf{k}}^{(\widetilde{\mathbf{X}}^c, \widetilde{\mathbf{X}}^c)}_i + \lambda} \\ \mathbf{y} &= [\mathbf{y}^o_{(\lfloor WH/2+1 \rfloor : WH)}, \mathbf{y}^o_{(1:\lfloor WH/2 \rfloor)}] \\ \mathbf{y}^o &= vec\left(\mathcal{G}\left((W/2, H/2), \sigma_G^2, W, H\right)\right), \end{aligned} \quad (15)$$

where λ represents a predefined regularization weight, N is the number of every grid in updating patch, and $\mathbf{k}$ uses the same definition in (4). $\mathbf{y}_i$ is the value of ith element on the desired response vector $\mathbf{y}$. $\mathcal{G}((x, y), \sigma^2, W, H)$ is a 2-dimensional Gaussian window with size $W \times H$, of which the mean and the variance equal (x, y) and σ^2, respectively. σ_G is fixed by $0.025\sqrt{WH}$ as represented in KCF [1].

To train KCF online, we estimate a smoothed filter $\boldsymbol{\alpha}_c$ for channel c by temporally smoothing the estimated filter $\boldsymbol{\alpha}^o_c$. In addition, we hold on a model feature map $\bar{\mathbf{M}}_c$ for channel c to calculate a kernel distance in tracking phase, which is obtained by temporal smoothing as

$$\begin{aligned} \boldsymbol{\alpha}_c &= \boldsymbol{\alpha}_{c,(t)} = \gamma \boldsymbol{\alpha}_{c,(t-1)} + (1-\gamma)\boldsymbol{\alpha}^o_c, \ \boldsymbol{\alpha}_{c,(0)} = \boldsymbol{\alpha}^o_c \\ \bar{\mathbf{M}}_c &= \bar{\mathbf{M}}_{c,(t)} = \gamma \bar{\mathbf{M}}_{c,(t-1)} + (1-\gamma)\widetilde{\mathbf{X}}^c, \ \bar{\mathbf{M}}_{c,(0)} = \widetilde{\mathbf{X}}^c, \end{aligned} \quad (16)$$

where γ is set to the same value in (13). We estimate $\boldsymbol{\alpha}_c$ and $\bar{\mathbf{M}}_c$ for each channel $c \in \{1, ..., C\}$. For the detail explanation of KCF, refer to [1].

3.3.3 Update of Priority and Reliability

To find an adequate combination of AtCF, we update priority and reliability estimates for each AtCF. Both the priority and reliability are newly estimated at every frame to rapidly adapt the integrated response for sudden changes of target. The jth AtCF response map $\mathbf{R}^j$ for updating is obtained by applying the filter to $\mathbf{X}_f$ as

$$\mathbf{R}^j = \mathbf{f}^j(\widetilde{\mathbf{X}}_{S(j)}), \ j \in \{1, ..., FK\}, \quad (17)$$

where $\mathbf{f}^j(\bullet)$ performs equally as (4) for jth AtCF.

Priority Estimation: We update the priority P_j for each AtCF according to the relative tracking performance. Letting N be the number of grids in $\mathbf{X}_f$, $\mathbf{R}^j_i, i = 1, ..., N$ is the response value of the jth AtCF for the ith grid feature, and $\mathbf{y}_i$ is the ith value of the desired response vector in (15). The priority is assigned successively by the order of low error obtained by the weighted-sum of grid errors between $\mathbf{R}^j_i$ and $\mathbf{y}_i$, for all $i = 1, ..., N$. In every assignment of priority, each grid weight $w_i, i = 1, ..., N$, is increased by multiplying a term proportional to the grid error of the assigned AtCF. This implies the large grid error leads to a large grid weight which increases the possibility to select the next AtCF with small error for the grid. Therefore, in the next priority assignment, we can select an AtCF well-discriminating the clutter which is hard to be distinguished by the previously assigned AtCFs. The update scheme is summarized in Algorithm 1.

Algorithm 1: The update of priority

$V = \{1, ..., FK\}, \omega_i = 1 \ for \ \forall i \in \{1, ..., N\}$;
for $k = 1...FK$ **do**
  $j = arg_l \min_{l \in V} \sum_{i=1}^{N} \omega_i \left\| \mathbf{R}^l_i - \mathbf{y}_i \right\|_2^2, P_j = k$
  $\omega_i = \omega_i \exp\left(\left\| \mathbf{R}^j_i - \mathbf{y}_i \right\|_2^2 \right) \ for \ i \in \{1, ..., N\}$
  $V = V - \{j\}$
end

Reliability Estimation: The reliability is obtained by the error between the response of updated filter and the desired response without any comparison to the other AtCFs. The reliability for jth AtCF is estimated by

$$Q_j = \exp\left(-\lambda_Q \left\| \mathbf{R}^j - \mathbf{y} \right\|_2\right), \quad (18)$$

where λ_Q is a predefined parameter and $\mathbf{y}$ equals the desired filter response of (15).

4. Experimental Result

We implemented two trackers: an SCT with six AtCFs (SCT6) and an SCT with four AtCFs (SCT4). For SCT6 and SCT4, we used two feature types, including a six-channel average of RGB and lab color and 31-bin histogram of oriented gradients (HOG) [43]. The kernel types used by SCT6 were Gaussian, polynomial, and linear kernel, while the linear kernel was excluded in SCT4. In addition, to analyze the variants of the proposed algorithm, we implemented two additional trackers: SCT-KCF6 and SCT-DT6. SCT-KCF6 is a tracker combining six KCFs by the SCM-based framework, so the attentional weight map is not utilized against SCT6. SCT-DT6 exchanged PGDT of SCT6 to a general classification decision tree [24].

4.1. Implementation

We selected the design parameters by experiments and from references. Fig. 4 (a) shows the performance with various β_w in (11) and λ_P in (6), which are two major parameters of the framework. Following the experiments, β_w

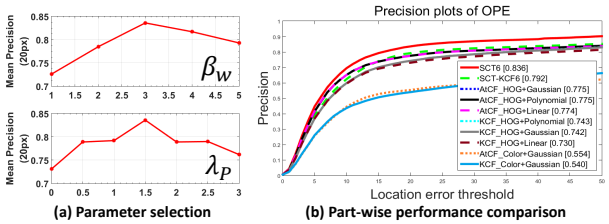

Figure 4. **Precision plot for self-comparison.** The scores next to the name of tracker are the precision values when the location error threshold equals 20 pixel. Only the top 10 trackers are presented.

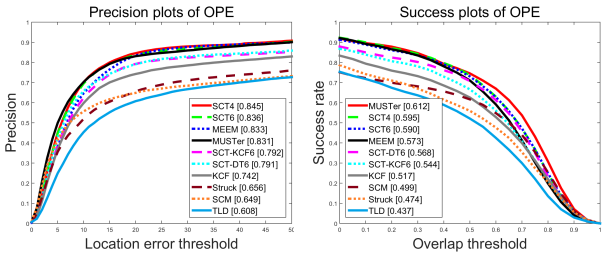

Figure 5. **Precision Plot for Entire Sequences.** Two precision curves based on the location error threshold and the overlap threshold are presented. For a left graph, the scores next to the name of tracker are the precision values when the location error threshold is 20 pixel. For a right graph, the AUC scores is presented instead. Only the top 10 trackers are presented.

and λ_P are set to 3 and 1.5, respectively. We select a moderately small value, 0.01, for λ_Q in order not to invert the order of weights determined by the priority. ϵ_{fo} in (3) and N_{fo} are set to 0.4 and 10 by some experiments with frames of full occlusion. The other parameters, such as $\epsilon = 0.4$ in (6), $s_p = 2.5$ and $s_g = 4$ in Sec. 3.1, $\lambda = 10^{-4}$ in (15), $\gamma = 0.98$ in (13) and (16), and the other parameters for kernel distances of KCFs, were set to the values presented in their references [1, 26]. Following the protocols proposed in [27], all the parameters were fixed for every video sequence. We used the first bounding box obtained from ground truth as an initial input for the trackers.

The program was implemented in MATLAB, except for the decision tree of SCT-DT6 and the PGDT developed by the Piotr library [37]. The test environment was using a 4 core 3.40GHz CPU, 16GB memory, and NVIDIA GTX650. With the unoptimized program, SCT4 and SCT6 reported 40fps and 37fps, respectively, as an average computational speed for the OOTB dataset [27]. The program and benchmark results are uploaded online.[1]

4.2. Evaluation

The proposed algorithms were evaluated using an OOTB dataset [27] containing 50 video sequences with ground truth. Because the sequences of OOTB include various environments to evaluate the general performance of a tracker, OOTB has been frequently utilized by visual tracking research groups [1, 4, 5, 7].

For the performance measure, we used an average precision curve of one-pass evaluation (OPE) proposed in [27]. The average precision curve was estimated by averaging the precision curves of all sequences, which was obtained using two bases: location error threshold and overlap threshold. The precision curve based on the location error threshold shows the percentage of correctly tracked frames on the basis of a distance between the centers of the tracked box and ground truth. The precision curve based on the overlap threshold indicates the percentage of correctly tracked frames on the basis of the overlap region between the tracked box and ground truth. Therefore, a higher precision at a lower threshold means that the tracker works more accurately. The precision curve is efficient because all the precisions can be obtained with fixed parameters.

[1] https://sites.google.com/site/jwchoivision/

Table 1. Summary of Experiments for Entire Sequences

	Algorithm	Mean pre. (20px)	Mean FPS
Proposed	SCT4	**84.5%**	40.0
	SCT6	83.6%	37.1
	SCT-KCF6	79.2%	96.2
	SCT-DT6	79.1%	22.90
Other algorithms	MUSTer [5]	83.1%	3.85
	MEEM [7]	83.3%	19.5
	KCF [1]	74.2%	**223.8**
	Struck [4]	65.6%	10.0
	SCM [10]	64.9%	0.4
	TLD [3]	60.8%	21.7

4.3. Experiments on the full dataset

We conducted an experiment to show the performance of each part, which is shown in Fig. 4 (b). Each single AtCF shows a better performance than KCFs using the same feature and kernel, which means AWM improves trackers. When the performance of SCT6 and SCT-KCF6 is compared to their individual trackers, it can be verified that the integration part of the SCM-based framework also contributes to the meaningful improvement of performance.

The quantitative results compared with the previous works are summarized in Fig. 5 and Table 1. The results of the state-of-the-art methods, including MUSTer [5], MEEM [7], and KCF [1], were obtained by the authors' programs. In the case of MUSTer, we averaged the results of the experiments repeated using five computers. For a large comparison, we additionally used the results of Struck [4], SCM [10], TLD [3], VTS [9], DFT [41], CSK [40], ASLA [11], MIL [38], and CT [39], which were available in the OOTB dataset.

As shown in Fig. 5, SCT4 and SCT6 demonstrated comparable performance to the state-of-the-art methods. Interestingly, even with a small number of elementary trackers, SCT4 presented a higher performance than SCT6. Gaussian

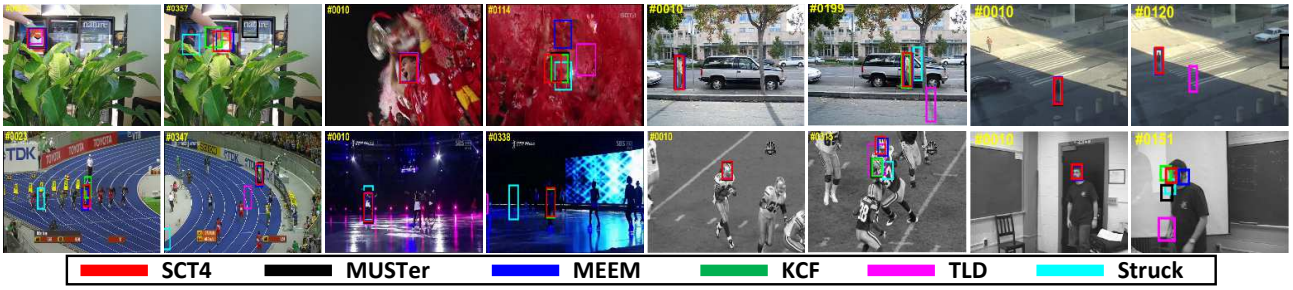

Figure 6. **Qualitative Results.** The tracking results of several trackers are shown. The used sequences are *Tiger1*, *Soccer*, *David3*, *Crossing*, *Bolt*, *Skating1*, *Football*, *Freeman1*, *Coke*, *Jogging*, *Matrix*, and *Shaking*, respectively. The frame indexes are in the top-left of each image.

kernel and polynomial kernel can separate the samples distinguished by the linear kernel, which causes two or more elementary trackers to learn the same property. As a consequence, the duplicated elementary trackers might be redundant or cause an overfitting, although these are trivial.

In Table 1, the precision scores of the top-ranked 10 trackers are presented with their computational speed. The precision score is the average precision when the location error threshold is 20 pixels. The proposed algorithms run sufficiently fast to be used in real time, which is up to 10 times faster than the state-of-the-art methods such as MUSTer and MEEM. The results of SCT6 and SCT-DT6 show that the use of the PGDT outperforms the general decision tree in addition to the merit of fast computational speed. The improvement came from the PGDT, which prevented overfitting by retaining the split parameters of the main tree. The qualitative results of SCT4 and the representative trackers (MUSTer, MEEM, KCF, TLD, Struck) are shown in Fig. 6.

4.4. Experiments with attribute subsets

The sequences of the OOTB dataset [27] are annotated by attributes that are challenging problems in visual tracking. To prove the contributions of the proposed algorithm, we executed the experiments using subsets with four attributes: background clutter, occlusion, illumination variation, and deformation.

The results are shown in Fig. 7, where SCT4 and SCT6 outperform the state-of-the-art methods for all the attributes. According to the results for the subset of background clutter, the proposed algorithm has the potential to distinguish the target from background clutter by updating multiple AtCFs with various feature and kernel types. The results for the subsets of illumination variation and deformation show that the proposed algorithm can adapt rapidly to drastic changes of the target's appearance through the instant update of combination in the integration stage. The results for the subset of occlusion show robustness against occlusions as a result of the AtCF reducing the weights for the background features occluding the target.

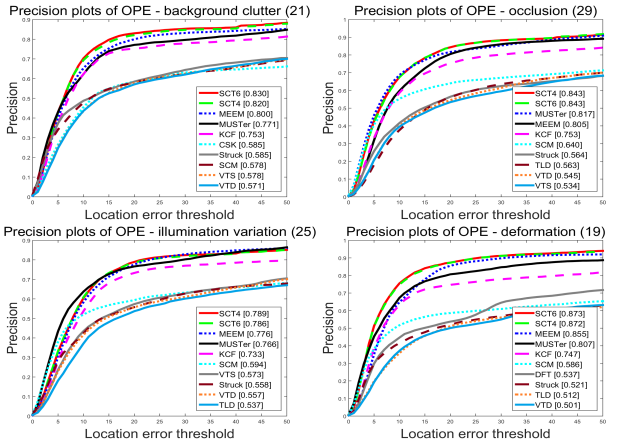

Figure 7. **Precision Plots for Attribute Subsets.** The plots show the results from the experiments using the subsets of 4 attributes: background clutter, occlusion, illumination variation, and deformation. The precision scores are shown in the legend. Only the top 10 trackers are presented.

5. Conclusion

In this paper, a new visual tracking framework was proposed. The proposed tracker worked in two major stages: disintegration and integration. In the disintegration stage, the target was trained by multiple elementary trackers with various types of feature and kernel, which improved the ability to discriminate the target from background clutter. In the integration stage, the responses of the multiple elementary trackers were combined according to the memorized priorities and reliabilities, with quick adaptation against sudden changes of the target's appearance. As the elementary tracker, the AtCF was suggested, which demonstrated robustness to partial occlusion and drift by reducing the weights for the background features in the bounding box of the target. The contributions of the proposed tracker were validated in experiments using a number of sequences contained in the OOTB datasets.

Acknowledgment: This work was partly supported by the ICT RD program of MSIP/IITP[B0101-15-0552, Development of Predictive Visual Intelligence Technology], the Brain Korea 21 Plus Project, and EU FP7 project WYSIWYD under Grant 612139.

References

[1] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters", *IEEE Transactions on PAMI,* 2015. 2, 3, 4, 6, 7

[2] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual Object Tracking using Adaptive Correlation Filters", *CVPR,* 2010. 2

[3] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection", *IEEE Transactions on PAMI,* 2010. 2, 7

[4] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured Output Tracking with Kernels", *ICCV,* 2011. 2, 7

[5] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "MUlti-Store Tracker (MUSTer): a Cognitive Psychology Inspired Approach to Object Tracking", *CVPR,* 2015. 2, 7

[6] M. Danellijan, G. Häger, F. S. Khan, M. Felsberg, "Accurate Scale Estimation for Robust Visual Tracking", *BMVC,* 2014. 2

[7] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust Tracking via Multiple Experts using Entropy Minimization", *ECCV,* 2014. 2, 7

[8] J. Kwon and K. M. Lee, "Visual Tracking Decomposition", *CVPR,* 2010. 2

[9] J. Kwon and K. M. Lee, "Tracking by Sampling Trackers", *ICCV,* 2011. 2, 7

[10] W. Zhong, H. Lu, and M.-H. Yang, "Robust Object Tracking via Sparsity-based Collaborative Model", *CVPR,* 2012. 7

[11] X. Jia, H. Lu, and M.-H. Yang, "Visual Tracking via Adaptive Structural Local Sparse Appearance Model", *CVPR,* 2012. 7

[12] J. Kim, D. Han, Y. Tai, and J. Kim, "Salient Region Detection via High-Dimensional Color Transform", *CVPR,* 2014. 2

[13] H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-line Boosting", *BMVC,* 2006. 2

[14] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep Networks for Saliency Detection via Local Estimation and Global Search", *CVPR,* 2015. 2

[15] N. Tong, H. Lu, X. Ruan, and M.-H. Yang, "Salient Object Detection via Bootstrap Learning", *CVPR,* 2015. 2

[16] C. Li, Y. Yuan, W. Cai, Y. Xia, and D. D. Feng, "Robust Saliency Detection via Regularized Random Walks Ranking", *CVPR,* 2015. 2

[17] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency Detection by Multi-Context Deep Learning", *CVPR,* 2015. 2

[18] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, "Salient Object Detection: A Discriminative Regional Feature Integration Approach", *CVPR,* 2013. 2

[19] C. Gong, D. Tao, W. Liu, S. J. Maybank, M. Fang, K. Fu, and J. Yang, "Saliency Propagation from Simple to Difficult", *CVPR,* 2015. 2

[20] X. Li, H. Lu, L. Zhang, X. Ruan, and M.-H. Yang, "Saliency Detection via Dense and Sparse Reconstruction", *ICCV,* 2013. 2

[21] W. Wang, J. Shen, and F. Porikli, "Saliency-Aware Geodesic Video Object Segmentation", *CVPR,* 2015. 2

[22] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky, "Hough Forests for Object Detection, Tracking, and Action Recognition", *IEEE Transactions on PAMI,* 2011. 2

[23] F. Yang, H. Lu, and Y. Chen, "Human Tracking by Multiple Kernel Boosting with Locality Affinity Constraints", *ACCV,* 2010. 2

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees", *Wadsworth & Brooks/Cole Advanced Books & Software,* 1984. 5, 6

[25] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term Correlation Tracking", *CVPR,* 2015. 2

[26] J. Choi and J. Y. Choi, "User Interactive Segmentation with Partially Growing Random Forest", *ICIP,* 2015. 3, 5, 7

[27] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark", *CVPR,* 2010. 2, 7, 8

[28] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review", *Neurocomputing,* 2011. 1

[29] K. Cannons, "A Review of Visual Tracking", *Technical Report CSE-2008-07, York University, Canada,* 2008. 1

[30] H. J. Chang, H. Jeong and J. Y. Choi, "Active attentional sampling for speed-up of background subtraction", *CVPR,* 2012. 2

[31] K. Lee, D. Ognibene, H. J. Chang, T. K. Kim and Y. Demiris, "STARE: Spatio-Temporal Attention Relocation for Multiple Structured Activities Detection", *IEEE Transactions on Image Processing*,2015. 2

[32] G. Sharma, F. Jurie, and C. Schmid, "Discriminative Spatial Saliency for Image Classification", *CVPR,* 2012. 2

[33] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei, "Object-centric Spatial Pooling for Image Classification", *ECCV,* 2012. 2

[34] B. Heo, H. Jeong, J. Kim, S. Choi, and J. Y. Choi, "Weighted Pooling Based on Visual Saliency for Image Classification", *ISVC,* 2014. 2

[35] S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network", *ICML,* 2015. 2

[36] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking", *IJCV,* 2011. 2

[37] P. Dollár, "Piotr's Computer Vision Matlab Toolbox (PMT)", *http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html*. 7

[38] B. Babenko, M.-H. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning", *TPAMI,* 2011. 7

[39] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time Compressive Tracking", *ECCV,* 2012. 7

[40] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the Circulant Structure of Tracking-by-detection with Kernels", *ECCV,* 2012. 7

[41] L. Sevilla-Lara and E. Learned-Miller, "Distribution Fields for Tracking", *CVPR,* 2012. 7

[42] A.W.M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual Tracking: an Experimental Survey", *TPAMI,* 2013. 2

[43] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models", *TPAMI,* 2010. 6

[44] Y. Lu, W. Zhang, C. Jin, and X. Xue, "Learning Attention Map from Images", *CVPR,* 2012. 2

[45] J. Kwon and K. M. Lee, "Tracking by Sampling and Integrating Multiple Trackers", *TPAMI,* 2014. 2

[46] Y. Yeshurun and M. Carrasco, "Attention Improves or Impairs Visual Performance by Enhancing Spatial Resolution", *Nature,* 1998. 1

[47] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, "Randomized Ensemble Tracking", *ICCV,* 2013. 2

[48] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer Learning Based Visual Tracking with Gaussian Processes Regression", *ECCV,* 2014. 2

[49] D. Lee, J. Sim, and C. Kim, "Multihypothesis Trajectory Analysis for Robust Visual Tracking", *CVPR,* 2015. 2

[50] B. Lee, K. Yun, J. Choi, J. Y. Choi, "Robust Pan-Tilt-Zoom Tracking via Optimization Combining Motion Features and Appearance Correlations", *Conference on Advanced Video and Signal based Surveillance,* 2015. 2