Online Robust Non-negative Dictionary Learning for Visual Tracking

Naiyan Wang[†] **Jingdong Wang**[‡] **Dit-Yan Yeung**[†]

[†] Hong Kong University of Science and Technology [‡] Microsoft Research

winsty@gmail.com jingdw@microsoft.com dyyeung@cse.ust.hk

Abstract

This paper studies the visual tracking problem in video sequences and presents a novel robust sparse tracker under the particle filter framework. In particular, we propose an online robust non-negative dictionary learning algorithm for updating the object templates so that each learned template can capture a distinctive aspect of the tracked object. Another appealing property of this approach is that it can automatically detect and reject the occlusion and cluttered background in a principled way. In addition, we propose a new particle representation formulation using the Huber loss function. The advantage is that it can yield robust estimation without using trivial templates adopted by previous sparse trackers, leading to faster computation. We also reveal the equivalence between this new formulation and the previous one which uses trivial templates. The proposed tracker is empirically compared with state-of-the-art trackers on some challenging video sequences. Both quantitative and qualitative comparisons show that our proposed tracker is superior and more stable.

1. Introduction

Visual tracking or object tracking in video sequences is a major topic in computer vision and related fields. It has a wide range of real-world applications, including security and surveillance, autonomous vehicles, automatic traffic control, medical imaging, human-computer interaction, and many more. A typical setting of the problem is that an object identified, either manually or automatically, in the first frame of a video sequence is tracked in the subsequent frames by estimating its trajectory as it moves around. While the problem is easy to state, it is often challenging to build a robust object tracker due to various factors which include noise, occlusion, fast and abrupt object motion, illumination changes, and variations in pose and scale. The focus of this paper is on the widely-studied single object tracking problem.

Mei and Ling proposed the ℓ_1 *tracker* (L1T) [18] for robust visual tracking under the particle filter framework [5] based on the sparse coding technique [22]. L1T and its extensions [30, 29] showed promising results in various tests.

(a) davidin (b) bolt

Figure 1. Learned templates for two video sequences: davidin and bolt. For each sequence, the learned templates cover different appearances of the tracked object in the video while rejecting the cluttered background.

L1T describes the tracking target using basis vectors which consist of object templates and trivial templates, and reconstructs each candidate (particle) by a sparse linear combination of them. While object templates correspond to the normal appearance of objects, trivial templates are used to handle noise or occlusion. Specifically, each trivial template has only one nonzero element being one for a specific feature. If it is selected to represent the particle, it means that the corresponding feature is occluded. It is shown in [18] that a good target candidate should involve fewer trivial templates while keeping the reconstruction error low. We review the L1T in detail in section 3.2.

In this paper, we present an online robust non-negative dictionary learning algorithm for updating the object templates. The learned templates for two video sequences are shown in Fig. 1. We devise a novel online projected gradient descent method to solve the dictionary learning problem. In contrast to the *ad hoc* manner by replacing the least used template with the current tracking result as in [18, 30], our algorithm blends the past information and the current tracking result in a principled way. It can automatically detect and reject the occlusion and cluttered background, yielding robust object templates. Besides, we formulate the particle representation problem using the Huber loss function [10]. This formulation can yield robust estimation without using trivial templates and thus lead to significant reduction of the computational cost. Moreover, we also establish the equivalence between this new formulation and the L1T.

2. Related Work

Object tracking is an extensively studied research topic. For a comprehensive survey of this topic, we refer readers

to the survey paper [26] and a recent benchmark [24]. Here we only review some representative works which are categorized into two approaches for building object trackers, namely, generative and discriminative methods.

Generative trackers usually learn an appearance model to represent the object being tracked and make decision based on the reconstruction error. *Incremental visual tracking* (IVT) [20] is a recent method which learns the dynamic appearance of the tracked object via incremental principal component analysis (PCA). *Visual tracking decomposition* (VTD) [15] decomposes the tracking problem into several basic motion and observation models and extends the conventional particle filter framework to allow different basic models to interact. The method that is most closely related to our paper is L1T [18] which, as said above, assumes that the tracked object can be represented well by a sparse linear combination of object templates and trivial templates. Its drawback of having high computational cost has been alleviated by subsequent works [19, 4] to improve the tracking speed. More recently, Zhang *et al.* found that considering the underlying relationships between sampled particles could greatly improve the tracking performance and proposed the *multitask tracker* (MTT) [30] and *low rank sparse tracker* (LRST) [29]. In [11], Jia *et al.* proposed using alignment pooling in the sparse image representation to alleviate the drifting problem. For a survey of sparse coding based trackers, we refer readers to [28].

Unlike the generative approach, discriminative trackers formulate object tracking as a binary classification problem which considers the tracked object and the background as belonging to two different classes. One example is the *online AdaBoost* (OAB) tracker [7] which uses online AdaBoost to select features for tracking. The *multiple instance learning* (MIL) tracker [3] formulates object tracking as an online multiple instance learning problem which assumes that the samples belong to the positive or negative bags. The *P-N tracker* [12] utilizes structured unlabeled data and uses an online semi-supervised learning algorithm. A subsequent method called *Tracking-Learning-Detection* (TLD) [13] augments it by a detection phase, which has the advantage of recovering from failure even after the tracker has failed for an extended period of time. The Struck [8] learns a kernelized structured output support vector machine online. It ranks top in the recent benchmark [24]. The *compressive tracker* (CT) [27] utilizes a random sparse compressive matrix to perform efficient dimensionality reduction on the integral image. The resulting features are like generalized Haar features which are quite effective for classification.

Generally speaking, when there is less variability in the tracked object, generative trackers tend to yield more accurate results than discriminative trackers because generative methods typically use richer features. However, in more complicated environments, discriminative trackers are often more robust than generative trackers because discriminative trackers use negative samples to avoid the drifting problem. A natural attempt is to combine the two approaches to give a hybrid approach, as in [31].

Besides object trackers, some other techniques related to our proposed method are (online) *dictionary learning* and (robust) *non-negative matrix factorization* (NMF). Dictionary learning seeks to learn from data a dictionary which is an adaptive set of basis vectors or atoms, so that each data sample is represented by a sparse linear combination of the basis vectors. It has been found that dictionaries learned this way are more effective than off-the-shelf ones (e.g., based on Gabor filters or discrete cosine transform) for many vision applications such as denoising [1] and image classification [25]. Most dictionary learning methods are based on K-SVD [1] or online dictionary learning [17]. As for NMF [21], it is a common data analysis technique for high-dimensional data. Due to the non-negativity constraints, it tends to produce parts-based decomposition of images which facilitates human interpretation and yields superior performance. There are also some NMF variants, such as sparse NMF [9] and robust NMF [14, 6]. Online learning of basis vectors under the robust setting has also aroused a lot of interest, e.g., [23, 16].

3. Background

To facilitate the presentation of our model in the next section, we first briefly review in this section the particle filter approach for visual tracking and the ℓ_1 tracker (L1T).

3.1. Particle Filters for Visual Tracking

The particle filter approach [5], also known as a sequential Monte Carlo (SMC) method for importance sampling, is commonly used for visual tracking. Like a Kalman filter, a particle filter sequentially estimates the latent state variables of a dynamical system based on a sequence of observations. The main difference is that, unlike a Kalman filter, the latent state variables are not restricted to the Gaussian distribution, not even distribution of any parametric form. Let $\mathbf{s}^t$ and $\mathbf{y}^t$ denote the latent state and observation, respectively, at time t. A particle filter approximates the true posterior state distribution $p(\mathbf{s}^t \mid \mathbf{y}^{1:t})$ by a set of samples $\{\mathbf{s}_i^t\}_{i=1}^n$ (a.k.a. particles) with corresponding weights $\{w_i^t\}_{i=1}^n$ which sum to 1. For the state transition probability $q(\mathbf{s}^{t+1} \mid \mathbf{s}^{1:t}, \mathbf{y}^{1:t})$, it is often assumed to follow a first-order Markov process so that it can be simplified to $q(\mathbf{s}^{t+1} \mid \mathbf{s}^t)$. In this case, the weights are updated as $w_i^{t+1} = w_i^t\, p(\mathbf{y}^t \mid \mathbf{s}_i^t)$. In case the sum of weights of the particles before normalization is less than a prespecified threshold, resampling is needed by drawing n particles from the current particle set in proportion to their weights and then resetting their weights to $1/n$.

In the context of object tracking, the state $\mathbf{s}_i$ is often

characterized by six affine parameters which correspond to translation, scale, aspect ratio, rotation and skewness. Each dimension of $q(\mathbf{s}^{t+1} \mid \mathbf{s}^t)$ is modeled by an independent Gaussian distribution. The tracking result at each time step is taken to be the particle with the largest weight. A key issue in the particle filter approach is to formulate the observation likelihood $p(\mathbf{y}^t \mid \mathbf{s}_i^t)$. In general, it should reflect the similarity of a particle and the object templates while being robust against occlusion or appearance changes. We will discuss this issue in greater depth later in section 5.1.

The particle filter framework is popularly used for visual tracking due partly to its simplicity and effectiveness. First, as said before, this approach is more general than using Kalman filters because it is not restricted to the Gaussian distribution. Also, the accuracy of the approximation generally increases with the number of particles used. Moreover, instead of using point estimation which may lead to overfitting, the probability distribution of the latent state variables is approximated by a set of particles, making it possible for the tracker to recover from failure. An excellent tutorial on using particle filters for visual tracking can be found in [2].

3.2. The ℓ_1 Tracker (L1T)

In each frame, L1T first generates candidate particles based on the particle filter framework. Let $\mathbf{Y} \in \mathbb{R}^{m \times n}$ denote the particles with each of the n columns for one particle. We further let $\mathbf{U} \in \mathbb{R}^{m \times r}$ denote the object templates and $\mathbf{V} \in \mathbb{R}^{n \times r}$ and $\mathbf{V}_T \in \mathbb{R}^{n \times m}$ denote the coefficients for the object templates and trivial templates, respectively. For sparse coding of the particles, L1T solves the following optimization problem:

$$\begin{aligned} \min_{\mathbf{V},\mathbf{V}_T} \quad & \frac{1}{2}\left\|\mathbf{Y} - [\mathbf{U}\ \mathbf{I}_m]\begin{bmatrix}\mathbf{V}' \\ \mathbf{V}_T'\end{bmatrix}\right\|_F^2 + \lambda\|\mathbf{V}_T\|_1 + \gamma\|\mathbf{V}\|_1 \\ s.t. \quad & \mathbf{V} \geq 0, \end{aligned} \tag{1}$$

where $\mathbf{I}_m$ denotes the identity matrix of size $m \times m$ which corresponds to the trivial templates, $\|\mathbf{A}\|_1 = \sum_{ij} |a_{ij}|$ for the ℓ_1 norm, and $\|\mathbf{A}\|_F = (\sum_{ij} a_{ij}^2)^{1/2}$ for the Frobenius norm. Then the weight of each particle is set to be inversely proportional to the reconstruction error. In each frame, the particle with the smallest reconstruction error (and hence largest weight) is chosen as the tracking result.

To reflect the appearance changes of an object, L1T takes an adaptive approach in updating the object templates. It first maintains a weight for each template according to its usage in representing the tracking result. When the current template set cannot represent the tracking result well, the template with the smallest weight will be replaced by the current tracking result.

4. Our Tracker

We present our object tracker in this section. The tracker consists of two parts. The first part is robust sparse coding which represents each particle using the dictionary templates by solving an optimization problem that involves the Huber loss function. The second part is dictionary learning which updates the object templates over time.

4.1. Robust Particle Representation

In terms of particle representation, we solve the following robust sparse coding problem based on the Huber loss:

$$\begin{aligned} \min_{\mathbf{V}} \quad & f(\mathbf{V};\mathbf{U}) = \sum_i \sum_j \ell_\lambda(y_{ij} - \mathbf{u}_{i\cdot}'\mathbf{v}_{j\cdot}) + \gamma\|\mathbf{V}\|_1 \\ s.t. \quad & \mathbf{V} \geq 0, \end{aligned} \tag{2}$$

where y_{ij} is an element of $\mathbf{Y}$, $\mathbf{u}_{i\cdot}$ and $\mathbf{v}_{j\cdot}$ are column vectors for the ith row of $\mathbf{U}$ and jth row of $\mathbf{V}$, respectively, and $\ell_\lambda(\cdot)$ denotes the Huber loss function [10] with parameter λ, which is defined as

$$\ell_\lambda(r) = \begin{cases} \frac{1}{2}r^2 & |r| < \lambda \\ \lambda|r| - \frac{1}{2}\lambda^2 & \text{otherwise.} \end{cases} \tag{3}$$

The Huber loss function is favorable here since it grows more slowly than the l_2 norm as the residue increases, making it less insensitive to outliers. Moreover, since it is smooth around zero, it is more stable than the l_1 norm when there exists small but elementwise noise. It encourages each candidate particle to be approximated by a sparse combination of the object templates and yet it can still accommodate outliers and noise caused by occlusion and the like.

Optimization: To minimize $f(\mathbf{V};\mathbf{U})$ w.r.t. $\mathbf{V}$ subject to the non-negativity constraint $\mathbf{V} \geq 0$, we use the following update rule:

$$v_{jk}^{p+1} = v_{jk}^p \frac{\left[(\mathbf{W}^p \odot \mathbf{Y})'\mathbf{U}\right]_{jk}}{\left[\left(\mathbf{W}^p \odot (\mathbf{U}(\mathbf{V}^p)')\right)'\mathbf{U}\right]_{jk} + \gamma}, \tag{4}$$

where the superscript p denotes the pth iteration of the optimization procedure for $\mathbf{V}$, $\odot$ denotes the Hadamard product (or called elementwise product) of matrices, and each element w_{ij} of $\mathbf{W}$, representing the weight for the jth feature of particle i, is defined as

$$w_{ij}^p = \begin{cases} 1 & |r_{ij}^p| < \lambda \\ \frac{\lambda}{|r_{ij}^p|} & \text{otherwise,} \end{cases} \tag{5}$$

where $r_{ij}^p = y_{ij} - \mathbf{u}_{i\cdot}'\mathbf{v}_{j\cdot}^p$ is the residue.

Theorem 1. *The objective function $f(\mathbf{V};\mathbf{U})$ is non-increasing under the update rule in Eqn. 4.*

Moreover, the correctness of the update rule is guaranteed by the following theorem:

Theorem 2. *The converged solution $\mathbf{V}^*$ obtained by applying the update rule in Eqn. 4 satisfies the Karush–Kuhn–Tucker (KKT) conditions.*

Due to space constraints, the proofs for Theorem 1 and Theorem 2 are provided in the supplemental material.

Connection to the trivial templates: Our approach based on the Huber loss function is actually related to the popular approach using trivial templates as in Eqn. 1. First, Eqn. 1 can be expressed equivalently in the following form:

$$\begin{aligned} \min_{\mathbf{V},\mathbf{E}} \quad & \frac{1}{2}\|\mathbf{Y}-\mathbf{U}\mathbf{V}'-\mathbf{E}\|_F^2+\lambda\|\mathbf{E}\|_1+\gamma\|\mathbf{V}\|_1 \\ s.t. \quad & \mathbf{V}\geq 0, \end{aligned} \tag{6}$$

where $\mathbf{V}'_T$ is renamed $\mathbf{E}$ with elements denoted by e_{ij}. The key to show their connections is to eliminate $\mathbf{E}$ by its optimal condition to give an equivalent optimization problem which only involves $\mathbf{V}$. As is well known from the sparse learning literature [17], the optimal $\mathbf{E}^*$ can be obtained by applying an elementwise soft-thresholding operation to the residue $\mathbf{R}=\mathbf{Y}-\mathbf{U}\mathbf{V}'$: $e^*_{ij}=\text{sgn}(r_{ij})\max(0,|r_{ij}|-\lambda)$, where $\text{sgn}(\cdot)$ is the sign operator and r_{ij} is an element of $\mathbf{R}$. Substituting the optimal $\mathbf{E}^*$ into Eqn. 6 yields Eqn. 2.

Due to space constraints, complete derivation is left to the supplemental material.

4.2. Robust Template Update

After we have processed some frames in the video sequence, it is necessary to update the object templates represented by $\mathbf{U}$ to reflect the changes in appearance or viewpoint. Suppose frame c is the current frame. We define a matrix $\mathbf{Z}=[z_{ij}]\in\mathbb{R}^{m\times c}$ in which each column represents the tracking result of one of the c frames processed so far.[1] We formulate it as a robust dictionary learning problem similar to Eqn. 2 except that $\mathbf{U}$ is now also a variable:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \psi(\mathbf{U},\mathbf{V})=\sum_i\sum_j \ell_\lambda(z_{ij}-\mathbf{u}'_{i\cdot}\mathbf{v}_{j\cdot})+\gamma\|\mathbf{V}\|_1 \\ s.t. \quad & \mathbf{U}\geq 0,\mathbf{V}\geq 0,\ \mathbf{u}'_{\cdot k}\mathbf{u}_{\cdot k}\leq 1,\forall k. \end{aligned} \tag{7}$$

Caution should be taken regarding a slight abuse of notation in exchange for simplicity. The matrix $\mathbf{V}$ here is a $c\times r$ matrix of sparse codes for c frames while that in the previous section is an $n\times r$ matrix of sparse codes for all n particles in one frame.

The rationale behind this formulation is that the appearance of the target does not change significantly and is thus similar from frame to frame. As a result, $\mathbf{Z}$ can be approximated well by a low-rank component and a sparse component whose nonzero elements correspond to outliers due to occlusion or other variations. What is more, the sparsity constraint facilitates each template to specialize in a certain aspect of the target. Unlike the incremental PCA approach in [20], a template will not be affected by the update of an irrelevant appearance and thus each template can capture better a distinctive aspect of the target. In contrast to the *ad hoc* way adopted by other sparse trackers, it can reject occlusion automatically when updating the templates. Moreover, it is advantageous to store the previous states of the target for an extended period of time and allow them to decay slowly, since it helps a lot in recovering from occlusion. As shown in Fig. 1, the learned templates summarize well different appearances of the target in the previous frames.

Optimization: To solve this optimization problem, as is often the case for similar problems, we alternate between updating $\mathbf{U}$ and $\mathbf{V}$. To optimize w.r.t. $\mathbf{V}$, the problem is the same as that in Eqn. 2 and so we use the update rule in Eqn. 4 as well. For $\mathbf{U}$, although in principle we may take the same approach by using multiplicative update, it is only limited to the batch mode. Recomputing it every time when new data arrive is very time consuming. We solve this problem by devising a projected gradient descent method, which simply refers to the gradient descent method followed by projecting the solution to the constraint set in order to satisfy the constraints. For convenience, we first present the batch algorithm and then extend it to an online version.

The projected gradient descent method updates $\mathbf{u}^p_{i\cdot}$ to $\mathbf{u}^{p+1}_{i\cdot}$ as follows:

$$\tilde{\mathbf{u}}^p_{i\cdot}=\mathbf{u}^p_{i\cdot}-\eta\,\nabla h(\mathbf{u}^p_{i\cdot}),\qquad \mathbf{u}^{p+1}_{\cdot k}=\Pi(\tilde{\mathbf{u}}^p_{\cdot k}), \tag{8}$$

where $\nabla h(\mathbf{u}^p_{i\cdot})$ denotes the gradient vector and $\eta>0$ is the step size. The gradient vector of $h(\mathbf{u}_{i\cdot};\mathbf{V})$ for each $\mathbf{u}_{i\cdot}$ is given by:

$$\nabla h(\mathbf{u}_{i\cdot})=\frac{\partial h(\mathbf{u}_{i\cdot};\mathbf{V})}{\partial \mathbf{u}_{i\cdot}}=\mathbf{V}'\mathbf{\Lambda}_i\mathbf{V}\mathbf{u}_{i\cdot}-\mathbf{V}'\mathbf{\Lambda}_i\mathbf{y}_{i\cdot}, \tag{9}$$

where $\mathbf{\Lambda}_i$ is a diagonal matrix with w^p_{ij} as its jth diagonal element and $\Pi(\mathbf{x})$ denotes a projection operation that projects each column of $\mathbf{U}$ onto the convex set $\mathcal{C}=\{\mathbf{x}:\mathbf{x}\geq 0,\mathbf{x}'\mathbf{x}\leq 1\}$. This can be done easily by first thresholding the negative elements of $\mathbf{x}$ to zero and then normalizing it in case the ℓ_2 norm of $\mathbf{x}$ is greater than one.

Inspired by recent works on online robust matrix factorization and dictionary learning [23, 16], we note that the update rule in Eqn. 8 can be expressed in terms of two matrices as sufficient statistics, which we use to devise an online algorithm. Up to frame c, the matrices are defined as:

$$\mathbf{A}^c_i=(\mathbf{V}^c)'\mathbf{\Lambda}^c_i(\mathbf{V}^c),\quad \mathbf{B}^c_i=(\mathbf{V}^c)'\mathbf{\Lambda}^c_i\mathbf{y}_{i\cdot} \tag{10}$$

After obtaining the tracking result for frame $c+1$, we update $\mathbf{A}_i$ and $\mathbf{B}_i$ as:

$$\begin{aligned} \mathbf{A}^{c+1}_i&=\rho\mathbf{A}^c_i+w_{i,c+1}\mathbf{v}_{c+1,\cdot}\mathbf{v}'_{c+1,\cdot} \\ \mathbf{B}^{c+1}_i&=\rho\mathbf{B}^c_i+w_{i,c+1}\mathbf{v}_{c+1,\cdot}\mathbf{y}_{i\cdot}, \end{aligned} \tag{11}$$

where ρ is a forgetting factor which gives exponentially less weight to past information. Then an update rule similar to the batch mode can be adopted:

$$\tilde{\mathbf{u}}^c_{i\cdot}=\mathbf{u}^c_{i\cdot}-\eta(\mathbf{A}^{c+1}_i\mathbf{u}^c_{i\cdot}-\mathbf{B}^{c+1}_i),\qquad \mathbf{u}^{c+1}_{\cdot k}=\Pi(\tilde{\mathbf{u}}^c_{\cdot k}). \tag{12}$$

In practice, we update the dictionary once every q frames. The speed of template update is affected by the parameters

[1] $\mathbf{Z}$ is similar to $\mathbf{Y}$ above except that it only has c columns.

ρ and q. Setting them small will give higher influence to more recent frames on the templates but it can lead to the potential risk of drifting. How to set the parameters will be discussed in section 6.1.

5. Implementation Details

In this section, we discuss some implementation details which further boost the performance of our tracker.

5.1. Sampling Background Templates

In the way it was presented above, our tracker is a purely generative model. However, when the object to track is heavily occluded or when it changes rapidly, the tracker may not be robust enough in avoiding the drifting problem. To further enhance model robustness, we augment the object templates with background templates randomly sampled from the previous frames. The dictionary templates $\mathbf{U}$ now consist of both the original object templates $\mathbf{U}_o$ and the background templates $\mathbf{U}_b$. Then the weights of the particles are determined by the difference in contribution of these two parts, i.e., $\exp\big(\beta(\|\mathbf{U}_o\mathbf{V}_o\|_1 - \|\mathbf{U}_b\mathbf{V}_b\|_1)\big)$ for some parameter β, which favors a result that can be represented well by the object templates but not by the background templates.

5.2. Feature Selection

There are two reasons why feature selection is needed. First, we have assumed that raw image patches in the shape of either rectangles or parallelograms are used to define features. However, when the tracked object has irregular shape, the cluttered background may be included in the image patch even when the tracking result is correct. Second, the tracked object may contain some variant parts which can incur adverse effects. Such effects should be reduced by selecting the most invariant and informative features. So we propose using ℓ_1-regularized logistic regression for feature selection:

$$\min_{\mathbf{w}} \sum_i \log\Big\{1 + \exp\big[-l_i(\mathbf{w}'\mathbf{y}_i + b)\big]\Big\} + \alpha\|\mathbf{w}\|_1, \quad (13)$$

where $\mathbf{y}_i$ is a sample from some previous frames and l_i is its corresponding label which is 1 if $\mathbf{y}_i$ is a positive sample and -1 otherwise. We solve this problem using the public-domain sparse learning package SLEP.[2] Then we project the original object templates and particles using a diagonal projection matrix $\mathbf{P}$ with each diagonal element $p_{ii} = \delta(w_i)$, where $\delta(\cdot)$ is the Dirac delta function. While this gives a more discriminative feature space for particle representation, it also reduces the computational cost. In practice, we always collect negative samples from the previous frames. As for positive samples, besides those collected, we also use the identified object from the first frame. We then run the feature selection method on these two sets separately and combine the two feature sets by finding their intersection. We start with $\alpha = 0.002$ and then decrease it by a factor of 0.75 if less than half of the pixels are selected.

[2] http://www.public.asu.edu/~jye02/Software/

(a) davidin (b) bolt

Figure 2. Comparison of two feature selection methods based on different loss functions. A white pixel indicates that the corresponding feature is selected, otherwise black. For each of the two sequences, the original image, the LASSO result and the sparse regularized logistic regression result are shown from left to right.

Although a similar feature selection step is also used in [31], it uses the Frobenius norm for the loss (as in the LASSO [22] formulation) instead of the logistic loss. We believe the logistic loss is a more suitable choice here since this is a classification problem. Fig. 2 shows the feature selection results obtained by the two methods on the first frames of the *davidin* and *bolt* sequences. The logistic loss obviously gives more discriminative features and ignores the background area. Moreover, since the sample size n is typically smaller than the data dimensionality (32×32 or 48×16 in our case), LASSO can never select more than n features due to its linearity. This restriction may lead to impaired performance.

6. Experiments

In this section, we compare the object tracking performance of the proposed tracker with several state-of-the-art trackers on some challenging video sequences. The trackers compared are MTT [30], CT [27], VTD [15], MIL [3], a latest variant of L1T [4], TLD [13], and IVT [20]. We downloaded the implementations of these methods from the websites of their authors. Except for VTD, all other methods cannot utilize the color information in the video directly. So we used the `rgb2gray` function in the MATLAB Image Processing Toolbox to convert the color video to grayscale before performing object tracking. The code implementing our method is available on the project page: http://winsty.net/onndl.html.

6.1. Parameter Setting

We set $\lambda = \gamma = 0.01$ and $\eta = 0.2$. The parameter β in Sec. 5.1 is set to 5. For template update in Sec. 4.2, ρ is set to 0.99 and q to 3 or 5 depending on whether the change in appearance of the tracked object is fast. The numbers of object templates and background templates are set to 20 and 100, respectively. We set the template size to 32×32 or 48×16 according to the shape of the object. The particle filter uses 600 particles. For the affine parameters in the particle filter, we only select the best candidate among four pos-

sible ones instead of performing an exhaustive grid search. Although performing grid search may lead to better results, we believe such tuning is not feasible in real environments. Compared to other methods, our method is relatively insensitive to the values of the affine parameters. More specifically, our method can achieve satisfactory results in 7 out of 10 video sequences when using the default parameters. The current implementation of our tracker runs at 0.7–1.5 frames per second (fps). This speed is much faster than the original L1T [18] and is comparable to the "realtime" L1T [4] if the same template size and same number of particles are used.

6.2. Quantitative Comparison

We use two common performance metrics for quantitative comparison: success rate and central-pixel error. For each frame, a tracker is said to be successful if the overlap percentage exceeds 50%. The overlap percentage is defined as $\frac{\text{area}(BB_T \cap BB_G)}{\text{area}(BB_T \cup BB_G)}$, where BB_T denotes the bounding box produced by a tracker and BB_G denotes the ground-truth bounding box. As for the central-pixel error, it is the Euclidean distance (in pixels) between the centers of BB_T and BB_G. The results are summarized in Table 1 and Table 2, respectively. For each video sequence (i.e., each row), we show the best result in red and second best in blue. We also report the central-pixel errors frame-by-frame for each video sequence in Fig. 3. Since TLD can report that the tracked object is missing in some frames, we exclude it from the central-pixel error comparison. In terms of the success rate, our method is always among the best two. With respect to the central-pixel error, our method is among the best two in 8 of the 10 sequences. For the other two sequences, the gaps are quite small. We believe they can be negligible in practical applications.

	Ours	MTT	CT	VTD	MIL	L1T	TLD	IVT
car4	100	100	24.7	35.2	24.7	30.8	0.2	100
car11	100	100	70.7	65.6	68.4	100	29.8	100
davidin	75.5	68.6	25.3	49.4	17.7	27.3	44.4	92.0
trellis	99.0	66.3	23.0	30.1	25.9	62.1	48.9	44.3
woman	91.5	19.8	16.0	17.1	12.2	21.1	5.8	21.5
bolt	74.7	19.5	46.2	28.3	72.7	7.5	6.8	85.7
shaking	98.9	12.3	92.3	99.2	26.0	0.5	15.6	1.1
skating1	92.5	9.5	3.8	93.3	6.8	5.3	47.0	6.5
football	82.9	70.7	69.6	80.4	76.2	30.7	74.9	56.3
basketball	97.2	14.8	24.6	98.6	32.3	7.4	2.3	17.1
average	91.2	48.2	39.6	59.7	36.3	35.8	29.3	52.5

Table 1. Comparison of 8 trackers on 10 video sequences in terms of success rate (in percentage).

6.3. Qualitative Comparison

Complete video sequences with the bounding boxes reported by different trackers are provided in the supplemental material. With limited space available, we try our best to give a qualitative comparison by showing in Fig. 4 some key frames of each sequence.

	Ours	MTT	CT	VTD	MIL	L1T	IVT
car4	5.3	3.4	95.4	41.5	81.8	16.8	4.2
car11	2.3	1.3	6.0	23.9	19.3	1.3	3.2
davidin	6.2	7.8	15.3	27.1	13.1	17.5	3.9
trellis	2.4	33.7	80.4	81.3	71.7	37.6	44.7
woman	7.3	257.8	109.6	133.6	123.7	138.2	111.2
bolt	7.4	35.6	12.1	22.3	9.6	237.6	7.4
shaking	6.7	28.1	10.9	5.2	28.6	90.8	138.4
skating1	7.2	184.0	98.2	7.4	104.4	140.4	146.9
football	6.5	15.9	12.0	6.5	14.7	27.6	17.7
basketball	9.9	161.7	106.3	6.8	63.2	159.2	31.6
average	6.1	72.9	54.6	35.6	53.0	87.6	50.9

Table 2. Comparison of 7 trackers on 10 video sequences in terms of central-pixel error (in pixels).

The *car4* sequence was captured on an open road. The illumination changes greatly due to the shades of trees and entrance of a tunnel. All trackers except TLD do a fine job before the car enters the tunnel at about frame 200. However, after that, only IVT, MTT and our tracker can track the car accurately. L1T can also track it but with incorrect scale. Other methods totally fail.

In the *car11* sequence, the tracked object is also a car but the road environment is very dark with background light. All methods can merely track the car in the first 200 frames. However, when the car makes a turn at about frame 260, VTD, MIL, TLD and CT drift from the car although CT can recover later to a certain degree. Other methods can track the car accurately.

The *davidin* sequence was recorded in an indoor environment. We need to track a moving face with illumination and scale changes. Most trackers drift from frame 160 due to the out-of-plane rotation. Different trackers can recover from it by various degrees.

In the *trellis* sequence, we need to track the same face as in *davidin* but in an outdoor environment with more severe illumination and pose changes. All trackers except MTT, L1T and our tracker fail at about frame 160 when the person walks out of the trellis. Furthermore, L1T and MTT lose the target at frames 310 and 350 due to the out-of-plane rotation. TLD is unstable in getting and losing the target several times. On the other hand, our tracker can track the face accurately along the whole sequence.

In the *woman* sequence, we track a walking woman in the street. The difficulty lies in that the woman is greatly occluded by the parked cars. TLD fails at frame 63 because the pose of the woman changes. All other trackers compared fail when the woman walks close to the car at about frame 130. Our tracker can follow the target accurately.

In the *bolt* sequence, the target is a running athlete with rapid changes in appearance. L1T and MTT fail at some in-

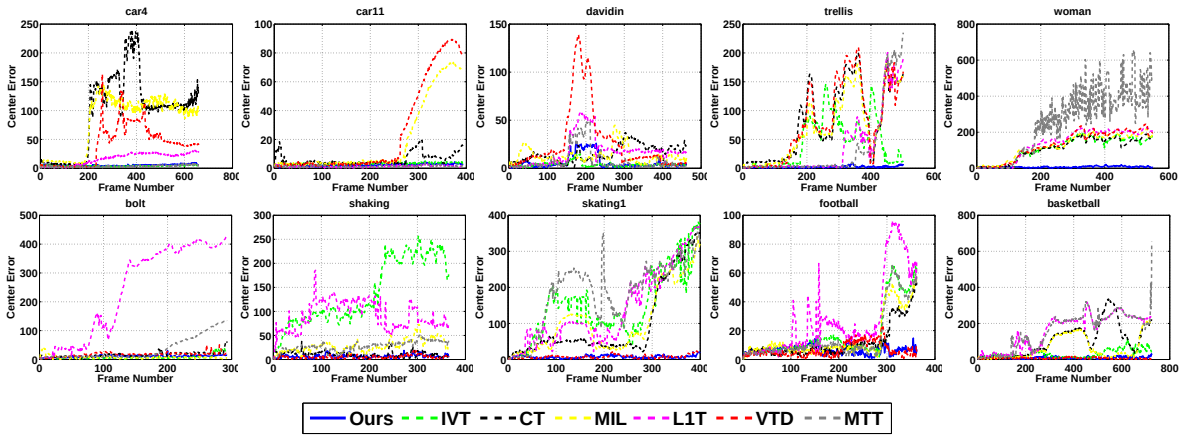

Figure 3. Frame-by-frame comparison of 7 trackers on 10 video sequences in terms of central-pixel error (in pixels).

termediate frames, but all other trackers can track the target successfully till the end. Among all methods, ours and IVT get the most accurate result.

In the *shaking* sequence, the tracked head is subject to drastic pose and illumination changes. L1T, IVT and TLD totally fail before frame 10, while MTT and MIL show some drifting effects then. VTD gives the best result, which is followed by our tracker and CT.

The *skating1* sequence contains abrupt motion, occlusion and significant illumination changes, which make most trackers fail. Our tracker gives comparable result with VTD.

The *football* sequence aims to track a helmet in a football match. The difficulty lies in that many helmets are similar in background. Moreover, collision with another player often confuses most trackers. Only our tracker and VTD can successfully locate the correct object.

In the *basketball* sequence, the appearance of the tracked player changes rapidly in the intense match. Moreover, the players in the same team have similar appearance. Only VTD and our tracker can survive to the end.

We note that the *football* and *basketball* sequences demonstrate the effectiveness of sampling background templates as described in Sec. 5.1. It helps a lot in distinguishing the true target from distractors in the background.

7. Conclusion and Future Work

In this paper, we have proposed a novel visual tracking method based on robust non-negative dictionary learning. Instead of taking an *ad hoc* approach to update object templates, we formulate this procedure as a robust non-negative dictionary learning problem and propose a novel online projected gradient descent method to solve it. The most appealing advantage is that it can detect and reject the occlusion and cluttered background automatically. Besides, we have proposed to get rid of the trivial templates by using the Huber loss function in particle representation. To solve the resulted optimization problem, we devise a simple and efficient multiplicative update rule. It can be further shown that our new formulation is equivalent to the approach of using trivial templates by other sparse trackers. We have conducted extensive comparative studies on some challenging benchmark video sequences and shown that our proposed tracker generally outperforms existing methods.

One possible extension of our current method is to exploit the underlying structures and relationships between particles, as in MTT [30]. We expect it to lead to further performance gain.

Acknowledgment

This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311– 4322, 2006. 2

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002. 3

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 2, 5

[4] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust L1 tracker using accelerated proximal gradient approach. In *CVPR*, pages 1830–1837, 2012. 2, 5, 6

[5] A. Doucet, D. N. Freitas, and N. Gordon. *Sequential Monte Carlo Methods In Practice*. Springer, New York, 2001. 1, 2

[6] L. Du, X. Li, and Y.-D. Shen. Robust nonnegative matrix factorization via half-quadratic minimization. In *ICDM*, pages 201–210, 2012. 2

[7] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, pages 47–56, 2006. 2

[8] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011. 2

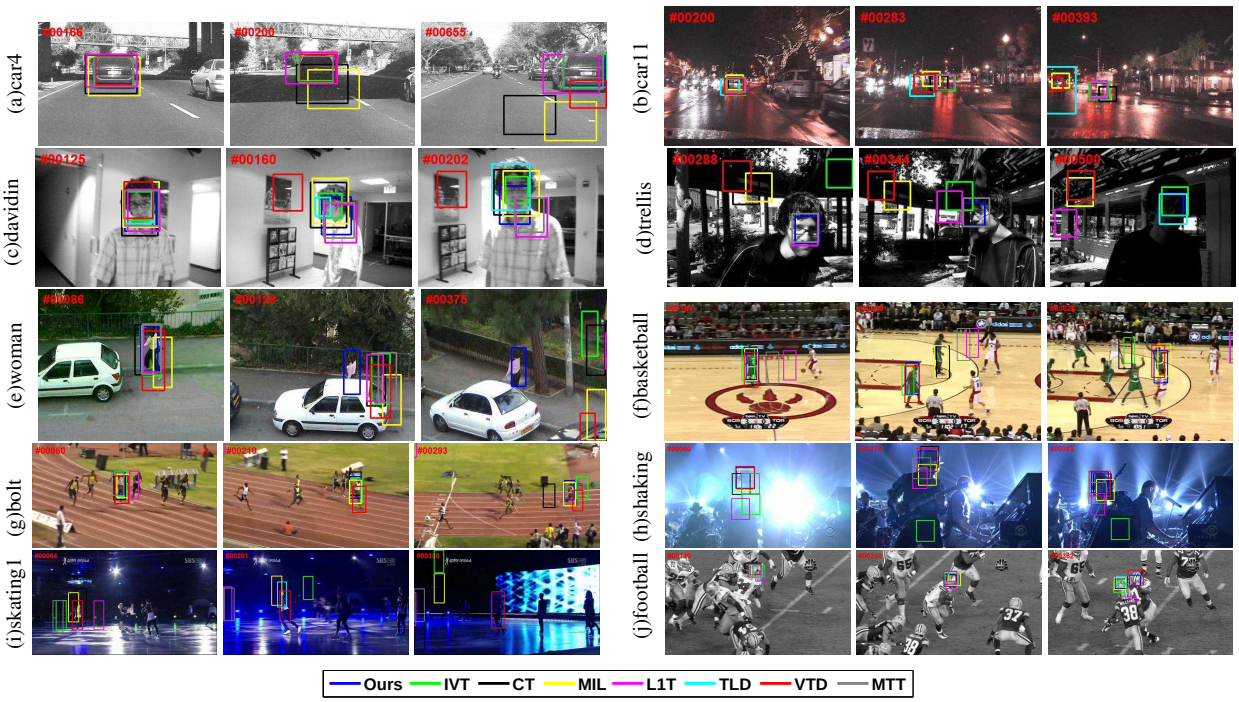

Figure 4. Comparison of 8 trackers on 10 video sequences in terms of bounding box reported.

[9] P. Hoyer. Non-negative sparse coding. In *Proceedings of the Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002. 2

[10] P. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. 1, 3

[11] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829, 2012. 2

[12] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, 2010. 2

[13] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. 2, 5

[14] D. Kong, C. Ding, and H. Huang. Robust nonnegative matrix factorization using L21-norm. In *CIKM*, pages 673–682, 2011. 2

[15] J. Kwon and K. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010. 2, 5

[16] C. Lu, J. Shi, and J. Jia. Online robust dictionary learning. In *CVPR*, 2013. 2, 4

[17] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010. 2, 4

[18] X. Mei and H. Ling. Robust visual tracking using l_1 minimization. In *ICCV*, pages 1436–1443, 2009. 1, 2, 6

[19] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient l_1 tracker with occlusion detection. In *CVPR*, pages 1257–1264, 2011. 2

[20] D. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008. 2, 4, 5

[21] D. Seung and L. Lee. Algorithms for non-negative matrix factorization. *NIPS*, pages 556–562, 2001. 2

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. 1, 5

[23] N. Wang, T. Yao, J. Wang, and D.-Y. Yeung. A probabilistic approach to robust matrix factorization. In *ECCV*, pages 126–139, 2012. 2, 4

[24] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 2

[25] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009. 2

[26] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006. 2

[27] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, pages 864–877, 2012. 2, 5

[28] S. Zhang, H. Yao, X. Sun, and X. Lu. Sparse coding based visual tracking: review and experimental comparison. *Pattern Recognition*, 46(7):1772–1788, 2012. 2

[29] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. *ECCV*, pages 470–484, 2012. 1, 2

[30] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, pages 2042–2049, 2012. 1, 2, 5, 7

[31] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845, 2012. 2, 5