Object Tracking via Dual Linear Structured SVM and Explicit Feature Map

Jifeng Ning[1], Jimei Yang[2], Shaojie Jiang[1], Lei Zhang[3] and Ming-Hsuan Yang[4]
[1]College of Information Engineering, Northwest A&F University, China
[2]Adobe Research, USA
[3]Department of Computing, The Hong Kong Polytechnic University, China
[4]Electrical Engineering and Computer Science, University of California at Merced, USA
jf_ning@sina.com, jimyang@adobe.com, shaojiejiang@126.com
cslzhang@comp.polyu.edu.hk,mhyang@ucmerced.edu

Abstract

Structured support vector machine (SSVM) based methods have demonstrated encouraging performance in recent object tracking benchmarks. However, the complex and expensive optimization limits their deployment in real-world applications. In this paper, we present a simple yet efficient dual linear SSVM (DLSSVM) algorithm to enable fast learning and execution during tracking. By analyzing the dual variables, we propose a primal classifier update formula where the learning step size is computed in closed form. This online learning method significantly improves the robustness of the proposed linear SSVM with lower computational cost. Second, we approximate the intersection kernel for feature representations with an explicit feature map to further improve tracking performance. Finally, we extend the proposed DLSSVM tracker with multi-scale estimation to address the "drift" problem. Experimental results on large benchmark datasets with 50 and 100 video sequences show that the proposed DLSSVM tracking algorithm achieves state-of-the-art performance.

1. Introduction

Object tracking aims to estimate the locations of a target in an image sequence. It can be applied to numerous tasks such as human-computer interaction, traffic monitoring, action analysis and video surveillance [34, 5, 26]. The main issue of object tracking is the incapability to account for large appearance variations due to viewpoint changes, occlusions, deformations and fast motions.

Existing object tracking algorithms can be broadly categorized as either generative or discriminative. Generative tracking algorithms [6, 22, 16, 17, 24] typically learn an appearance model to represent a target and use the model to search for interesting regions in the next frame with minimal reconstruction error. Instead of constructing a model to represent the appearance of a target, discriminative approaches [1, 2, 30, 3, 23, 11, 8, 33] consider the tracking problem as a classification or regression problem of finding the decision boundary that best separates the target from the background. In recent years, the tracking-by-detection approach has attracted more attention due to its strength to deal with targets undergoing large appearance variations.

Numerous classification algorithms such as support vector machines [1], boosting [2, 30], multiple instance learning [3] and random forests [23, 11] have been used in recent tracking-by-detection methods. However, the goal of binary classifiers is not seamlessly aligned with the one of object trackers due to the structured output space of tracking. To overcome this problem, Hare et al. [8] propose a kernelized Structured SVM (Struck) for object tracking. The Struck method treats object tracking as a structured output prediction problem that admits a consistent target representation for both learning and detection. Especially, in a recent tracking benchmark studies [18, 31, 32], Struck [8] shows the state-of-the-art performance.

However, the high complexity of optimization and detection processes for Struck [8] with nonlinear kernels limits its usage of high dimension features. It is critical to tracking performance because object representation with high dimensional features can model the target better than low dimensional ones. For example, KCF [10] greatly outperforms its original version CSK [9] by only replacing the low dimensional image feature with high dimensional HOG feature. On the other hand, the primal SSVM can be learned efficiently with linear kernels, which is very useful for fast training and detection even if it uses high dimensional features to represent the target. However, existing sub-gradients methods [25, 21] are sensitive to the step size when applied to online tasks. Therefore, it is of great interest to design a proper SSVM tracking algorithm that can run sufficiently fast with high dimensional features.

In this work, we propose a simple but effective dual linear SSVM (DLSSVM) tracking algorithm to solve these problems. First, we formulate object tracking as a linear SSVM detection problem to enable fast model updates in its primal form. By analyzing the relationship between the dual and primal variables, we present a closed form solution to compute the step size of the model update, which is critical for the tracking performance of linear SSVMs. Second, to exploit nonlinear kernels while maintaining the linearity of the proposed SSVM, we approximate the nonlinear kernels with explicit feature maps. Third, to overcome the drifting problem caused by large scale changes, we extend the proposed tracker with multi-scale estimation. Experiments on benchmark datasets [31, 32] of 50 and 100 image sequences show that the DLSSVM tracker achieves the state-of-the-art performance.

The main contributions of this work are summarized as follows: First, a dual linear SSVM classifier is derived in closed form, which is faster to train and evaluate than non-linear classifiers and this important technique constitutes the basis of this work. Second, with explicit feature maps, the proposed DLSSVM tracker can exploit high dimensional linear features to better represent objects for visual tracking than non-linear SSVM methods in terms of speed and accuracy. Third, the multi-scale estimation further improves the performance by accounting for the large scale changes during tracking.

2. Preliminaries

In this section we briefly introduce the structured SVM formulation of the tracking-by-detection approach before presenting the proposed algorithm.

2.1. Tracking-by-Detection

The tracking-by-detection method learns an online classifier to distinguish a target from its local background. We review its main components and discuss the difference between traditional binary discriminative classifiers and structured SVMs. For the ease of illustration, we use the same notations as [8] in the following. Let p_t denote the object central location at frame t, y is a relative transform according to location p_t. We represent a new position by $\mathrm{p}_t \circ \mathrm{y}$ where $\circ$ is a transformation operator (e.g., displacement, Euclidean or affine transform). At the image location $\mathrm{p}_t \circ \mathrm{y}$, we extract image patches $\mathrm{x}_t^{\mathrm{p}_t \circ \mathrm{y}}$. Let $h(\cdot)$ be a learned classifier and r be the radius of a search space Y that contains all candidate locations.

Assume that we have the initial discriminative classifier according to the first frame. First, we crop out a set of image patches $\mathrm{x}_t^{\mathrm{p}_{t-1} \circ \mathrm{y}}$ from search space Y with radius $\|\mathrm{y}\| \leq r$ and compute feature vectors. Second, we use the discriminative classifier to update tracker location $\mathrm{y}^* = \arg\max_{\mathrm{y} \in \mathrm{Y}} h(\mathrm{x}_t{}^{\mathrm{p}_{t-1} \circ \mathrm{y}})$ and obtain the location $\mathrm{p}_t = \mathrm{p}_{t-1} \circ \mathrm{y}^*$ in the current frame. Third, members of the sample set $\mathrm{x}_t{}^{\mathrm{p}_t \circ \mathrm{Y}}$ are cropped out around the current tracking position p_t to update the discriminative classifier. For a binary classifier [2, 3, 38], the image patches $\mathrm{x}_t{}^{\mathrm{p}_t \circ \mathrm{Y}}$ are divided into two groups, whose labels are respectively +1 and -1. For the structured SVM based classifier [8], the label of sample x_t is structured, i.e., the candidate positions y of the target. Finally, the discriminative classifier is updated with the newly arrived samples.

2.2. Structured SVM

In structured prediction, the goal is to predict a structured output $\mathrm{y} \in \mathrm{Y}$ for a given input $\mathrm{x} \in X$ where y can be arbitrary output for different problems. In our tracker, y is defined as a bounding box. The feature vector $\Phi(\mathrm{x}, \mathrm{y})$ is a function defined over a pair of input and output (x, y) which encodes the relevant information. We learn a discriminative classifier with parameter w defined by

$$\mathrm{y}^* = \arg\max_{\mathrm{y} \in \mathrm{Y}} h(\mathrm{x},\mathrm{y},\mathrm{w}) \tag{1}$$

where $h(\mathrm{x},\mathrm{y},\mathrm{w}) = \mathrm{w}^\top \Phi(\mathrm{x},\mathrm{y})$ and w can be learned in a large-margin framework from a sample set of $\{(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_n, \mathrm{y}_n)\}$ by solving the following global optimization problem:

$$\begin{aligned} &\min_{\mathrm{w}} \frac{1}{2}\|\mathrm{w}\|^2 + c\sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \quad \forall i : \xi_i \geq 0 \\ &\qquad \forall i, \forall \mathrm{y} \neq \mathrm{y}_i : \langle \mathrm{w}, \Psi_i(\mathrm{y}) \rangle \geq L(\mathrm{y}_i, \mathrm{y}) - \xi_i \end{aligned} \tag{2}$$

where $\Psi_i(\mathrm{y}) = \Phi(\mathrm{x}_i, \mathrm{y}_i) - \Phi(\mathrm{x}_i, \mathrm{y})$ and $L(\mathrm{y}_i, \mathrm{y})$ denotes the task-dependent structured error of predicted output y instead of the observed output y_i. The slack variable ξ_i measures the surrogate loss for the i-th data point and c is the regularization parameter. The loss function expresses a finer distinction between y_i and y, which plays an important role in the structured SVM. Similar to the Struck method [8], we choose to base the loss function on the bounding box overlap rate

$$L(\mathrm{y}_i, \mathrm{y}) = 1 - s^o_{\mathrm{p}_t}(\mathrm{y}_i, \mathrm{y}) \tag{3}$$

where $s^\circ_{\mathrm{p}_t}(\mathrm{y}_i, \mathrm{y}) = \frac{(\mathrm{p}_t \circ \mathrm{y}_i) \cap (\mathrm{p}_t \circ \mathrm{y})}{(\mathrm{p}_t \circ \mathrm{y}_i) \cup (\mathrm{p}_t \circ \mathrm{y})}$. The Lagrange dual of the above n-slack formulation is given by

$$\min_{\alpha \geq 0} \; f(\alpha) := \frac{1}{2}\| \sum_{i, \mathrm{y} \neq \mathrm{y}_i} \alpha_{\mathrm{i,y}} \Psi_{\mathrm{i}}(\mathrm{y}) \|^2 - \sum_{i, \mathrm{y} \neq \mathrm{y}_i} L(\mathrm{y}_{\mathrm{i}}, \mathrm{y}) \alpha_{\mathrm{i,y}} \tag{4a}$$

$$\text{s.t.} \quad \forall i, \forall \mathrm{y} \neq \mathrm{y}_i : \alpha_{i,y} \geq 0 \tag{4b}$$

$$\forall i : \sum_{\mathrm{y} \neq \mathrm{y}_i} \alpha_{i,\mathrm{y}} \leq c \tag{4c}$$

In the dual structured SVM, the discriminative classifier can be defined as $\mathrm{w} = \sum_{i, \mathrm{y} \neq \mathrm{y}_i} \alpha_{i,\mathrm{y}} \Psi_i(\mathrm{y})$.

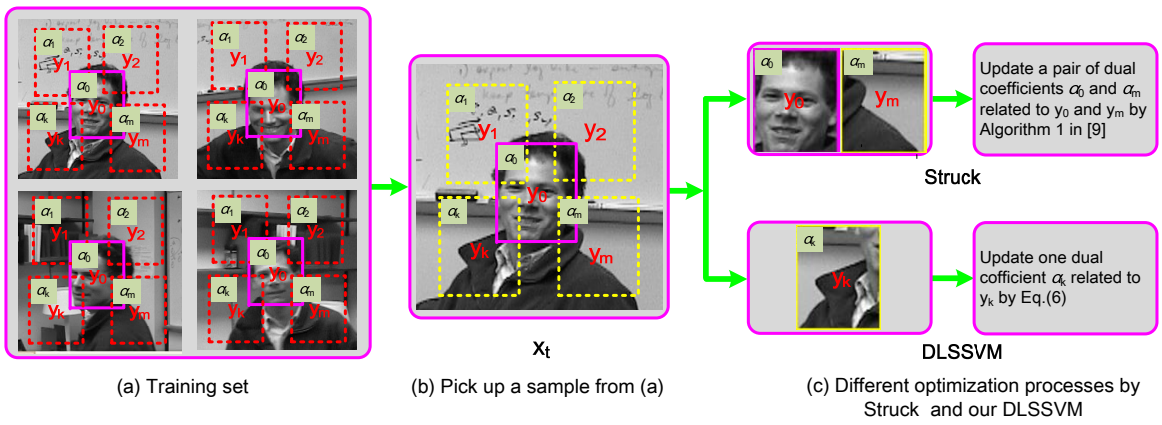

Figure 1. Comparing optimization processes for the Struck [8] and DLSSVM methods. (a) First, both SSVM trackers crop structured samples around the tracking result in each frame. Each structured output of each sample has a dual coefficient α. (b) Second, a selected sample is used to update its dual coefficients related to different structured outputs, i.e. support vectors. The Struck [8] and DLSSVM methods use different optimization schemes to update it. (c) Third, a pair of dual variables need to be carefully chosen and optimized in the Struck [8] method while only one dual variable is selected by a simple method and then updated in the proposed DLSSVM algorithm.

2.3. SSVM Based Tracking Analysis

Although dual SSVMs with non-linear kernels usually perform better than ones with linear kernels for tracking, the training and detection processes are more complex. As a result, if a non-linear kernel is used in the SSVM for object tracking, we cannot obtain the classifier parameter w explicitly so that the object detection can be only evaluated in the kernel space with a high computational cost.

The Struck method [8] uses low-dimensional features (192-dimensional Haar-like features) to represent target for reducing the computational cost. The sequential minimal optimization (SMO) [19] used by the Struck method [8] has a high computational cost as well because it requires finding a violation pair for each update and its convergence rate does not scale well with the size of the output space [39].

To alleviate the computational issue with non-linear kernels, we use a linear SSVM as the discriminative classifier w because it can be obtained explicitly. The use of linear kernels could accommodate high dimensional features for target representation and at the same time maintains a relatively low computational load for both training and detection. In terms of representation power, we note that the explicit feature map [27] can approximate non-linear kernels for non-linear decision efficiently and effectively.

3. Proposed Dual Linear SSVM Tracker

In this section, we first present our algorithm to efficiently solve a dual SSVM with linear kernels. Next, we use an unary representation [15] to approximate the intersection kernel for modeling object appearance, which improves the performance of the proposed DLSSVM tracker. Finally, we present the proposed tracking algorithm via our online dual linear SSVM optimization process. Figure 1 summarizes the differences between the Struck and DLSSVM methods.

3.1. Dual Linear SSVM Optimization

We present an online learning algorithm to train a dual linear SSVM for object tracking. We follow the basic dual coordinate descent (DCD) [20] optimization process for the dual SSVM and consider (4a) as a multivariate function with respect to dual coefficients $\alpha_{i,\mathrm{y}}$.

Optimization with closed form solution. In the DCD approach, the basic process is that in each iteration only one sample is optimized. For a sample k, the DCD method first selects one violated variable with maximum error as,

$$\mathrm{y}_k^* = \underset{\mathrm{y}\in\mathrm{Y}_k}{\operatorname{argmax}}\, L(\mathrm{y},\mathrm{y}_k) - \mathrm{w}^\top \Psi_k(\mathrm{y}) \tag{5}$$

Note that we keep the primal classifier w for efficient model evaluation during tracking.

To estimate $\alpha_{k,\mathrm{y}_k^*}$, we first compute the derivative of (4a) with respect to $\alpha_{k,\mathrm{y}_k^*}$ (which is related to structured output y_k^*) and set it to zero. As a result, the new coefficient $\alpha'_{k,\mathrm{y}_k^*}$ is given by

$$\alpha'_{k,\mathrm{y}_k^*} = \frac{L(\mathrm{y}_k,\mathrm{y}_k^*)}{\|\Psi_k(\mathrm{y}_k^*)\|^2} - \frac{\sum_{i,\mathrm{y}\neq\mathrm{y}_i}(\alpha_{i,\mathrm{y}}\Psi_{\mathrm{i}}^\top(\mathrm{y}) - \alpha_{k,\mathrm{y}_k^*}\Psi_k^\top(\mathrm{y}_k^*))\Psi_k(\mathrm{y}_k^*)}{\|\Psi_k(\mathrm{y}_k^*)\|^2}$$

According to $\mathrm{w} = \sum_{i,\mathrm{y}\neq\mathrm{y}_i}\alpha_{i,\mathrm{y}}\Psi_i(\mathrm{y})$, we obtain a simple $\alpha_{k,\mathrm{y}_k^*}$ update formula (6) for the above equation,

$$\alpha'_{k,\mathrm{y}_k^*} = \alpha_{k,\mathrm{y}_k^*} + \frac{L(\mathrm{y}_k,\mathrm{y}_k^*) - \mathrm{w}^\top\Psi_k(\mathrm{y}_k^*)}{\|\Psi_k(\mathrm{y}_k^*)\|^2} \tag{6}$$

With the constraint in (4c), we have $\alpha'_{k,\mathrm{y}^*} \in [0, c - \sum_{\mathrm{y}\neq\mathrm{y}^*} \alpha_{k,\mathrm{y}}]$. Therefore, the second term on the right hand side of (6), which defines the increment of the dual coefficient by

$$\gamma = \frac{L(\mathrm{y}_k, \mathrm{y}_k^*) - \mathrm{w}^\top \Psi_k(\mathrm{y}^*)}{\|\Psi_k(\mathrm{y}^*)\|^2} \tag{7}$$

is normalized as $\gamma \in [-\alpha_{k,\mathrm{y}^*}, c - \sum_{\mathrm{y}} \alpha_{k,\mathrm{y}}]$.

Note that in (5) and (6) we use a linear kernel to *explicitly* compute the primal parameters w. Compared to non-linear kernels, where w is *implicitly* represented by the sum of all dual coefficients multiplying kernel transformation of support vectors, the primal classifier only requires simple vector inner products which leads to much less complex training and detection. After optimizing one sample, we obtain the updated primal classifier immediately,

$$\mathrm{w} = \mathrm{w} + \gamma \Psi_k(\mathrm{y}^*) \tag{8}$$

In (8) the update of w is similar to the sub-gradient descent (SSG) [21] method [25]. However, the SSG method is sensitive to the step size γ. Our update step size is derived in closed form in the dual space. It takes advantage of the DCD optimization with linear kernels and offers fast convergence guarantee [20]. Therefore, after each iteration, we obtain immediately the explicit classifier, which is subsequently used for the next update.

After learning the discriminative classifier w, we carry out object detection for the frame at time t using learned w via the simple matrix operation defined by

$$\mathrm{y}^* = \underset{\mathrm{y}\in\mathrm{Y}}{\operatorname{argmax}}\ \mathrm{w}^\top \Psi_t(\mathrm{y}).$$

The structured output y^* with maximum response is considered as the object location.

The DCD optimization [20] used by this work is simpler than the SMO optimization technique [19] in the Struck tracker [8]. In our method we only pick up one violation variable each iteration and update its support vector coefficient while the SMO [19] method needs to carefully find a pair of violated variables and update their coefficients. Compared to Struck [8], the process to update and maintain support vectors is simpler as it needs to only optimize one dual coefficient at each step. The main difference between the Struck [8] and DLSSVM methods is shown in Figure 1.

Budget of support vectors. The number of support vectors in the SSVM increases gradually over time and a fixed amount is maintained for memory efficiency. In our method, the number of support vectors does not increase the complexity of the training process because it is only used to control the number of samples, which is different from the Struck method [8].

As each dual coefficient is relatively independent in our linear SSVM optimization, we remove the support vector with the smallest norm, which is irrelevant to other support vectors. When the number of support vectors in the SSVM detector exceeds the budget, we remove one according to the following formula,

$$\alpha^* = \underset{\alpha_{i,\mathrm{y}}\in\alpha}{\operatorname{argmin}} \quad \|\alpha_{i,\mathrm{y}} \Psi_i(\mathrm{x}_i, \mathrm{y})\|^2 \tag{9}$$

where $\alpha_{i,\mathrm{y}}$ is the coefficient of the support vector $\Psi_i(\mathrm{x}_i, \mathrm{y})$ of sample x_i.

3.2. Explicit Feature Map for Non-Linear Kernels

We employ an image kernel between pairs of the patches cropped from a frame x at location y for the proposed DLSSVM tracker,

$$K_{image}(\mathrm{x}, \mathrm{y}, \bar{\mathrm{x}}, \bar{\mathrm{y}}) = K(\mathrm{x}^{\mathrm{p}\circ\mathrm{y}}, \bar{\mathrm{x}}^{\bar{\mathrm{p}}\circ\bar{\mathrm{y}}}) \tag{10}$$

For each patch, we normalize it to about 400 pixels and employ the feature representation in the MEEM method [36] based on the CIE Lab color space. In addition, we apply the non-parametric local rank transform (LRT) [35] to the lightness channel to increase invariance to illumination change.

We denote z_i as the feature vector of one image patch consisting of Lab and LRT channels, and measure the similarity of two image patches using an intersection kernel.

$$K(\mathrm{x}^{\mathrm{p}\circ\mathrm{y}}, \bar{\mathrm{x}}^{\bar{\mathrm{p}}\circ\bar{\mathrm{y}}}) = K(\mathrm{z}_i, \mathrm{z}_j) = \sum_k \min(\mathrm{z}_i^k, \mathrm{z}_j^k), \tag{11}$$

where k is the k-th element of feature vector z_i. To obtain the primal classifier w in our DLSSVM formulation, we use the explicit feature map $\Psi(\mathrm{x}^{\mathrm{p}\circ\mathrm{y}})$ for the intersection kernel. As an additive kernel, the explicit feature map can be approximated by the unary representation [15].

Let N denote the number of discrete levels, $U(n)$ denote the unary representation of the integer n, e.g., $U(3) = \{1, 1, 1, 0, 0, 0\}$ when $N = 6$, and $R(\cdot)$ denote the rounding function. The unary representation of the feature z^k is defined by

$$\phi(\mathrm{z}^k) = \sqrt{\frac{1}{N}} U(R(N\mathrm{z}^k)) \tag{12}$$

Based on this unary representation [15], the intersection kernel can be approximated by

$$\sum_k \min(\mathrm{z}_i^k, \mathrm{z}_j^k) \approx \sum_k < \phi(\mathrm{z}_i^k), \phi(\mathrm{z}_i^k) > \tag{13}$$

such that $\Psi(\mathrm{x}^{\mathrm{p}\circ\mathrm{y}}) \approx [\phi(\mathrm{z}_i^1), \phi(\mathrm{z}_i^2), \ldots, \phi(\mathrm{z}_i^k), \ldots]$.

We set the quantization number $N = 4$ for color sequences. For grayscale sequences, since the color channels are not available, we set the quantization number $N = 8$ for more accurate approximation of the intersection kernel.

For the dimension of features, the original features of color image include four channels (Lab+LRT). Using unary

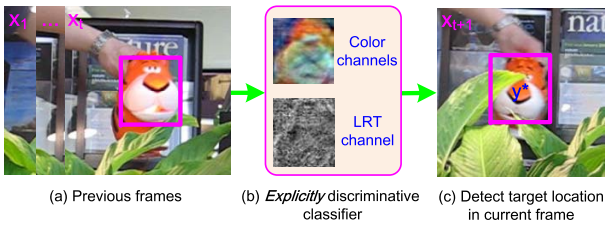

Figure 2. Main steps of the proposed DLSSVM tracker. (a) First, we crop structured samples around the tracking results of the previous frame. (b) Second, we update the discriminative classifier via dual linear SSVM optimization. Note that the *explicitly* discriminative classifier is very important for object tracking to rapidly train and detect. (c) We apply the updated classifier to detect the object in the current frame.

representation with the quantization number N=4, we have 400*4*4=6400 dimensional vectors. For gray image, its original features includes two channels (Gray+LRT). Using unary representation with the quantization number N=8, we also get 400*2*8=6400 dimensional vectors. In contrast, the Struck [8] with a non-linear Gaussian kernel only uses 192 dimensional feature vectors due to computational loads.

3.3. Multi-scale Estimation

It is difficult for tracking methods with a fixed scale representation to deal with target objects undergoing large scale changes. To alleviate the drifting problem caused by large scale changes, we extend the DLSSVM method with a multiple scale estimation. In this work, we use DLSSVMs at three different scales in parallel, and use the maximum responses as the tracking results.

3.4. Tracking Algorithm

We follow a common optimization strategy [20, 4] to implement our tracking algorithm. Figure 2 show the main steps of the proposed DLSSVM tracker, and the details are presented in Algorithm 1.

Especially, because both search region and discriminative classifier w actually belong to image features (Figure 2b), we can use the Fast Fourier Transform (FFT) algorithm to speed up the detection process. However, it is difficult for Struck [8] because it only gets implicitly discriminative classifier.

4. Experimental Results

We first discuss the experimental setup, dataset, and evaluation metrics, and then present two sets of experiments:

- Analysis of proposed DLSSVM and related SSVM trackers;
- Comparisons with state-of-the-art trackers.

Algorithm 1: DLSSVM tracking algorithm

input : Initial discriminative learner $\mathrm{w} = \mathbf{0}$ and initial object location p_0.
Output: Tracking result location p_i of each frame.

repeat
1. Estimate change in object location.
$\mathrm{y}_t = \arg\max_{\mathrm{y}\in\mathrm{Y}} \ \mathrm{w}^\top \mathrm{x}_t^{\mathrm{p}_{t-1}\circ\mathrm{y}}$
$\mathrm{p}_t = \mathrm{p}_{t-1} \circ \mathrm{y}_t$
2. Crop samples $\mathbf{X}_t = \mathbf{x}_t^{\mathbf{p}_t\circ\mathbf{y}}$ from current frame and append it to end of dataset.
3. Update DLSSVM discriminative classifier.
Get the number n of samples data.
For $j = 1 : n_1$
$i = n - \lfloor((j-1) * n/n_1)\rfloor$
Select a sample X_i from sample set X.
According to (5), select y^* from structured labels Y_i of sample X_i.
According to (6), update α_{i,y^*} corresponding to y^*.
According to (8), update w.
Maintain support vectors budget based on (9).
Get the number n of samples data.
For $p = 1 : n_2$
$i = n - \lfloor((p-1) * n/n_2)\rfloor$
Select a sample X_i from sample set X.
According to (5), select y^* from structured label Y_i of sample X_i with non-zero dual coefficients .
According to (6), update α_{i,y^*} corresponding to y^*.
According to (8), update w.
End For
End For
until *End of video sequences*;
Note: n_1 and n_2 are the numbers of iterations of exploring (external loop) and optimization (internal loop) [20] [4], and are fixed in all experiments.

More experimental results and videos can be found in the supplementary material. All the MATLAB source codes will be made available to the public.

4.1. Experimental Setup

Parameter Setting. For all sequences, we use fixed parameter values for fair evaluations. For the SSVM optimization (2), c is set to 100. We set the budget of support vectors to 100. The search radius for training and detection process is automatically determined by square root of the target area. The size of image patch is normalized to 400 pixels according to a trade-off between accuracy and speed. For scale estimation, we use the conservative scaling pool S={1 0.995,

Table 1. Characteristics of SSVM trackers. NU means no unary representation for the features

SSVM trackers	closed form solution	kernel type	feature type	feature dimensions	high dimension feature	non-linear decesion	discriminative classifier
SSG	no	linear	image feature	1600	yes	no	explicit
Struck	yes	Gaussian	Haar-like	192	no	yes	implicit
Linear-Struck-NU	yes	linear	image feature	1600	yes	no	explicit
Linear-Struck	yes	linear	image feature	6400	yes	yes	explicit
DLSSVM-NU	yes	linear	image feature	1600	yes	no	explicit
DLSSVM	yes	linear	image feature	6400	yes	yes	explicit

Table 2. Experimental comparisons of the proposed DLSSVM and related trackers with different parameters settings: B50, B100 and B500 mean the budgets of support vectors are 50, 100 and 500 respectively. The entries in **Bold red** indicate the best results and the ones in blue indicate the second best.

SSVM trackers	OPE		TRE		SRE		Mean FPS
	precision (20 pixels)	success (AUC)	precision (20 pixels)	success (AUC)	precision (20 pixels)	success (AUC)	
DLSSVM-NU	0.794	0.557	0.810	0.581	0.724	0.508	28.88
DLSSVM-B50	0.828	0.587	0.846	0.606	0.780	0.543	10.10
DLSSVM-B100	0.829	0.589	0.856	0.610	0.783	0.545	10.22
DLSSVM-B500	0.826	0.588	0.852	0.609	0.787	0.548	10.37
Scale-DLSSVM	**0.861**	**0.608**	**0.857**	**0.615**	**0.811**	**0.565**	5.40
SSG	0.608	0.443	0.665	0.486	0.584	0.424	**46.13**
Struck	0.656	0.474	0.707	0.514	0.634	0.449	0.90
Linear-Struck-NU	0.703	0.506	0.751	0.540	0.655	0.462	1.46
Linear-Struck	0.792	0.556	0.824	0.589	0.736	0.515	1.20

1.005}, which is similar to [13].

In our algorithm, we implement the training and detection process in MATLAB while the feature extraction step in C++ for runtime performance as in the Multi-Expert Entropy Minimization (MEEM) method [36]. It runs at 10 fps on a desktop computer with Intel i5-2400 CPU (3.10 GHz) and 6 GB memory.

Dataset. We evaluate the proposed DLSSVM algorithm on the TB50 [31] and TB100 [32] benchmark datasets. For detailed analysis, these sequences are annotated with 11 challenging attributes including illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of plane rotation (OPR), out-of-view (OV), background clutters (BC) and low resolution (LR).

Evaluation Protocol and Metrics. As suggested in [31], we evaluate the tracking algorithms using three protocols: one-pass evaluation (OPE), temporal robustness evaluation (TRE), and spatial robustness evaluation (SRE) using precision and success rates. We present the main findings in this manuscript and more results can be found in the supplementary material.

4.2. Analysis of Proposed DLSSVM and Related SSVM Trackers

We evaluate the DLSSVM method and the related trackers on the TB50 [31] dataset. Table 1 summarizes the characteristics of those SSVM trackers. Table 2 shows the experimental results of those related SSVM trackers including the run-time performance. The mean FPS (frames per second) is estimated on a long sequence *liquor* with 1741 frames.

we denote the DLSSVM tracker without the unary representation as DLSSVM-NU, and the method using 50, 100 and 500 support vectors as DLSSVM-B50 ,DLSSVM-B100 and DLSSVM-B500, respectively. The DLSSVM with multi-scale estimation is denoted as Scale-DLSSVM.

Analysis of DLSSVM tracker. Based on the results of the DLSSVM-NU and DLSSVM-B100 methods using the OPE, TRE and SRE protocols, it is clear that the explicit feature map with the unary representation plays an important role in robust object tracking. Overall, the DLSSVM tracker is insensitive to different numbers of support vectors (e.g., from 50 to 500). Furthermore, the Scale-DLSSVM method obtain better accuracies than the DLSSVM scheme at the expense of lower processing speed. In the following, the DLSSVM tracker is referred to the one with 100 support vectors for evaluations against other state-of-the-art tracking methods, unless specified otherwise.

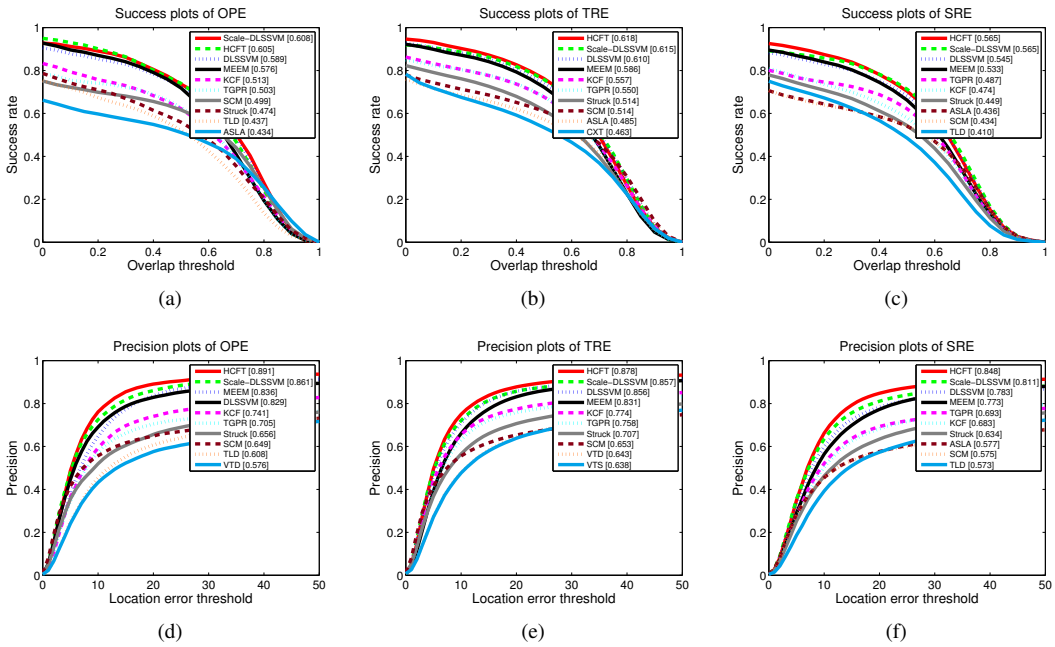

Figure 3. Average precision plot (top row) and success plot (bottom row) for the OPE, TRE and SRE on the TB50 [31] dataset. For presentation clarity, only the top ten trackers with respect to the ranking score are shown in each plot.

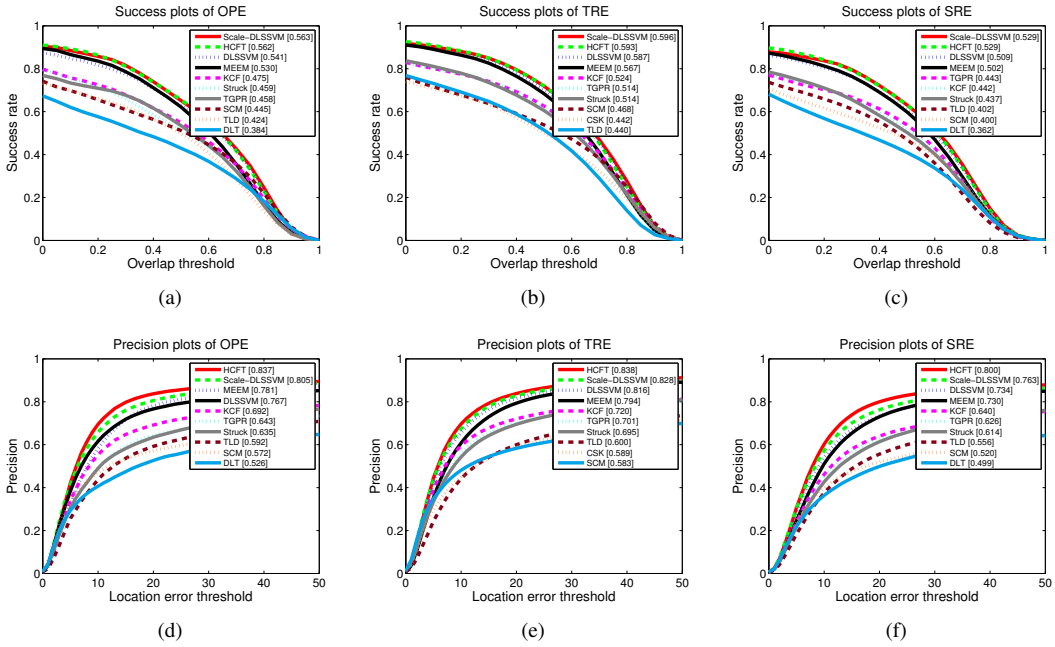

Figure 4. Average precision plot (top row) and success plot (bottom row) for the OPE, TRE and SRE on the TB100 [32] dataset. For presentation clarity, only the top ten trackers with respect to the ranking score are shown in each plot.

Comparisons with Other SSVM trackers. We first implement a linear SSVM tracker with the sub-gradient optimization method [25], and refer it as the SSG tracker (i.e., a baseline SSVM tracker). The learning rate to update classifiers is manually selected without using the closed form solution via (7) (i.e.,the step size in the four methods from the bottom of Table 2 is manually set). As shown in Table 2, although the SSG tracker uses the same high dimensional

features as the DLSSVM-NU tracker at a higher processing rate, the tracking accuracy in all three indices is lower than that of the Struck [8] method with a small margin.

Second, we implement the original Struck [8] method and a linear Struck approach in MATLAB for fair comparisons. Note that Haar-like feature used by Struck [8] is not proper for computing the explicit feature map of intersection kernel so we compare Struck and our method using our feature representations. In addition, we also evaluate the performance of the Struck method with a linear kernel with (Linear-Struck) and without (Linear-Struck-NU) using the explicit feature map.

For the Struck [8] method, we note that the linear Struck method with high dimensional features (Linear-Struck-NU) outperforms the original non-linear kernel Struck in terms of both accuracy (all metrics) and speed, which suggests that linear Struck with high dimensional feature is more proper than Struck with Gaussian kernel for visual tracking. The DLSSVM tracker (i.e., DLSSVM-B100) performs favorably against the Struck [8] and Linear-Struck methods in accuracy and speed. On the other hand, the experimental comparisons between the SSG, Linear-Struck and DLSSVM methods in Table 2 indicate that the linear SSVM classifier with the step size in closed form solution is crucial to robust object tracking. With simpler optimization process, the proposed DLSSVM tracker performs favorably against the Struck [8] method using non-linear and linear kernels in terms of accuracy and speed. It indicates that DCD optimization [20] used by our DLSSVM is better than SMO [19, 4] used by Struck [8] for visual tracking.

4.3. Comparisons with State-of-the-Art Trackers

We evaluate the DLSSVM and Scale-DLSSVM trackers against the state-of-the-art methods on the TB50 [31] and TB100 [32] datasets, where the results of 29 trackers are reported. In addition, we include 6 most recent trackers for performance evaluation. The HCFT [14] and DLT [28] methods are developed based on hierarchical features via deep learning. The STC [37] and KCF [10] schemes are based correlations filters. Furthermore, the TGPR [7] and MEEM [36] algorithms are developed based on regression and multiple trackers. The precision and success rates for the top ten trackers on the TB50 [31] and TB100 [32] datasets are presented in Figure 3 and Figure 4.

The KCF tracker [10] exploits circulant matrix computations and achieves high run-time speed. In addition, the recent method [13] shows that the performance of the KCF method can be further improved by a more effective representation based on color name attributes [12]. Overall, the proposed DLSSVM tracker with simple color and spatial features performs favorably over the KCF method in terms of accuracy using all metrics.

The MEEM [36] tracking method uses a mixture of experts based on entropy minimization where a linear SVM with twin prototypes [29] is used as the base tracker. The proposed DLSSVM tracker performs well against the MEEM method in most metrics except the OPE precision. In addition, the Scale-DLSSVM method with multi-scale estimation outperforms the MEEM tracker [36] in all metrics on both TB-50 and TB-100 datasets.

Compared to deep learning based methods, the proposed DLSSVM method performs favorably against the DLT [28] tracker on the TB50[31] and TB100 [32] datasets, and the Scale-DLSSVM algorithm performs comparably against the state-of-the-art HCFT [14] method which is based on both correlation filters and hierarchical convolutional features. We note that the proposed DLSSVM and Scale-DLSSVM methods only use simple image features while the HCFT method takes advantage of complex hierarchical convolutional features that requires offline training on a large dataset. These experimental results show that the dual linear optimization scheme used by the proposed SSVM trackers is effective and efficient for robust object tracking.

5. Conclusions

In this paper, we propose an efficient and effective SSVM formulation for robust object tracking via a dual linear SSVM optimization method and an explicit feature map. By using linear kernels, we can easily update the primal classifier and speed up the algorithm. With the dual SSVM formulation, we derive a closed form update scheme for the primal classifier which is critical for robust object tracking. We approximate intersection kernel with the explicit feature map to make non-linear decision by our linear SSVM classifier for better performance. The DLSSVM tracking method is further improved with multi-scale estimation to account for large scale changes. Experimental results show that the proposed DLSSVM tracker performs favorably against the state-of-the-art methods on large benchmark datasets.

Acknowledgment

J. Ning and S. Jiang are supported in part by the National Natural Science Foundation of China under Grants 61473235 and 31501228, the Fundamental Research Funds for the Central Universities under Grants QN2013055, QN2013062, Shaanxi Province Natural Science Foundation under Grant 2015JM3110 and Science Computing and Intelligent Information Processing of GuangXi Higher Education Key Laboratory under Grant GXSCIIP201406. J. Yang and M.-H. Yang are supported in part by the NSF CAREER Grant #1149783, NSF IIS Grant #1152576, and a gift from Adobe. L. Zhang is supported by Hong Kong RGC GRF grant (PolyU 152124/15E).

References

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

[2] S. Avidan. Ensemble tracking. *PAMI*, 29(2):61–271, 2007.

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

[4] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with larank. In *ICML*, 2007.

[5] K. Cannons. A review of visual tracking. *Dept. Comput. Sci. Eng., York Univ., Toronto, Canada, Tech. Rep. CSE-2008-07*, 2008.

[6] R. T. Collins. Mean-shift blob tracking through scale space. In *CVPR*, 2003.

[7] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014.

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

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

[10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 2014.

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

[12] F. Khan, R. Anwer, J. Weijer, A. Bagdanov, A. Lopez, and M. Felsberg. Coloring action recognition in still images. *IJCV*, 105(3):205–221, 2013.

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

[14] C. Ma, J. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[15] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

[16] X. Mei and H. Ling. Robust visual tracking using l1 minimization. In *ICCV*, 2009.

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

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

[19] J. Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methodssupport vector learning*, 3, 1999.

[20] D. Ramanan. Dual coordinate solvers for large-scale structural svms. In *http://arxiv.org/abs/1312.1743*, 2014.

[21] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. (online) subgradient methods for structured prediction. In *ICAIS*, 2007.

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

[23] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV*, 2009.

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

[25] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

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

[27] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 34(3):480–492, 2012.

[28] N. Wang and D. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[29] Z. Wang and S.Vucetic. online training on a budget of support vector machines using twin prototypes. In *SADM*, 2010.

[30] L. Wen, Z. Cai, Z. Lei, and S. Li. Online spatio-temporal structural context learning for visual tracking. In *ECCV*, 2012.

[31] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[32] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *PAMI*, 37(9):1834–1848, 2015.

[33] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. Hengel. Part-based visual tracking with online latent structural learning. In *CVPR*, 2013.

[34] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[35] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, 1994.

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

[37] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast tracking via dense spatio-temporal context learning. In *ECCV*, 2014.

[38] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, 2012.

[39] X. Zhang, A. Saha, and S. V. N. Vishwanathan. Accelerated training of max-margin markov networks with kernels. *Theoretical Computer Science*, 519:88–102, 2014.