Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network

Seunghoon Hong[1] MAGA33@POSTECH.AC.KR
Tackgeun You[1] YOUTK@POSTECH.AC.KR
Suha Kwak[2] SUHA.KWAK@INRIA.FR
Bohyung Han[1] BHHAN@POSTECH.AC.KR
[1]Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea
[2]Inria–WILLOW Project, Paris, France

Abstract

We propose an online visual tracking algorithm by learning discriminative saliency map using Convolutional Neural Network (CNN). Given a CNN pre-trained on a large-scale image repository in offline, our algorithm takes outputs from hidden layers of the network as feature descriptors since they show excellent representation performance in various general visual recognition problems. The features are used to learn discriminative target appearance models using an online Support Vector Machine (SVM). In addition, we construct target-specific saliency map by back-projecting CNN features with guidance of the SVM, and obtain the final tracking result in each frame based on the appearance model generatively constructed with the saliency map. Since the saliency map reveals spatial configuration of target effectively, it improves target localization accuracy and enables us to achieve pixel-level target segmentation. We verify the effectiveness of our tracking algorithm through extensive experiment on a challenging benchmark, where our method illustrates outstanding performance compared to the state-of-the-art tracking algorithms.

1. Introduction

Object tracking has played important roles in a wide range of computer vision applications. Although it has been studied extensively during past decades, object tracking is still a difficult problem due to many challenges in real world videos such as occlusion, pose variations, illumination changes, fast motion, and background clutter. Success in object tracking relies heavily on how robust the representation of target appearance is against such challenges.

For this reason, target appearance modeling algorithms have been studied actively, and they are classified into two major categories depending on learning strategies: generative and discriminative methods. In generative framework, the target appearance is typically described by a statistical model estimated from tracking results in previous frames. To maintain the target appearance model, various approaches have been proposed including sparse representation (Bao et al., 2012; Jia et al., 2012; Mei & Ling, 2009; Zhang et al., 2012; Zhong et al., 2012), online density estimation (Han et al., 2008), incremental subspace learning (Ross et al., 2004), etc. On the other hand, discriminative framework aims to learn a classifier that discriminates target from surrounding background. Various learning algorithms have been incorporated including online boosting (Grabner et al., 2006; Saffari et al., 2010), multiple instance learning (Babenko et al., 2011), structured SVM (Hare et al., 2011), and online random forest (Gall et al., 2011; Schulter et al., 2011). These approaches are limited to using too simple and/or hand-crafted features for target representation, such as template, Haar-like features, histogram features and so on, which may not be effective to handle latent challenges imposed on video sequences.

Convolutional Neural Network (CNN) has recently drawn a lot of attention in computer vision community due to its representation power. (Krizhevsky et al., 2012) trained a network using 1.2 million images for image classification and demonstrated significantly improved performance in ImageNet challenge (Berg et al., 2012). Since the huge success of this work, CNN has been applied to representing images or objects in various computer vision tasks including object detection (Girshick et al., 2014; Sermanet et al., 2014; He et al., 2014), object recognition (Oquab et al., 2014; Donahue et al., 2014; Zhang et al., 2014), pose

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

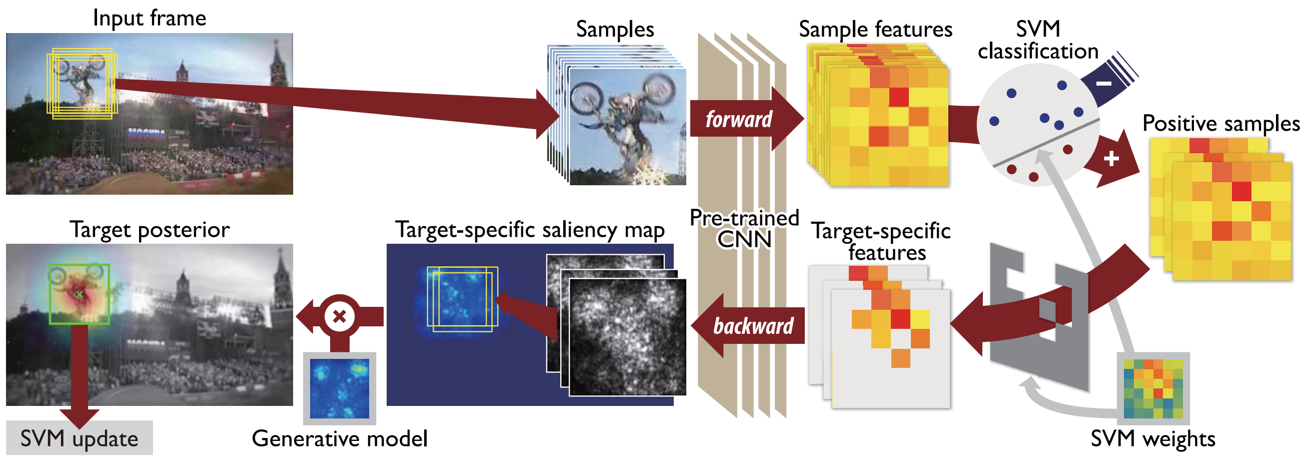

Figure 1. Overall procedure of the proposed algorithm. Our tracker exploits a pre-trained CNN for both image representation and target localization. Given a set of samples on the input frame, we first extract their features using a pre-trained CNN (Section 3.1), and classify them by the online SVM trained until the previous time step. For each positive sample, we back-project the features relevant to target, which are identified by observing the model parameter of the SVM, through the network to obtain a saliency map of the sample that highlights the regions discriminating target from background. The saliency maps of the positive examples are aggregated to build the target-specific saliency map (Section 3.2). Finally, tracking is performed by a sequential Bayesian filtering using the target-specific saliency map as observation. To this end, a generative model is learned from target appearances in the previous saliency maps, and a dense likelihood map is calculated by convolution between the appearance model and the target-specific saliency map (Section 3.3). Based on the tracking result of the current frame, the SVM and generative model are updated for subsequent tracking (Section 3.4).

estimation (Toshev & Szegedy, 2014), semantic segmentation (Hariharan et al., 2014), etc.

Despite such popularity, there are only few attempts to employ CNNs for visual tracking since offline classifiers are not appropriate for visual tracking conceptually and online learning based on CNN is not straightforward due to large network size and lack of training data. In addition, the feature extraction from the deep structure may not be appropriate for visual tracking because the visual features extracted from top layers encode semantic information and exhibit relatively poor localization performance in general. (Fan et al., 2010) presents a human tracking algorithm based on a network trained offline, but it needs to learn a separate class-specific network to track other kind of objects. On the other hand, (Li et al., 2014a) proposes a target-specific CNN for object tracking, where the CNN is trained incrementally during tracking with new examples obtained online. The network used in this work is shallow since learning a deep network using a limited number of training examples is challenging, and the algorithm fails to take advantage of rich information extracted from deep CNNs. There is a tracking algorithm based on a pre-trained network (Wang & Yeung, 2013), where a stacked denoising autoencoder is trained using a large number of images to learn generic image features. Since this network is trained with tiny gray images and has no shared weight, its representation power is limited compared to recently proposed CNNs.

We propose a novel visual tracking algorithm based on a pre-trained CNN, where the network is trained originally for large-scale image classification and the learned representation is transferred to describe target. On top of the hidden layers in the CNN, we put an additional layer of an online Support Vector Machine (SVM) to learn a target appearance discriminatively against background. The model learned by SVM is used to compute a target-specific saliency map by back-projecting the information relevant to target to input image space (Simonyan et al., 2014). We exploit the target-specific saliency map to obtain generative target appearance models (filters) and perform tracking with understanding of spatial configuration of target. The overview of our algorithm is illustrated in Figure 1, and the contributions of this paper are summarized below:

- Although recent tracking methods based on CNN typically attempt to learn a network in an online manner (Li et al., 2014a), our algorithm employs a pre-trained CNN to represent generic objects for tracking and achieves outstanding performance empirically.

- We propose a technique to construct a target-specific saliency map by back-projecting only relevant features through CNN, which overcomes the limitation of the existing method to visualize saliency corresponding to the predefined classes only. This technique also enable us to obtain pixel-level target segmentation.

- We learn a simple target-specific appearance filter online and apply it to the saliency map; this strategy improves target localization performance even with shift-invariant property of CNN-based features.

The rest of this paper is organized as follows. We first describe the overall framework of our algorithm in Section 2 and the detailed methodology is discussed in Section 3. The performance of our algorithm is presented in Section 4.

2. Overview of Our Algorithm

Our tracking algorithm employs a pre-trained CNN to represent target. In each frame, it first draws samples for candidate bounding boxes near the target location in the previous frame, takes their image observations, and extracts feature descriptors for the samples using the pre-trained CNN. We found out that the features from the CNN capture semantic information of target effectively and handle various geometric and photometric transformations successfully as reported in (Oquab et al., 2014; Karayev et al., 2014; Donahue et al., 2014). However, it may lose some spatial information of the target due to pooling operations in CNN, which is not desirable for tracking since the spatial configuration is a useful cue for accurate target localization.

To fully exploit the representation power of CNN features while preserving spatial information of target, we adopt the target-specific saliency map as our observation for tracking, which is generated by back-projecting target-specific information of CNN features to input layer. This technique is inspired by (Simonyan et al., 2014), where class-specific saliency map is constructed by back-projecting the information corresponding to the identified label to visualize the region of interest. Since target in visual tracking problem belongs to an arbitrary class and its label is unknown in advance, the model for target class is hard to be pre-trained.

Hence, we employ an online SVM, which discriminates target from background by learning target-specific information in the CNN features; the target-specific information learned by the online SVM can be regarded as label information in the context of (Simonyan et al., 2014). The SVM classifies each sample, and we compute the saliency map for each positive example by back-projecting its CNN feature along the pre-trained CNN with guidance of the SVM till the input layer. Each saliency map highlights regions discriminating target from background. The saliency maps of the positive examples are aggregated to build the target-specific saliency map. The target-specific saliency map alleviates the limitation of CNN features for tracking by providing important spatial configuration of target.

Our tracking algorithm is then formulated as a sequential Bayesian filtering framework using the target-specific saliency map for observation in tracking. A generative appearance model is constructed by accumulating target observations in target-specific saliency maps over time, which reveals meaningful spatial configuration of target such as shape and parts. A dense likelihood map of each frame is computed efficiently by convolution between the target-specific saliency map and the generative appearance model. The overall algorithm is illustrated in Figure 1.

Our algorithm exploits the discriminative properties of online SVM, which helps generate target-specific saliency map. In addition, we construct the generative appearance model from the saliency map and perform tracking through sequential Bayesian filtering. This is a natural combination of discriminative and generative approaches, and we take the benefits from both frameworks.

3. Proposed Algorithm

This section describes the comprehensive procedure of our tracking algorithm. We first discuss the features obtained from pre-trained CNN. The method to construct target-specific saliency map are presented in detail, and how the saliency map can be employed for constructing generative models and tracking object is described. After that, we present online SVM technique employed to learn target appearance in a discriminative manner sequentially.

3.1. Pre-Trained CNN for Feature Descriptor

To represent target appearances, our tracking algorithm employs a CNN, which is pre-trained on a large number of images. The pre-trained generic model is useful especially for online tracking since it is not straightforward to collect a sufficient number of training data. In this paper, R-CNN (Girshick et al., 2014) is adopted as the pre-trained model, but other CNN models can be used alternatively. Out of the entire network structure, we take outputs from the first fully-connected layer as they tend to capture general characteristics of objects and have shown excellent generalization performance in many other domains as described in (Donahue et al., 2014).

For a target proposal $\mathbf{x}_i$, the CNN takes its corresponding image observation $\mathbf{z}_i$ as its input, and returns an output from the first fully-connected layer $\phi(\mathbf{x}_i)$ as a feature vector of $\mathbf{x}_i$. We apply the SVM to each CNN feature vector $\phi(\mathbf{x}_i)$ and classify $\mathbf{x}_i$ into either positive or negative.

3.2. Target-Specific Saliency Map Estimation

For target tracking, we first compute SVM scores of candidate samples represented by the CNN features and classify them into target or background. Based on this information, one naïve option to complete tracking is to simply select the optimal sample with the maximum score as

$$\mathbf{x}^* = \arg\max_i \mathbf{w}^\top \phi(\mathbf{x}_i).$$

However, this approach typically has the limitation of inaccurate target localization since, when calculating $\phi(\mathbf{x}_i)$, the spatial configuration of target may be lost by spatial pooling operations (Fan et al., 2010).

To handle the localization issue while enjoying the effectiveness of CNN features, we propose the target-specific saliency map, which highlights discriminative target regions within the image. This is motivated by the class-specific saliency map discussed in (Simonyan et al., 2014). The class-specific saliency map of a given image I is the gradient of class score $S_c(I)$ with respect to the image as

$$g_c(I) = \frac{\partial S_c(I)}{\partial I}. \tag{1}$$

The saliency map is constructed by back-propagation. Specifically, let $f^{(1)}, \dots, f^{(L)}$ and $F^{(1)}, \dots, F^{(L)}$ denote the transformation functions and their outputs in the network, where $F^{(l)} = f^{(l)} \circ f^{(l-1)} \circ \cdots \circ f^{(1)}(x)$ and $S_c(I) = F^{(L)}$. Eq. (1) is computed using chain rule as

$$\frac{\partial S_c(I)}{\partial I} = \frac{\partial F^{(L)}}{\partial F^{(L-1)}} \frac{\partial F^{(L-1)}}{\partial F^{(L-2)}} \cdots \frac{\partial F^{(1)}}{\partial I}. \tag{2}$$

Intuitively, the pixels that are closely related to the class c affect changes in S_c more, which means that nearby regions of such pixels would have high values in saliency map.

When calculating such saliency map for object tracking, we impose target-specific information instead of class membership due to the reasons discussed in Section 2. For the purpose, we adopt the SVM weight vector $\mathbf{w} = (w_1, \dots, w_n)^\top$, which is learned online to discriminate between target and background. Since the last fully-connected layer corresponds to the online SVM, the outputs of the last two layers in our network are given by

$$F^{(L)} = \mathbf{w}^T F^{(L-1)} + b \tag{3}$$

$$F^{(L-1)} = \phi(\mathbf{x}_i). \tag{4}$$

Plugging Eq. (3) and (4) into Eq. (2), the gradient map of the target proposal $\mathbf{x}_i$ is given by

$$g(\mathbf{x}_i) = \frac{\partial F^{(L)}}{\partial F^{(L-1)}} \frac{\partial F^{(L-1)}}{\partial \mathbf{z}_i} = \mathbf{w}^{\mathrm{T}} \left(\frac{\partial \phi(\mathbf{x}_i)}{\partial \mathbf{z}_i} \right), \tag{5}$$

where $\mathbf{z}_i$ is the image observation of $\mathbf{x}_i$.

Instead of using all entries in $\phi(\mathbf{x}_i)$ to generate target-specific saliency map, we only select the dimensions corresponding to positive weights in $\mathbf{w}$ since they have clearer contribution to make $\mathbf{x}_i$ positive. Note that every element in $\phi(\mathbf{x}_i)$ is positive due to ReLU operations in CNN learning. Then, we obtain the target-specific feature $\phi^+(\mathbf{x}_i)$ as

$$\phi_k^+(\mathbf{x}_i) = \begin{cases} w_k \phi_k(\mathbf{x}_i), & \text{if } w_k > 0 \\ 0, & \text{otherwise} \end{cases},$$

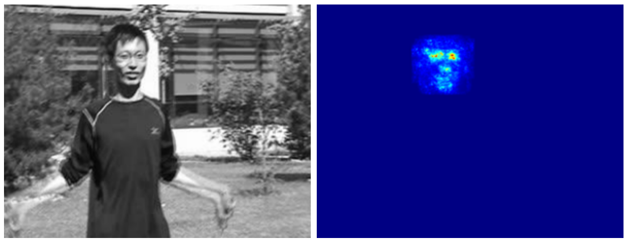

Figure 2. An example of target-specific saliency map. The face of a person in left image is being tracked. The target-specific saliency map reveals meaningful spatial configuration of the target, such as eyes, a nose and lips.

where $\phi_k(\mathbf{x}_i)$ denotes the k-th entry of $\phi(\mathbf{x}_i)$. Then the gradient of target-specific feature $\phi^+(\mathbf{x}_i)$ with respect to the image observation is obtained by

$$g(\mathbf{x}_i) = \frac{\partial \phi^+(\mathbf{x}_i)}{\partial \mathbf{z}_i}. \tag{6}$$

Since the gradient is computed only for the target-specific information $\phi^+(\mathbf{x}_i)$, pixels to distinguish the target from background would have high values in $g(\mathbf{x}_i)$.

The target-specific saliency map M is obtained by aggregating $g(\mathbf{x}_i)$ of samples with positive SVM scores in image space. As $g(\mathbf{x}_i)$ is defined over sample observation $\mathbf{z}_i$, we first project it to image space and zero-pad outside of $\mathbf{z}_i$; we denote the result by G_i afterwards. Then, the target-specific saliency map is obtained by taking the pixelwise maximum magnitude of the gradient maps G_i's corresponding to positive examples, which is given by

$$M(p) = \max_i |G_i(p)|, \quad \forall i \in \{j | \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_j) + b > 0\}, \tag{7}$$

where p denotes pixel location. We suppress erroneous activations from background by considering only positive examples when aggregating sample gradient maps. An example of target-specific saliency map is illustrated in Figure 2, where strong activations typically come from target areas and spatial layouts of target are exposed clearly.

3.3. Target Localization with Saliency Map

Given the target-specific saliency map at frame t denoted by M_t, the next step of our algorithm is to locate the target through sequential Bayesian filtering. Let $\mathbf{x}_t$ and M_t denote the state and observation variables at current frame t, respectively, where saliency map is used for measurement. The posterior of the target state $p(\mathbf{x}_t | M_{1:t})$ is given by

$$p(\mathbf{x}_t | M_{1:t}) \propto p(M_t | \mathbf{x}_t) p(\mathbf{x}_t | M_{1:t-1}), \tag{8}$$

where $p(\mathbf{x}_t | M_{1:t-1})$ denotes the prior distribution predicted from the previous time step, and $p(M_t | \mathbf{x}_t)$ means observation likelihood.

The prior distribution $p(\mathbf{x}_t|M_{1:t-1})$ of target state at the current time step is estimated from the posterior at the previous frame through prediction, which is given by

$$p(\mathbf{x}_t|M_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|M_{1:t-1})d\mathbf{x}_{t-1}, \quad (9)$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ denotes a state transition model. Target dynamics between two consecutive frames is given by a simple linear equation as

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{d}_t + \boldsymbol{\varepsilon}_t, \quad (10)$$

where $\mathbf{d}_t$ denotes a displacement of target location, and $\boldsymbol{\varepsilon}_t$ indicates a Gaussian noise. Both $\mathbf{d}_t$ and $\boldsymbol{\varepsilon}_t$ are unknown before tracking in general, but is estimated from the samples classified as target by our online SVM in our case. Specifically, $\mathbf{d}_t$ and $\boldsymbol{\varepsilon}_t$ are given respectively by

$$\mathbf{d}_t = \boldsymbol{\mu}_t - \mathbf{x}^*_{t-1}, \qquad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Sigma_t), \quad (11)$$

where $\mathbf{x}^*_{t-1}$ denotes the target location at the previous frame, and $\boldsymbol{\mu}_t$ and Σ_t indicate mean and variance of locations of positive samples at the current frame, respectively. From Eq. (10) and (11), the transition model for prediction is derived as follows:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t - \mathbf{x}_{t-1}; \mathbf{d}_t, \Sigma_t). \quad (12)$$

Since the transition model is linear with Gaussian noise, computation of the prior in Eq. (9) can be performed efficiently by transforming the posterior $p(\mathbf{x}_{t-1}|M_{1:t-1})$ at the previous step by $\mathbf{d}_t$ and applying Gaussian smoothing with covariance Σ_t.

The measurement density function $p(M_t|\mathbf{x}_t)$ represents the likelihood in the state space, which is typically obtained by computing the similarity between the appearance models of target and candidates. In our case, we utilize M_t, target-specific saliency map at frame t, for observation to compute the likelihood of each target state. Note that pixel-wise intensity and its spatial configuration in the saliency map provide useful information for target localization. At frame t, we construct the target appearance model H_t given the previous saliency maps $M_{1:t-1}$ in a generative way. Let $M_k(\mathbf{x}^*_k)$ denote the target filter at frame k, which is obtained by extracting the subregion in M_k at the location corresponding to the optimal target bounding box given by $\mathbf{x}^*_k$. The appearance model H_t is constructed by aggregating the recent target filters as follows:

$$H_t = \frac{1}{m}\sum_{k=t-m}^{t-1} M_k(\mathbf{x}^*_k), \quad (13)$$

where m is a constant for the number of target filters to be used for model construction. The main idea behind Eq. (13) is that the local saliency map nearby the optimal target location in a frame plays a role as a filter to identify the target within the saliency map in the subsequent frames. Since the target filter is computed based on m recent filters, we need to store the m filters to update the target filter. Therefore, given the appearance model defined in Eq. (13), the observation likelihood $p(M_t|\mathbf{x}_t)$ is computed by simple convolution between H_t and M_t by

$$p(M_t|\mathbf{x}_t) \propto H_t \otimes M_t(\mathbf{x}_t), \quad (14)$$

where $\otimes$ denotes convolution operator. This is similar to the procedure in object detection, e.g., (Felzenszwalb et al., 2010), where the filter is constructed from features to represent the object category and applied to the feature map to localize the object by convolution.

Given the prior in Eq. (9) and the likelihood in Eq. (14), the target posterior at the current frame is computed simply by applying Eq. (8). Once the target posterior is obtained, the optimal target state is given by solving the maximum a posteriori problem as

$$\mathbf{x}^*_t = \arg\max_{\mathbf{x}} p(\mathbf{x}_t|M_{1:t}). \quad (15)$$

Once tracking at frame t is completed, we update the classifier based on $\mathbf{x}^*_t$, which is discussed next.

3.4. Discriminative Model Update by Online SVM

We employ an online SVM to learn a discriminative model of target. Our SVM can be regarded as a fully-connected layer with a single node but provides a fast and exact solution in a single pass to learn a model incrementally.

Given a set of samples with associated labels, $\{(\mathbf{x}'_i, y'_i)\}$, obtained from the current tracking results, we hope to update a weight vector $\mathbf{w}$ of SVM. The label y'_i of a new example $\mathbf{x}'_i$ is given by

$$y'_i = \begin{cases} +1, & \text{if} \quad \mathbf{x}'_i = \mathbf{x}^*_t \\ -1, & \text{if} \quad \frac{\mathsf{BB}(\mathbf{x}^*_t)\cap\mathsf{BB}(\mathbf{x}'_i)}{\mathsf{BB}(\mathbf{x}^*_t)\cup\mathsf{BB}(\mathbf{x}'_i)} < \delta \end{cases}, \quad (16)$$

where $\mathsf{BB}(\mathbf{x})$ denotes the bounding box corresponding to the given state $\mathbf{x}$ and δ denotes a pre-defined threshold. Note that the examples with the bounding box overlap ratios larger than δ are not included in the training set for our online learning to avoid drift problem.

Before discussing online SVM, we briefly review the optimization procedure of an offline learning algorithm. Given training examples $\{(\mathbf{x}_i, y_i)\}$, the offline SVM learns a weight vector $\mathbf{w} = (w_1, \ldots, w_n)^\top$ by solving a quadratic convex optimization problem. The dual form of SVM objective function is given by

$$\min_{0\le a_i \le C} : W = \frac{1}{2}\sum_{i,j} a_i Q_{ij} a_j - \sum_i a_i + b\sum_i y_i a_i, \quad (17)$$

where $\{a_i\}$ are Largrange multipliers, b is bias, and $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. In our tracking algorithm, the kernel function is defined by the inner product between two CNN features, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. In online tracking, it is not straightforward for conventional QP solvers to handle the optimization problem in Eq. (17) as training data are given sequentially, not at once. Incremental SVM (Diehl & Cauwenberghs, 2003; Cauwenberghs & Poggio, 2000) is an algorithm designed to learn SVMs in such cases. The key idea of the algorithm is to retain KKT conditions on all the existing examples while updating model with a new example, so that it guarantees an exact solution at each increment of dataset. Specifically, KKT conditions are the first-order necessary conditions for the optimal solution of Eq. (17), which are given by

$$\frac{\partial W}{\partial a_i} = \sum_j Q_{ij} a_j + y_i b - 1 \begin{cases} \geq 0, \text{ if } a_i = 0 \\ = 0, \text{ if } 0 < a_i < C \\ \leq 0, \text{ if } a_i = C, \end{cases} \tag{18}$$

$$\frac{\partial W}{\partial b} = \sum_j y_j a_j = 0, \tag{19}$$

where $\frac{\partial W}{\partial a_i}$ is related to the margin of the i-th example that is denoted by m_i afterwards. By the conditions in Eq. (18), each training example belongs to one of the following three categories: E_1 for support vectors lying on the margin ($m_i = 0$), E_2 for support vectors inside the margin ($m_i < 0$), and E_3 for non-support vectors.

Given the k-th example, incremental SVM estimates its Lagrangian multiplier a_k while retaining the KKT conditions on all the existing $k-1$ training examples. In a nutshell, a_k is initialized to 0 and updated by increasing its value over iterations. In each iteration, the algorithm estimates the largest possible increment Δa_k that guarantees KKT conditions on the existing examples, and updates a_k and existing model parameters with Δa_k. This iterative procedure will stop when the k-th example becomes a support vector or at least one existing example changes its membership across E_1, E_2, and E_3. We can generalize this online update procedure easily when multiple examples are provided as new training data. With the new and updated Lagrangian multipliers, the weight vector $\mathbf{w}$ is given by

$$\mathbf{w} = \sum_{i \in E_1 \cup E_2} a_i y_i \phi(\mathbf{x}_i). \tag{20}$$

For efficiency, we maintain only a fixed number of support vectors with smallest margins during tracking. We ask to refer to (Diehl & Cauwenberghs, 2003; Cauwenberghs & Poggio, 2000) for more details. Also, note that any other methods for online SVM learning, such as LaSVM (Bordes et al., 2005) and LaRank (Bordes et al., 2007), can also be adopted in our framework.

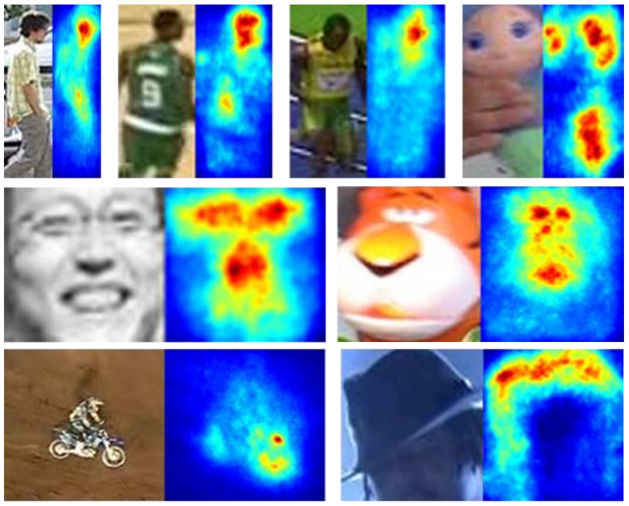

Figure 3. Examples of generative models learned by our algorithm. In each example, the left and right image indicate the target and learned model, respectively.

4. Experiments

This section describes our implementation details and experimental setting. The effectiveness of our tracking algorithm is then demonstrated by quantitative and qualitative analysis on a large number of benchmark sequences.

4.1. Implementation Details

For feature extraction, we adopt the R-CNN model built upon the Caffe library (Jia, 2013). The CNN takes an image from sample bounding box, which is resized to 227×227, and outputs a 4096-dimensional vector from its first fully-connected (fc_6) layer as a feature vector corresponding to the sample. To generate target candidates in each frame, we draw $N(=120)$ samples from a normal distribution as $\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}^*_{t-1}, \sqrt{wh}/2)$, where w and h denote the width and height of target, respectively. The SVM classifier and the generative model are updated only if at least one example is classified as positive by the SVM. When generating training examples for our SVM, the threshold δ in Eq. (16) is set to 0.3. The number of observations m used to build generative model in Eq. (13) is set to 30. To obtain segmentation mask, we employ *GrabCut* (Rother et al., 2004), where pixels that have saliency value larger than 70% of maximum saliency are used as foreground seeds, and background pixels around the target bounding box up to 50 pixels margin are used as background seeds. All parameters are fixed for all sequences throughout our experiment.

4.2. Analysis of Generative Appearance Models

The generative model H_t is used to localize the target using the target-specific saliency map. As described earlier, the target-specific saliency map shows high responses around

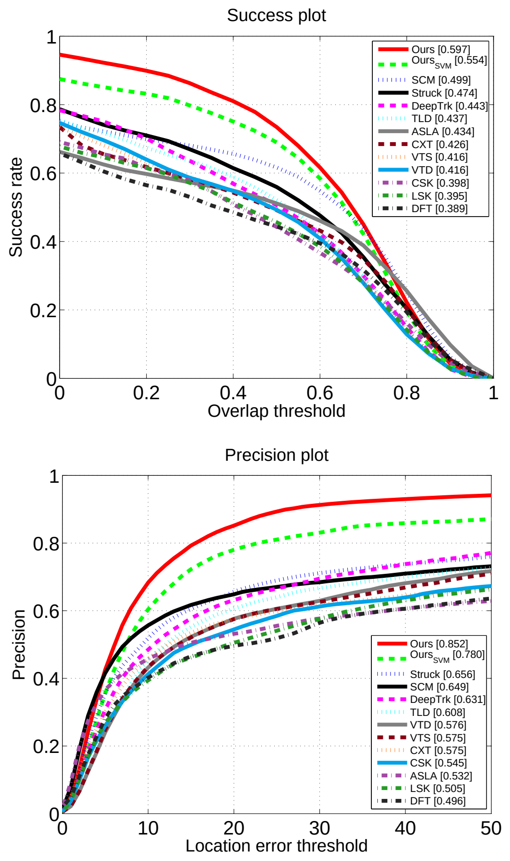

Figure 4. Average success plot (top) and precision plot (bottom) over 50 benchmark sequences.

discriminative target regions; our generative model exploits such property and is constructed using the saliency maps in the previous frames. Figure 3 illustrates examples of the learned generative models in several sequences. Generally, the model successfully captures parts and shape of an object, which are useful to discriminate the target from background. More importantly, the distribution of responses within the model reveals the spatial configuration of the target, which provides a strong cue for precise localization. This can be clearly observed in examples of face and doll, where the scores from the areas of eyes and nose can be used to localize the target. When target is not rigid (e.g., person), we observe that the model has stronger responses on less deformable parts of the target (e.g., head) and localization relies more on the stable parts consequently.

4.3. Evaluation

Dataset and compared algorithms To evaluate the performance, we employ all 50 sequences from the recently released tracking benchmark dataset (Wu et al., 2013). The sequences in the dataset involve various tracking challenges such as illumination variation, deformation, motion blur, background clutter, etc. We compared our method with top 10 trackers in (Wu et al., 2013), which include SCM (Zhong et al., 2012), Struck (Hare et al., 2011), TLD (Kalal et al., 2012), ASLA (Jia et al., 2012), CXT (Dinh et al., 2011), VTD (Kwon & Lee, 2010), VTS (Kwon & Lee, 2011), CSK (Henriques et al., 2012), LSK (Liu et al., 2011) and DFT (Sevilla-Lara & Learned-Miller, 2012), and DeepTrk (Li et al., 2014b). We used the reported results in (Wu et al., 2013) or available source code to reproduce the results.

Evaluation methodology We follow the evaluation protocols in (Wu et al., 2013), where the performance of trackers are measured based on two different metrics: success rate and precision. In both metrics, the ratio of successfully tracked frames is measured by a set of thresholds, where bounding box overlap ratio and center location error are employed in success rate and precision plot, respectively. We rank the tracking algorithms based on Area Under Curve (AUC) for success rate plot and center location error threshold of 20 pixels for precision plot.

Quantitative Results We evaluate our method quantitatively and make a comparative study with other methods in all the 50 benchmark sequences; the results are summarized in Figure 4. For bounding box tracking, our method outperforms all other trackers in terms of both success rate and precision with substantial margins. It is probably because the CNN features are more effective to represent high-level concept of target than hand-crafted ones although the network is trained offline for other purpose. We also compare our full algorithm with its reduced version denoted by Ours$_{\text{SVM}}$, which is a tracking-by-detection method based only on SVM scores. Our full algorithm achieves non-trivial performance improvement over the reduced version, which shows that our generative model based on target-specific saliency map is useful to localize target in general. The segmentation accuracy of our method is evaluated on 9 sequences in the benchmark dataset[1]. When the segmentation ground-truth is used, the success rate of our target segmentation is 0.598, which is significantly higher than all other methods including our bounding box trackers (TLD:0.315, Struck: 0.280, SCM: 0.272, Ours: 0.456).

To gain more insight about the proposed algorithm, we present bounding box tracker performances for individual attributes provided by the benchmark dataset in Table 1 and 2, where the numbers next to the attributes indicate the number of sequences involving the corresponding attribute. As illustrated in the tables, our algorithm consistently outperforms others in almost all challenges, and our

[1]Since accurate segmentation annotation is labor intensive and time consuming, we selected a subset of sequences for evaluation: *Bolt, Coke, Couple, Jogging, MotorRolling, MountainBike, Walking, Walking2*, and *Woman* sequences.

Table 1. Average success rate scores on individual attributes. Red: best, blue: second best.

	DFT	LSK	CSK	VTS	VTD	CXT	ASLA	TLD	Struck	SCM	Ours_{SVM}	Ours
Illumination variation (25)	0.383	0.371	0.369	0.429	0.420	0.368	0.429	0.399	0.428	0.473	0.522	0.556
Out-of-plane rotation (39)	0.387	0.400	0.386	0.425	0.434	0.418	0.422	0.420	0.432	0.470	0.524	0.582
Scale variation (28)	0.329	0.373	0.350	0.400	0.405	0.389	0.452	0.421	0.425	0.518	0.456	0.513
Occlusion (29)	0.381	0.409	0.365	0.398	0.403	0.372	0.376	0.402	0.413	0.487	0.539	0.563
Deformation (19)	0.439	0.377	0.343	0.368	0.377	0.324	0.372	0.378	0.393	0.448	0.623	0.640
Motion blur (12)	0.333	0.302	0.305	0.304	0.309	0.369	0.258	0.404	0.433	0.298	0.572	0.565
Fast motion (17)	0.320	0.328	0.316	0.300	0.302	0.388	0.247	0.417	0.462	0.296	0.545	0.545
In-plane rotation (31)	0.365	0.411	0.399	0.416	0.430	0.452	0.425	0.416	0.444	0.458	0.501	0.571
Out of view (6)	0.351	0.430	0.349	0.443	0.446	0.427	0.312	0.457	0.459	0.361	0.592	0.571
Background clutter (21)	0.407	0.388	0.421	0.428	0.425	0.338	0.408	0.345	0.458	0.450	0.519	0.593
Low resolution (4)	0.200	0.235	0.350	0.168	0.177	0.312	0.157	0.309	0.372	0.279	0.438	0.461
Weighted average	0.389	0.395	0.398	0.416	0.416	0.426	0.434	0.437	0.474	0.499	0.554	0.597

Table 2. Average precision scores on individual attributes. Red: best, blue: second best.

	DFT	LSK	CSK	VTS	VTD	CXT	ASLA	TLD	Struck	SCM	Ours_{SVM}	Ours
Illumination variation (25)	0.475	0.449	0.481	0.573	0.557	0.501	0.517	0.537	0.558	0.594	0.725	0.780
Out-of-plane rotation (39)	0.497	0.525	0.540	0.604	0.620	0.574	0.518	0.596	0.597	0.618	0.745	0.832
Scale variation (28)	0.441	0.480	0.503	0.582	0.597	0.550	0.552	0.606	0.639	0.672	0.679	0.827
Occlusion (29)	0.481	0.534	0.500	0.534	0.545	0.491	0.460	0.563	0.564	0.640	0.734	0.770
Deformation (19)	0.537	0.481	0.476	0.487	0.501	0.422	0.445	0.512	0.521	0.586	0.870	0.858
Motion blur (12)	0.383	0.324	0.342	0.375	0.375	0.509	0.278	0.518	0.551	0.339	0.764	0.745
Fast motion (17)	0.373	0.375	0.381	0.353	0.352	0.515	0.253	0.551	0.604	0.333	0.735	0.723
In-plane rotation (31)	0.469	0.534	0.547	0.579	0.599	0.610	0.511	0.584	0.617	0.597	0.720	0.836
Out of view (6)	0.391	0.515	0.379	0.455	0.462	0.510	0.333	0.576	0.539	0.429	0.744	0.687
Background clutter (21)	0.507	0.504	0.585	0.578	0.571	0.443	0.496	0.428	0.585	0.578	0.716	0.789
Low resolution (4)	0.211	0.304	0.411	0.187	0.168	0.371	0.156	0.349	0.545	0.305	0.536	0.705
Weighted average	0.496	0.505	0.545	0.575	0.576	0.575	0.532	0.608	0.656	0.649	0.780	0.852

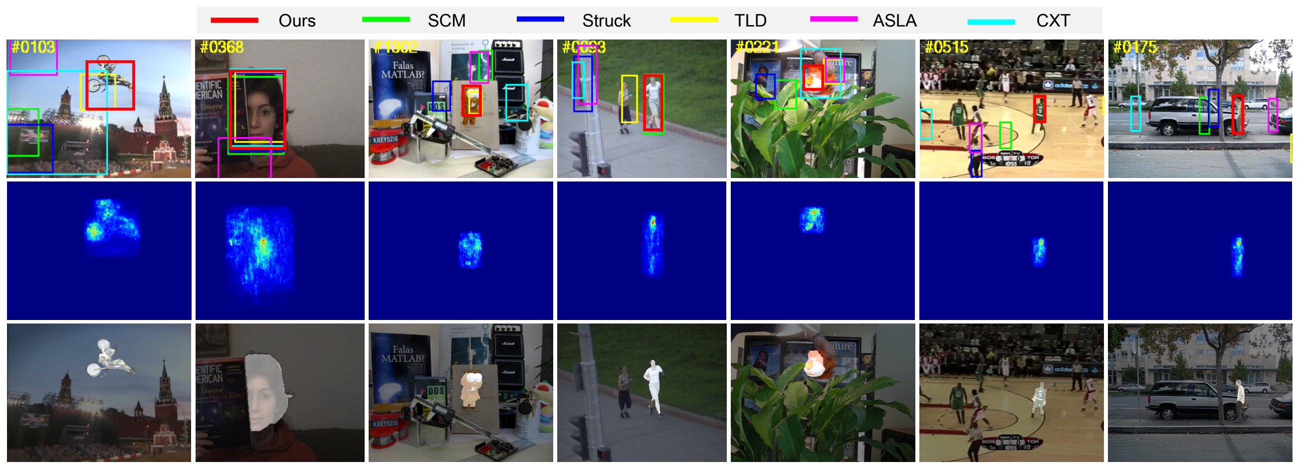

Figure 5. Qualitative results for *MotorRolling*, *FaceOcc1*, *Lemming*, *Jogging*, *Tiger*, *Basketball* and *David3* sequences. (Row1) Comparisons to other trackers. (Row2) Target-specific saliency maps. (Row3) Segmentation by *GrabCut* with target-specific saliency maps.

full algorithm is generally better than its reduced version.

Qualitative Results We present the results of several sequences in Figure 5, where original frames with tracking results, target-specific saliency maps, and segmentation results are illustrated. We can observe that our algorithm demonstrates outstanding performance qualitatively.

5. Conclusion

We proposed a novel visual tracking algorithm based on pre-trained CNN, where outputs from the last convolutional layer of the CNN are employed as generic feature descriptors of objects, and discriminative appearance models are learned online using an online SVM. With CNN features and learned discriminative model, we compute the target-specific saliency map by back-projection, which highlights the discriminative target regions in spatial domain. Tracking is performed by sequential Bayesian filtering with the target-specific saliency map as observation. The proposed algorithm achieves substantial performance gain over the existing state-of-the-art trackers and shows the capability for target segmentation.

Acknowledgments

This work was supported by the ICT R&D program of MSIP/IITP [B0101-15-0552, Novel computer vision and machine learning technology with the ability to predict and forecast; B0101-15-0307, Basic software research in human-level lifelong machine learning (Machine Learning Center)].

References

Babenko, Boris, Yang, Ming-Hsuan, and Belongie, Serge. Robust object tracking with online multiple instance learning. *TPAMI*, 33, 2011.

Bao, Chenglong, Wu, Yi, Ling, Haibin, and Ji, Hui. Real time robust l1 tracker using accelerated proximal gradient approach. In *CVPR*, 2012.

Berg, Alex, Deng, Jia, and Fei-Fei, L. Large scale visual recognition challenge (ILSVRC). `http://www.image-net.org/challenges/LSVRC/2012/`, 2012.

Bordes, Antoine, Ertekin, Seyda, Weston, Jason, and Bottou, Léon. Fast kernel classifiers with online and active learning. *JMLR*, 6, 2005.

Bordes, Antoine, Bottou, Léon, Gallinari, Patrick, and Weston, Jason. Solving multiclass support vector machines with larank. In *ICML*, 2007.

Cauwenberghs, Gert and Poggio, Tomaso. Incremental and decremental support vector machine learning. In *NIPS*, 2000.

Diehl, C.P. and Cauwenberghs, G. SVM incremental learning, adaptation and optimization. In *Proceedings of the International Joint Conference on Neural Networks*, 2003.

Dinh, Thang Ba, Vo, Nam, and Medioni, G. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011.

Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Fan, Jialue, Xu, Wei, Wu, Ying, and Gong, Yihong. Human tracking using convolutional neural networks. *Neural Networks*, 21, 2010.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *TPAMI*, 32, 2010.

Gall, J., Yao, A., Razavi, N., Gool, L. Van, and Lempitsky, V. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 33, 2011.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Grabner, H., Grabner, M., and Bischof, H. Real-time tracking via on-line boosting. In *BMVC*, 2006.

Han, B., Comaniciu, D., Zhu, Y., and Davis, L. S. Sequential kernel density approximation and its application to real-time visual tracking. *TPAMI*, 30, 2008.

Hare, S., Saffari, A., and Torr, P. H S. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

Hariharan, Bharath, Arbeláez, Pablo, Girshick, Ross, and Malik, Jitendra. Simultaneous detection and segmentation. In *ECCV*, 2014.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

Henriques, Joao F., Caseiro, Rui, Martins, Pedro, and Batista, Jorge. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

Jia, Xu, Lu, Huchuan, and Yang, Ming-Hsuan. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012.

Jia, Y. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

Kalal, Zdenek, Mikolajczyk, Krystian, and Matas, Jiri. Tracking-Learning-Detection. *TPAMI*, 2012.

Karayev, Sergey, Trentacoste, Matthew, Han, Helen, Agarwala, Aseem, Darrell, Trevor, Hertzmann, Aaron, and Winnemoeller, Holger. Recognizing image style. In *BMVC*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

Kwon, Junseok and Lee, Kyoung-Mu. Visual tracking decomposition. In *CVPR*, 2010.

Kwon, Junseok and Lee, Kyoung Mu. Tracking by sampling trackers. In *ICCV*, 2011.

Li, H., Li, Y., and Porikli, F. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *BMVC*, 2014a.

Li, Hanxi, Li, Yi, and Porikli, Fatih. Robust online visual tracking with an single convolutional neural network. In *ACCV*, 2014b.

Liu, Baiyang, Huang, Junzhou, Yang, Lin, and Kulikowski, Casimir A. Robust tracking using local sparse appearance model and k-selection. In *CVPR*, 2011.

Mei, Xue and Ling, Haibin. Robust visual tracking using $l1$ minimization. In *ICCV*, 2009.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

Ross, D., Lim, J., and Yang, M.-H. Adaptive probabilistic visual tracking with incremental subspace update. In *ECCV*, 2004.

Rother, Carsten, Kolmogorov, Vladimir, and Blake, Andrew. ”grabcut”: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

Saffari, A., Godec, M., Pock, T., Leistner, C., and Bischof, H. Online multi-class LPBoost. In *CVPR*, 2010.

Schulter, Samuel, Leistner, Christian, Roth, Peter M., Gool, Luc Van, , and Bischof, Horst. Online hough-forests. In *BMVC*, 2011.

Sermanet, Pierre, Eigen, David, Zhang, Xiang, Mathieu, Michael, Fergus, Rob, and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

Sevilla-Lara, L. and Learned-Miller, E. Distribution fields for tracking. In *CVPR*, 2012.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.

Toshev, A. and Szegedy, C. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

Wang, Naiyan and Yeung, Dit-Yan. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

Wu, Yi, Lim, Jongwoo, and Yang, Ming-Hsuan. Online object tracking: A benchmark. In *CVPR*, 2013.

Zhang, Ning, Donahue, Jeff, Girshick, Ross, and Darrell, Trevor. Part-based R-CNNs for fine-grained category detection. In *ECCV*, 2014.

Zhang, Tianzhu, Ghanem, Bernard, Liu, Si, and Ahuja, Narendra. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012.

Zhong, Wei, Lu, Huchuan, and Yang, Ming-Hsuan. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012.