Object Tracking by Occlusion Detection via Structured Sparse Learning

Tianzhu Zhang[1], Bernard Ghanem[1,2], Changsheng Xu[3], Narendra Ahuja[1,4]
[1] Advanced Digital Sciences Center of Illinois, Singapore
[2] King Abdullah University of Science and Technology, Saudi Arabia
[3] Institute of Automation, Chinese Academy of Sciences, P. R. China, CSIDM
[4] University of Illinois at Urbana-Champaign, Urbana, IL USA
tz.zhang@adsc.com.sg, bernard.ghanem@kaust.edu.sa, csxu@nlpr.ia.ac.cn, n-ahuja@illinois.edu

Abstract

Sparse representation based methods have recently drawn much attention in visual tracking due to good performance against illumination variation and occlusion. They assume the errors caused by image variations can be modeled as pixel-wise sparse. However, in many practical scenarios these errors are not truly pixel-wise sparse but rather sparsely distributed in a structured way. In fact, pixels in error constitute contiguous regions within the object's track. This is the case when significant occlusion occurs. To accommodate for non-sparse occlusion in a given frame, we assume that occlusion detected in previous frames can be propagated to the current one. This propagated information determines which pixels will contribute to the sparse representation of the current track. In other words, pixels that were detected as part of an occlusion in the previous frame will be removed from the target representation process. As such, this paper proposes a novel tracking algorithm that models and detects occlusion through structured sparse learning. We test our tracker on challenging benchmark sequences, such as sports videos, which involve heavy occlusion, drastic illumination changes, and large pose variations. Experimental results show that our tracker consistently outperforms the state-of-the-art.

1. Introduction

Visual tracking is a classical problem in computer vision; it is a core task for many applications e.g. automatic surveillance, robotics, human computer interaction, etc. It is also very challenging due to appearance variations such as occlusion, illumination change, significant motion, background clutter, etc. Over the years, a significant amount of effort has been made to overcome these challenges. To survey many of these algorithms, we refer the reader to [21].

A truly robust tracking method must be able to handle occlusion. However, modeling occlusion is not straightforward and non-trivial by far. There exists a significant amount of work that addresses this issue through statistical analysis [8, 18], robust statistics [1, 3], patch matching [19], the use of multiple cameras [5, 15], context information [20], model analysis [6], and learning occlusion with Likelihoods [10]. Recently, sparse representation has been successfully applied to visual tracking [14, 24, 25, 23] under the particle filter framework as an attempt to alleviate the occlusion problem in tracking. In these methods, particles are randomly sampled around the current state of the tracked object according to a zero-mean Gaussian distribution. At time t, n particles are sampled. The observation (pixel color values) of each particle in the frame is denoted as: $\vec{\mathbf{x}} \in \mathbb{R}^d$. In the noiseless case, each particle $\vec{\mathbf{x}}$ is represented as a linear combination $\vec{\mathbf{z}}$ of templates that form a dictionary $\mathbf{D} = \left[\vec{\mathbf{d}}_1, \vec{\mathbf{d}}_2, \cdots, \vec{\mathbf{d}}_m\right]$, such that $\vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}}$. $\mathbf{D}$ can be constructed from an overcomplete sampling of the

Figure 1. (a) Frames from six different video sequences, portraying significant occlusion. The ground truth track of each object is designated in *green*. Clearly, occlusion renders the tracking problem very difficult. However, certain assumptions about the structuredness of occlusion (e.g. spatial contiguity) can be exploited to alleviate its affect on tracking performance.

978-0-7695-4990-3/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPRW.2013.150

target object, based on an initial bounding box at the start of tracking, and dynamically updated to maintain an up-to-date target appearance model.

In many visual tracking scenarios, targets are often partially occluded or corrupted by noise. Occlusion is unpredictable as it may affect any part, or occlude any amount, of the target. The occluded object can be either a connected region or a number of randomly scattered pixels, though the former is more likely in natural images. In addition, only a sparse number of these templates is required to reliably represent each particle, which encourages $\vec{\mathbf{z}}$ to be sparse. To incorporate these two pieces of information, each particle $\vec{\mathbf{x}}$ should be represented as a sparse linear combination, while allowing for sparse error $\vec{\mathbf{e}}$ to encode occlusion: $\vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}} + \vec{\mathbf{e}}$. The sparse coefficients $\vec{\mathbf{z}}$ and sparse error $\vec{\mathbf{e}}$ are recovered by solving the following ℓ_1 minimization problem. The current tracking result is usually chosen to the be the particle $\vec{\mathbf{x}}$ with minimum reconstruction error w.r.t. dictionary $\mathbf{D}$.

$$\min \|\vec{\mathbf{z}}\|_1 + \|\vec{\mathbf{e}}\|_1 \qquad s.t. \quad \vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}} + \vec{\mathbf{e}} \tag{1}$$

This approach has demonstrated to be robust against partial occlusions, which improves tracking performance. However, it suffers from the following drawbacks.

(1) The error (due to occlusion) is not sparse for many tracking scenarios, as exemplified in Figure 1. Because a portion of the target is significantly occluded, we need to discard that portion for the sparsity assumption to still hold.

(2) This kind of algorithm does not exploit any prior information about the occlusion, especially the important property that occlusion is spatially contiguous. By modeling error pixels as structured and sparse, the representation is made more faithful and better defined.

Motivated by the above drawbacks and inspired by previous work, we propose a new particle filter tracker that involves tracking by occlusion detection, thus, appropriately named the TOD tracker. In each frame, particles are represented in a structured sparse learning framework, which exploits prior information about the location of occlusion and its spatial contiguity. This prior is propagated from previous frames in the form of an occlusion mask. The main goal of this paper is to show how this prior information can be effectively incorporated into the sparse representation framework, thus, improving its robustness against more types of realistic occlusions.

Contributions: Compared with existing methods, the contributions of this work are two-fold. **(1)** We propose a structured sparse learning method for occlusion detection in object tracking. It exploits structure information to make occlusion both sparse and spatially continuous for more robust performance. To the best of our knowledge, this is the first work to use occlusion prior information through structured sparsity in object tracking. **(2)** Compared to the popular L_1 tracker [14] that does not model occlusion explicitly, our method is generic. In fact, it yields the L_1 tracker as a special case.

The paper is organized as follows. The proposed tracking approach and optimization methodology are presented in Sections 2 and 3 respectively. In Section 4, we report and analyze extensive experimental results.

2. Tracking by Occlusion Detection (TOD)

In this section, we give a detailed description of our particle filter based tracking method, which makes use of occlusion prior information in a structured sparse learning framework to represent particle samples.

2.1. Occlusion Detection via Structured Sparsity

In this section, we discuss how we incorporate a sparsity-inducing norm that also encodes prior structural information (spatial contiguity) regarding the support of the error incurred when sparse linear representation is used to describe particles. We expect that such structural information renders a more faithful and robust representation model that can handle occlusions in object tracking.

In our particle filter based tracking method, particles are randomly sampled around the current state of the tracked object according to a zero-mean Gaussian distribution. Similar to [14], we assume an affine motion model between consecutive frames. Therefore, the state of a particle $\mathbf{s}_t$ consists of the six affine transformation parameters (2D linear transformation and translation). By applying an affine transformation based on $\mathbf{s}_t$, we crop the region of interest $\mathbf{y}_t^*$ from the image and normalize it to the same size as the target templates in our dictionary. The state transition distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is modeled to be a zero-mean Gaussian, with the dimensions of $\mathbf{s}_t$ independent. The observation model $p(\mathbf{y}_t^*|\mathbf{s}_t)$ reflects the similarity between a particle and target templates in the dictionary. In this paper, $p(\mathbf{y}_t^*|\mathbf{s}_t)$ is inversely proportional to the reconstruction error obtained by linearly representing $\mathbf{y}_t^*$ using the template dictionary.

In the t^{th} frame, we sample n particles, where the observation (pixel color values) of the i^{th} particle is denoted in vector form as: $\vec{\mathbf{x}} \in \mathbb{R}^d$ (for simplicity, we ignore the subscript i). The observation $\vec{\mathbf{x}}$ of a particle is represented as a sparse linear combination $\vec{\mathbf{z}}$ of m dictionary templates $\mathbf{D} \in \mathbb{R}^{d \times m}$, as shown in Eq (1). $\mathbf{D}$ is updated dynamically to handle frame-to-frame changes in target appearance. The dictionary update issue is addressed later.

The L1 tracking work [14], which represents each particle by solving an ℓ_1 LASSO problem, can be generalized as shown in Eq (1).

$$\min_{\vec{\mathbf{z}},\vec{\mathbf{e}}} \|\vec{\mathbf{z}}\|_1 + \varphi(\vec{\mathbf{e}}) \quad s.t. \quad \vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}} + \vec{\mathbf{e}}, \tag{2}$$

In the L1 tracker, the regularizer $\varphi(\bullet)$ on $\vec{\mathbf{e}}$ is chosen to be $\|\vec{\mathbf{e}}\|_1$. This regularization scheme encourages the error (e.g. occlusion) to be pixel-wise sparse. This assumption fails in many tracking scnearios as exemplified in Fig. 1. It also does not incorporate the structural information inherent to occlusion, namely spatial contiguity. Basically, the ℓ_1-norm regularization treats each entry (pixel) in $\vec{\mathbf{e}}$ independently. It does not take into account any specific structures or possible relations among subsets of the entries.

To encode this structured prior information, we assume that the spatial support of the error is contiguous. This can be enforced by modeing the error as spatially smooth. Also, this error can be assumed to be sparse, if any significant occlusion is detected and removed beforehand. Note that we assume that some pixels in a particle are occluded and those are determined by an occlusion mask that is propagated from frame-to-frame. At every frame, this mask is used to determine the pixels, from which the particle representation $\vec{\mathbf{z}}$ is computed. This representation is used to estimate the error at *each* pixel in the particle. By thresholding this error with a predefined threshold, the occlusion mask is updated and propagated to the next frame.

To incorporate pairwise relationships between pixels in the particle, we adopt a graph-guided fused LASSO framework that explicitly takes into account the complex dependency structure represented as a graph, whose nodes are pixels in the particle. We assume that the d pixels in each particle are organized in a graph G with a set of nodes V and edges E. In this paper, we adopt a simple strategy for constructing such a graph, whereby an edge exists between any pair of neighboring pixels and its weight is proportional to the correlation of their intensity values and inversely proportional to the Euclidean distance between them. More sophisticated methods can be employed, but they are not the focus of this paper. Let $\mathbf{w}_{ml}$ denote the weight of an edge $(m,l) \in E$ that represents the strength of correlation between pixels m and l. Therefore, to encourage spatial contibuity between particle pixels, we emply a graph-guided fusion penalty, which extends the standard LASSO by fusing the $\mathbf{e}_m$ and $\mathbf{e}_l$ if $(m,l) \in E$. With the above notation, we formulate the representation problem as a structured sparse ℓ_1 problem as follows. Details of solving this problem are provided in Section 3.

$$\begin{aligned} &\min_{\vec{\mathbf{z}},\vec{\mathbf{e}}} \|\vec{\mathbf{z}}\|_1 + \lambda\|\vec{\mathbf{e}}\|_1 + \gamma \sum_{(m,l)\in E} \mathbf{w}_{ml}\|\mathbf{e}_m - \mathbf{e}_l\|_1 \\ &s.t. \quad \vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}} + \vec{\mathbf{e}}, \end{aligned} \tag{3}$$

where λ and γ are tradeoff parameters that control the complexity of the model. A larger value for γ leads to a greater fusion effect. The $\mathbf{w}_{ml}$ weighs the fusion penalty for each edge such that $\mathbf{e}_m$ and $\mathbf{e}_l$ for highly correlated pixels have a large $\mathbf{w}_{ml}$.

Discussion: As shown in Eq (3), we propose a generic formulation for robust object tracking using structured sparse learning. By defining γ differently, different object trackers are obtained. When $\gamma = 0$, TOD becomes the popular L_1 tracker [14]. In this way, the popular L_1 tracker [14] is a special case of our formulation. To the best of our knowledge, introducing the structured information in occlusion detection for tracking has not been proposed in any of the previous works. In Fig. 2, we present an example of how our TOD tracker works as compared to the L1 tracker. In the top row, we show a result of representing particle $\vec{\mathbf{x}}$ using structured sparse learning instead of traditional sparse learning (used in L1 tracking) , whose result is shown in the bottom row. Clearly, the error generated by TOD leads to a high response at the actual location of the occlusion, while it is missed by traditional sparse learning. It is evident that by enforcing spatial contiguity on the error values, the occlusion can be better localized. This error is thresholded to produce an occlusion mask that is propagated to the next frame.

2.2. Dictionary Template Update

A large body of work in the literature has proposed the use of object templates for visual tracking [12]. Target appearance remains the same only for a certain period of time, but eventually the object templates are no longer an accurate representation of its appearance. A fixed appearance template is not sufficient to handle changes in appearance due to occlusion or changes in illumination and pose. Also, if the templates are updated too often, small errors are introduced each time a template is updated, errors accumulate, and the tracker may drift from the target. Many approaches have been proposed over the years to address the drift problem [13, 9]. In this paper, we do so by dynamically updating templates in $\mathbf{D}$.

To initialize the object and background dictionaries, we sample equal-sized patches at and around the initial position of the object. In our experiments, we shift the initial bounding box by 1-3 pixels in each direction, thus, resulting in $m = 20$ object templates as in [14]. Note that m is a user-defined parameter. All templates are normalized. To each object template, we allocate a weight ω_i that is indicative of how representative the template is. In fact, the more a template is used to represent tracking results, the higher is its weight. Next, we describe how we use these weights to update $\mathbf{D}$.

As mentioned earlier, the tracking result at instance t is the particle $\vec{\mathbf{z}}_i$ that is best represented by $\mathbf{D}$ such that $i = \arg\min_{k=1,\ldots,n} (\|\vec{\mathbf{x}}_k - \mathbf{D}\vec{\mathbf{z}}_k\|_2)$. The weight of an object template in $\mathbf{D}$ is updated depending on how much that

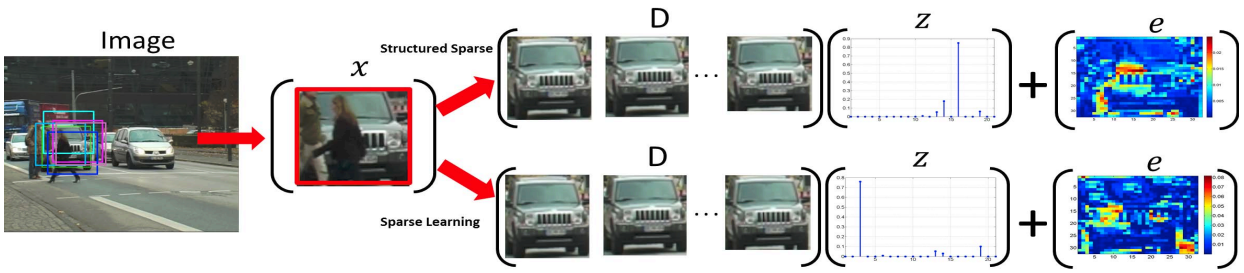

Figure 2. Schematic example of TOD. The representation $\mathbf{z}$ of particle $\mathbf{x}$ w.r.t. dictionary $\mathbf{D}$ is learned by solving Eq (3). Notice that $\mathbf{z}$ is sparse in general, i.e. a few dictionary templates are used to represent $\mathbf{x}$. The first row is our TOD, and the second row is the popular L_1 tracker. Compared with the L_1 tracker, our methods can obtain much more continuous occlusion detection result.

Algorithm 1: *Dictionary Template Update*

Predefined threshold ϵ_1 and ϵ_2
$\vec{\mathbf{y}}^*$ is the newly chosen tracking target and $\vec{\mathbf{z}}_i$ its representation. Set $\Delta d_i = \|\vec{\mathbf{x}}_i - \mathbf{D}\vec{\mathbf{z}}_i\|_2$ and $sim_i = sim(\mathbf{D}, \vec{\mathbf{y}}^*)$, where sim is the maximum similarity between $\vec{\mathbf{y}}$ and all elements in $\mathbf{D}$.
$\vec{\omega}$ is the current weight vector of templates in $\mathbf{D}$
Update weights according to the coefficients of the target templates: $\omega_k \leftarrow \omega_k \exp(\vec{\mathbf{z}}_i(k))$
if $(sim_i < \epsilon_1$ & $\Delta d_i > \epsilon_2)$ **then**
  $r \leftarrow \arg\min_{k=1,\ldots,m_O} \omega_k$
  $\mathbf{D}(:, r) \leftarrow \vec{\mathbf{y}}^*$, /*replace template with $\vec{\mathbf{y}}^*$ */
  $\omega_r \leftarrow$ median$(\vec{\omega})$, /*replace weight*/
end if
Normalize $\vec{\omega}$ such that $\|\vec{\omega}\|_1 = 1$

template is used in representing $\vec{\mathbf{z}}_i$. If $\vec{\mathbf{z}}_i$ is sufficiently represented (up to a predefined threshold) by the dictionary, then there is no need to update it. Otherwise, the current tracking result replaces the object template that has the smallest weight. The weight of this new template is set to the median of the current normalized weight vector $\vec{\omega}$. This template update scheme is summarized in Algorithm 1. We have two criteria: (1) The similarity sim_i between the current tracking result and template should be smaller than ϵ_1, which avoids updating templates frequently and thus avoids tracker drift; Once the current tracking result leads to a big variance, we add it to the dictionary by replacing it with the 'least' used dictionary template; (2) The error Δd_i should be smaller than ϵ_2, which means we update the dictionary template if only if there is no occlusion; In our experiments, ϵ_1 and ϵ_2 are set to be 0.6, and 0.7, respectively.

3. Optimization

In this section, we provide a detailed description of how Eq (3) is solved efficiently. First, we rewrite the graph-guided fusion LASSO problem in Eq (3), using a vertex-edge incident matrix $\mathbf{W} \in \mathbb{R}^{|E| \times d}$, as follows:

$$\sum_{(m,l)\in E} \mathbf{w}_{ml} \|\mathbf{e_m} - \mathbf{e_l}\|_1 = \|\mathbf{W}\vec{\mathbf{e}}\|_1$$

where each row in $\mathbf{W}$ corresponds to an edge in the graph. If we label each edge with a linear index, we can define $\mathbf{W}$ formally as below:

$$\mathbf{W}_{j,k} = \begin{cases} \mathbf{w}_{ml} & if \quad j = (m,l) \quad and \quad k = m \\ -\mathbf{w}_{ml} & if \quad j = (m,l) \quad and \quad k = l \\ 0 & otherwise \end{cases}$$

Therefore, the overall penalty in Eq (3) including both LASSO and graph-guided fusion penalty functions can be written as $\|\mathbf{B}\vec{\mathbf{e}}\|_1$, where $\mathbf{B} = [\lambda\mathbf{W}; \gamma\mathbf{I}]$ and $\mathbf{I} \in \mathbb{R}^{d\times d}$ denotes an identity matrix. Then, the structured sparsity problem in Eq (3) is converted into the following problem:

$$\min_{\vec{\mathbf{z}},\vec{\mathbf{e}}} \|\vec{\mathbf{z}}\|_1 + \|\mathbf{B}\vec{\mathbf{e}}\|_1 \quad s.t. \quad \vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}} + \vec{\mathbf{e}} \tag{4}$$

To solve Eq 4, we introduce two slack variables and add two equality constraints, thus, converting it into Eq (5).

$$\min_{z,e} \ \|\vec{\mathbf{z}}_1\|_1 + \left\|\vec{\mathbf{f}}\right\|_1 \tag{5}$$
$$\text{such that: } \vec{\mathbf{x}} = \mathbf{D}\vec{\mathbf{z}}_2 + \vec{\mathbf{e}};\ \vec{\mathbf{z}}_2 = \vec{\mathbf{z}}_1;\ \vec{\mathbf{f}} = \mathbf{B}\vec{\mathbf{e}}$$

This transformed problem can be minimized using the conventional Inexact Augmented Lagrange Multiplier (IALM) method that has attractive quadratic convergence properties and is extensively used in matrix rank minimization problems [16]. IALM is an iterative method that augments the traditional Lagrangian function with quadratic penalty terms. This allows closed form updates for each of the unknown variables. By introducing augmented lagrange multipliers (ALM) to incorporate the equality constraints into the cost function, we obtain the Lagrangian function in Eq (6) that we show, in what follows, can be optimized

through a sequence of simple closed form update operations (refer to Eq (7)).

$$
\begin{aligned}
&L(\vec{\mathbf{z}}_{1-2}, \vec{\mathbf{y}}_{1-3}, u_{1-3}) \\
&= \|\vec{\mathbf{z}}_1\|_* + \left\|\vec{\mathbf{f}}\right\|_1 \\
&\quad + tr\left[\vec{\mathbf{y}}_1^T\left(\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}}\right)\right] + \frac{u_1}{2}\left\|\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}}\right\|_F^2 \\
&\quad + tr\left[\vec{\mathbf{y}}_2^T\left(\vec{\mathbf{z}}_2 - \vec{\mathbf{z}}_1\right)\right] + \frac{u_2}{2}\left\|\vec{\mathbf{z}}_2 - \vec{\mathbf{z}}_1\right\|_F^2 \\
&\quad + tr\left[\vec{\mathbf{y}}_3^T\left(\vec{\mathbf{f}} - \mathbf{B}\vec{\mathbf{e}}\right)\right] + \frac{u_3}{2}\left\|\vec{\mathbf{f}} - \mathbf{B}\vec{\mathbf{e}}\right\|_F^2 \qquad (6) \\
&\Rightarrow \min_{\vec{\mathbf{z}}_{1-2}, \vec{\mathbf{y}}_{1-3}, u_{1-3}} L(\vec{\mathbf{z}}_{1-2}, \vec{\mathbf{y}}_{1-3}, u_{1-3}) \qquad (7)
\end{aligned}
$$

$\vec{\mathbf{y}}_1$, $\vec{\mathbf{y}}_2$, and $\vec{\mathbf{y}}_3$ are lagrange multipliers, and $u_1 > 0$, $u_2 > 0$, and $u_3 > 0$ are three penalty parameters. The above problem can by solved by either exact or inexact ALM algorithms [11]. For efficiency, we choose the inexact ALM, whose details we outline in Algorithm (2). Its convergence properties can be proven similar to those in [11]. In fact, both IALM is an iterative algorithm that solves for each variable in a coordinate descent manner. In other words, each iteration of IALM involves the updating of each variable one-at-a-time, with the other variables fixed to their most recent values. Consequently, we obtain five update steps corresponding to the five sets of variables we need to optimize for. Note that Steps 1-5 all have closed form solutions.

Step 1: [Update $\vec{\mathbf{z}}_1$] Updating $\vec{\mathbf{z}}_1$ requires the solution to the optimization problem in Eq (8). This solution can be computed in closed form in Eq (9), where $\mathcal{S}_\lambda(\mathbf{z}_{ij}) = sign(\mathbf{z}_{ij})\max(0, |\mathbf{z}_{ij}| - \lambda)$ is the soft-thresholding operator, and $\mathbf{z}_{ij}$ is the jth element of vector $\vec{\mathbf{z}}$.

$$
\vec{\mathbf{z}}_1^* = \arg\min_{\vec{\mathbf{z}}_1} \frac{1}{u_1}\|\vec{\mathbf{z}}_1\|_* + \frac{1}{2}\left\|\vec{\mathbf{z}}_1 - \left(\vec{\mathbf{z}}_2 + \frac{1}{u_2}\vec{\mathbf{y}}_2\right)\right\|_F^2 \qquad (8)
$$

$$
\Rightarrow \boxed{\vec{\mathbf{z}}_1^* = \mathcal{S}_{\frac{1}{u_1}}\left(\vec{\mathbf{z}}_2 + \tfrac{1}{u_2}\vec{\mathbf{y}}_2\right)} \qquad (9)
$$

Step 2: [Update $\vec{\mathbf{f}}$] $\vec{\mathbf{f}}$ is updated by solving the optimization problem in Eq (10) with the closed form solution shown in Eq (11).

$$
\vec{\mathbf{f}}^* = \arg\min_{\vec{\mathbf{f}}} \frac{1}{u_3}\left\|\vec{\mathbf{f}}\right\|_1 + \frac{1}{2}\left\|\vec{\mathbf{f}} - \left(\mathbf{B}\vec{\mathbf{e}} + \frac{1}{u_3}\vec{\mathbf{y}}_3\right)\right\|_F^2 \qquad (10)
$$

$$
\Rightarrow \boxed{\vec{\mathbf{f}}^* = \mathcal{S}_{\frac{1}{u_3}}\left(\mathbf{B}\vec{\mathbf{e}} + \tfrac{1}{u_3}\vec{\mathbf{y}}_3\right)} \qquad (11)
$$

Step 3: [Update $\vec{\mathbf{e}}$] $\vec{\mathbf{e}}$ is updated by solving the optimization problem in Eq (12) with the closed form solution shown in Eq (13).

$$
\begin{aligned}
\vec{\mathbf{e}}^* &= \arg\min_{\vec{\mathbf{e}}} \; tr[\vec{\mathbf{y}}_1^t(\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}})] + \frac{u_1}{2}\|\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}}\|_F^2 \\
&\quad + tr[\vec{\mathbf{y}}_3^t(\mathbf{B}\vec{\mathbf{e}} - \vec{\mathbf{f}})] + \frac{u_3}{2}\left\|\mathbf{B}\vec{\mathbf{e}} - \vec{\mathbf{f}}\right\|_F^2 \qquad (12) \\
\Rightarrow &\boxed{\vec{\mathbf{e}}^* = \left(\mathbf{B}^T\mathbf{B} + \mathbf{I}\right)^{-1}\mathbf{G}}, \qquad (13)
\end{aligned}
$$

where $\mathbf{G} = \vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 + \frac{1}{u_1}\vec{\mathbf{y}}_1 - \mathbf{B}^T\left(\frac{1}{u_3}\vec{\mathbf{y}}_3 - \vec{\mathbf{f}}\right)$.

Step 4: [Update $\vec{\mathbf{z}}_2$] $\vec{\mathbf{z}}_2$ is updated by solving the optimization problem in Eq (14) with the closed form solution shown in Eq (15).

$$
\begin{aligned}
\vec{\mathbf{z}}_2^* &= \arg\min_{\vec{\mathbf{z}}_2} \; tr[\vec{\mathbf{y}}_1^t(\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}})] + \frac{u_1}{2}\|\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}}\|_F^2 \\
&\quad + tr[\vec{\mathbf{y}}_2^t(\vec{\mathbf{z}}_2 - \vec{\mathbf{z}}_1)] + \frac{u_2}{2}\|\vec{\mathbf{z}}_2 - \vec{\mathbf{z}}_1\|_F^2 \qquad (14) \\
\Rightarrow &\boxed{\vec{\mathbf{z}}_2^* = \left(\mathbf{D}^T\mathbf{D} + \mathbf{I}\right)^{-1}\mathbf{G}}, \qquad (15)
\end{aligned}
$$

where $\mathbf{G} = \mathbf{D}^T(\vec{\mathbf{x}} - \vec{\mathbf{e}} + \frac{1}{u_1}\vec{\mathbf{y}}_1) + \vec{\mathbf{z}}_1 - \frac{1}{u_2}\vec{\mathbf{y}}_2$.

Step 5: Update Multipliers $\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2$: We update the Lagrange multipliers in Eq (16), where $\rho > 1$.

$$
\begin{cases}
\vec{\mathbf{y}}_1 = \vec{\mathbf{y}}_1 + u_1(\vec{\mathbf{x}} - \mathbf{D}\vec{\mathbf{z}}_2 - \vec{\mathbf{e}}) \\
\vec{\mathbf{y}}_2 = \vec{\mathbf{y}}_2 + u_2(\vec{\mathbf{z}}_2 - \vec{\mathbf{z}}_1) \\
\vec{\mathbf{y}}_3 = \vec{\mathbf{y}}_3 + u_3(\vec{\mathbf{f}} - \mathbf{B}\vec{\mathbf{e}}) \\
u_1 = \rho u_1;\; u_2 = \rho u_2;\; u_3 = \rho u_3
\end{cases} \qquad (16)
$$

The IALM algorithm that solves Eq (5) is shown in Algorithm (2), where convergence is reached when the change in objective function or solution $\vec{\mathbf{z}}$ is below a user-defined threshold $\epsilon = 10^{-3}$. Empirically, we find that our IALM algorithm is insensitive to a large range of ϵ values. In our implementation, $u_1 = u_2 = u_3$.

Computational Complexity

For the proposed TOD, it just uses the soft-thresholding operator, and is also very fast. This complexity is on par with that of other fast particle-based tracking algorithms. In comparison, the computational complexity of the L_1 tracker [14], which uses a sparse linear representation similar to our proposed tracker, is at least $\mathcal{O}\left(nd^2\right)$, since the number of dictionary templates (object and trivial) is $(m + 2d)$ and n Lasso problems are solved independently. Clearly, our method is more computationally attractive than L_1 tracker. When $m = 21$, $n = 400$, and $d = 32 \times 32$, the average per-frame run-time for TOD and L_1 trackers are about 5 seconds and 6 minutes, respectively.

4. Experimental Results

In this section, we do experimental results that validate the effectiveness and efficiency of our TOD method. We also make a thorough comparison between TOD and state-of-the-art tracking methods where applicable.

Algorithm 2: *Structured sparse learning for occlusion detection (Solving Eq (5))*

Input : data $\vec{\mathbf{x}}$, parameters λ, γ, and ρ
Output: $\vec{\mathbf{z}}$

Initialize $\vec{\mathbf{z}}_2 = \mathbf{0}$, $\vec{\mathbf{y}}_1 = 0$, $\vec{\mathbf{y}}_2 = 0$, $\vec{\mathbf{y}}_3 = 0$
while *not converged* **do**
  fix other variables and update $\vec{\mathbf{z}}_1$ [Eq (9)]
  fix other variables and update $\vec{\mathbf{f}}$ [Eq (11)]
  fix other variables and update $\vec{\mathbf{e}}$ [Eq (13)]
  fix other variables and update $\vec{\mathbf{z}}_2$ [Eq (15)]
  update multipliers and parameters [Eq (16)]
  Update final solution $\vec{\mathbf{z}} \leftarrow \vec{\mathbf{z}}_2$
end

Datasets and Baseline Trackers: We compile a set of 10 challenging tracking sequences to evaluate TOD. The sequences are sports videos and general videos include challenging appearance variations due to changes in pose, illumination, scale, and occlusion. Most of them involve various types of partial occlusions or multiple occlusions. We compare our TOD method to 6 recent and state-of-the-art trackers denoted as: L_1 [14], RCT [22], MIL [2], IVT [17], Frag [1], and OAB [7]. We implemented them using publicly available source codes or binaries provided by the authors. They were initialized using their default parameters.

Implementation Details: The initial position of the target is selected manually, and we shift the initial bounding box by 1-3 pixels in each dimension, thus, resulting in $m = 21$ target templates $\mathbf{D}$ (similar to L_1 tracker [14]). All our experiments are done using MATLAB on a 2.66GHZ Intel Core2 Duo PC with 18GB RAM. For all experiments, we model $p(\vec{\mathbf{s}}_t|\vec{\mathbf{s}}_{t-1}) \sim \mathcal{N}(\vec{\mathbf{0}}, \text{diag}(\vec{\sigma}))$, where $\vec{\sigma} = [0.005, 0.0005, 0.0005, 0.005, 3, 3]^T$. We set the number of particles $n = 400$. In Algorithm 2, we set $\lambda = 1$ and $\gamma = 5$. Next, we give a qualitative and quantitative analysis of TOD, and compare it against the 6 baseline methods. Our experiments show that TOD produces more robust and accurate tracks.

4.1. Qualitative Comparison

In Fig. 3 and Fig. 4, we show tracking results of the 7 trackers on a subset of the videos. The details are introduced as follows.

In the *AF1* sequence, a player is tracked with appearance changes due to camera motion. Tracking results for frames $\{10, 162, 300, 400\}$ are presented in Fig. 3(a). IVT and MIL start to drift around frame 162. Due to changes in appearance, OAB and L_1 start to undergo target drift from frame 300. Frag starts to fail after frame 400. TOD and RCT can track the target through the whole sequence; however, these tracks are not as robust or accurate as the TOD tracker.

For the *AF2* sequence, the player is subject to changes in illumination and pose. Based on the results in Fig. 3(b), OAB, RCT, and L_1 start to drift from the target at frame 200, while MIL and Frag drift at frame 277 and finally lose the target. IVT tracks the target quite well with a little drift. However, the target is successfully tracked throughout the entire sequence by TOD.

In *So1* shown in Fig. 3(c), a player with white color is tracked. The results at 4 frames are shown in Fig. 3(c). Because there is only minor occlusion by other players, most of the methods can track the face accurately except Frag , which drifts around frame 170.

The *So2* sequence contains abrupt object and camera motion with significant scale changes, which cause most of the trackers to drift as shown in Fig. 3(d). TOD, L_1 and RCT handle these changes well. Compared with L_1, TOD obtains much better performance, which shows that harnessing local structure between pixels is useful for object tracking.

In the *So3* sequence, tracking results for frames $\{1, 27, 92, 230\}$ are presented in Fig. 3(e). Frag and IVT start to drift around frame 27 and 92, respectively. Due to changes in lighting and camera motion, most of the trackers drift including L_1 and OAB. TOD, MTT and RCT can track the target through the whole sequence; however, the proposed TOD tracker shows the best performance.

Results on the *faceocc2* sequence are shown in Fig. 4(a). Most trackers start drifting from the man's face when it is almost fully occluded by the book. Because the L_1 and TOD methods explicitly handle partial occlusions, and update the object dictionary progressively, they handle the appearance changes in this sequence very well.

Fig. 4(b) shows tracking results for the *girl* sequence. Performance on this sequence exemplifies the robustness of TOD to occlusion (complete occlusion of the girl's face as she swivels in the chair) and large pose change (the face undergoes significant 3D rotation). TOD and L_1 are capable of tracking the target during the entire sequence. Other trackers experience drift at different time instances.

Fig. 4(c) shows tracking results for the *onelsr1* sequence. In this sequence, partial occlusion happens, and it is much more easier. Therefore, many trackers (except OAB) can track the target through the whole video sequence.

In the *onelsr2* sequence (refer to Fig. 4(d)), the walking woman is partially occluded by a walking man. IVT, MIL, Frag, OAB, and RCT lose the target woman, start tracking the man when partial occlusion occurs around frame 200, and are unable to recover from this failure. TOD and L_1 track the woman quite well.

In the *tud_crossing* sequence, the target is severely occluded by multiple humans as shown in Fig. 4(e). RCT and MIL start to drift around frame 32. Due to multiple occlusions, IVT starts to undergo target drift from frame 83. Oth-

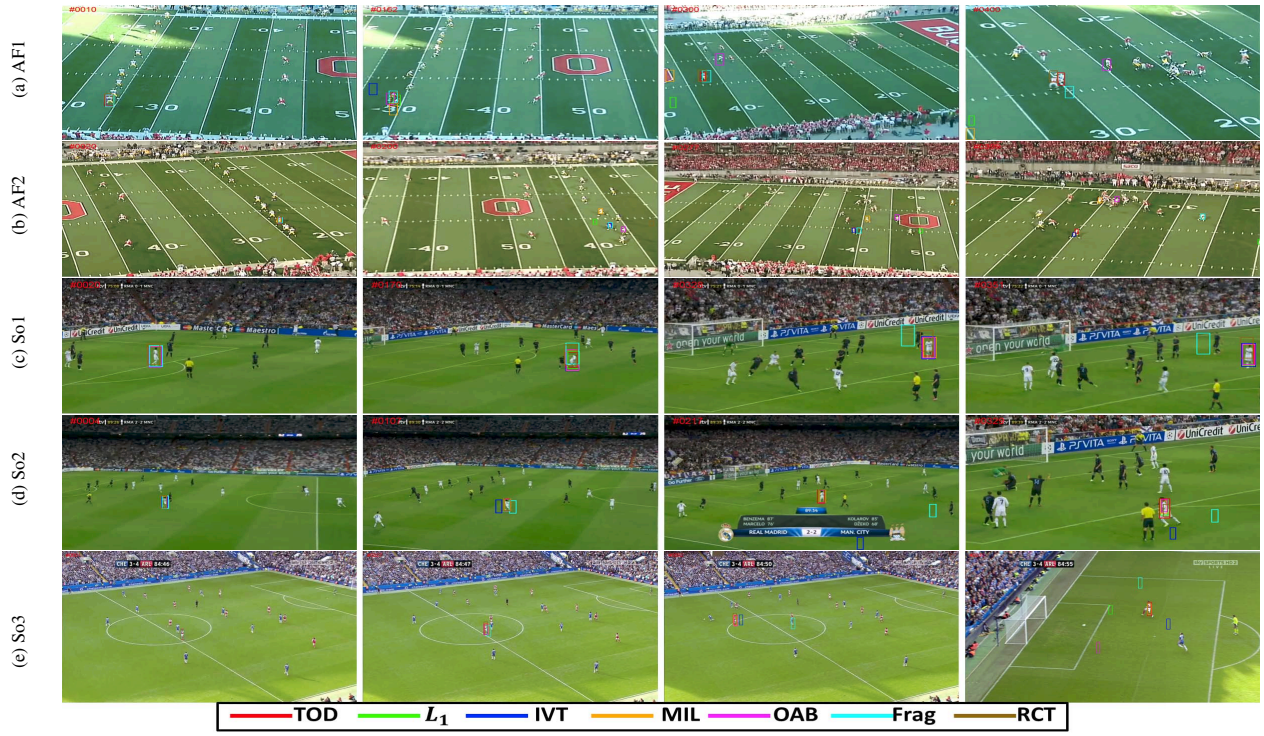

Figure 3. Tracking results of 7 methods on 5 sports video sequences. Frame numbers are denoted in red and the 7 tracking results (bounding boxes) are color-coded in each frame.

er trackers, TOD, L_1, and Frag can track the target through the whole video; however, among all of the trackers, the proposed TOD shows the best performance.

4.2. Quantitative Comparison

To give a fair quantitative comparison among the 7 trackers, we obtain manually labeled ground truth tracks for all the sequences. Most of the ground truth can be downloaded with the sequences. Tracking performance is evaluated according to the average per-frame distance (in pixels) between the center of the tracking result and that of ground truth as used in [2, 14, 4]. Clearly, this distance should be small. In Fig. 5, the average center distance for each tracker over the 10 sequences is plotted. TOD consistently outperform the other trackers in all sequences except for *AF2* and *onelsr1*, where they obtain very similar results to IVT. OAB is effected by background clutter and easily drifts from the target. MIL performs well except under severe illumination changes. RCT is not stable on several video sequences, especially those that contain occlusion and illumination variations. Frag and L_1 handle partial occlusion well, but tend to fail under severe illumination and pose changes. IVT is hardly affected by parameter settings and obtains good results in the absence of severe illumination changes. TOD can consistently produce a smaller distance than other trackers. This implies that TOD can accurately track the target despite severe occlusions and pose variations.

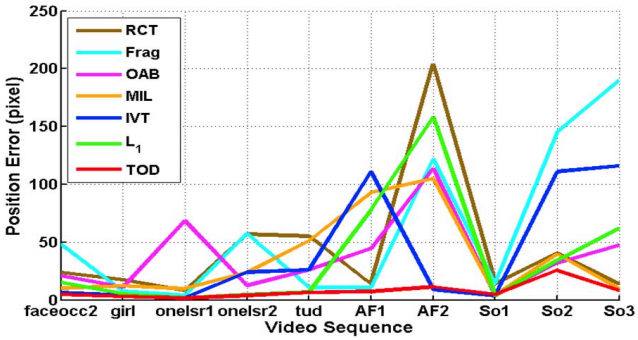

Figure 5. Average distance of 7 trackers applied to 10 sequences

Now, we compare TOD with the L_1 and RCT trackers, which are the most related trackers to ours based on sparse learning and have shown state-of-the-art performance [14, 22]. Based on the results in Fig. 5, TOD outperform the L_1 tracker and RCT. This is primarily due to the use of structure information for occlusion modeling, which makes TOD robust to occlusion problem. In addition, about the computational cost, TOD is much more efficient than L_1 as discussed in Section 3.

5. Conclusion

In this paper, we propose a novel tracking method that allows for occlusion modeling and detection via structured sparse learning. By considering the structural information

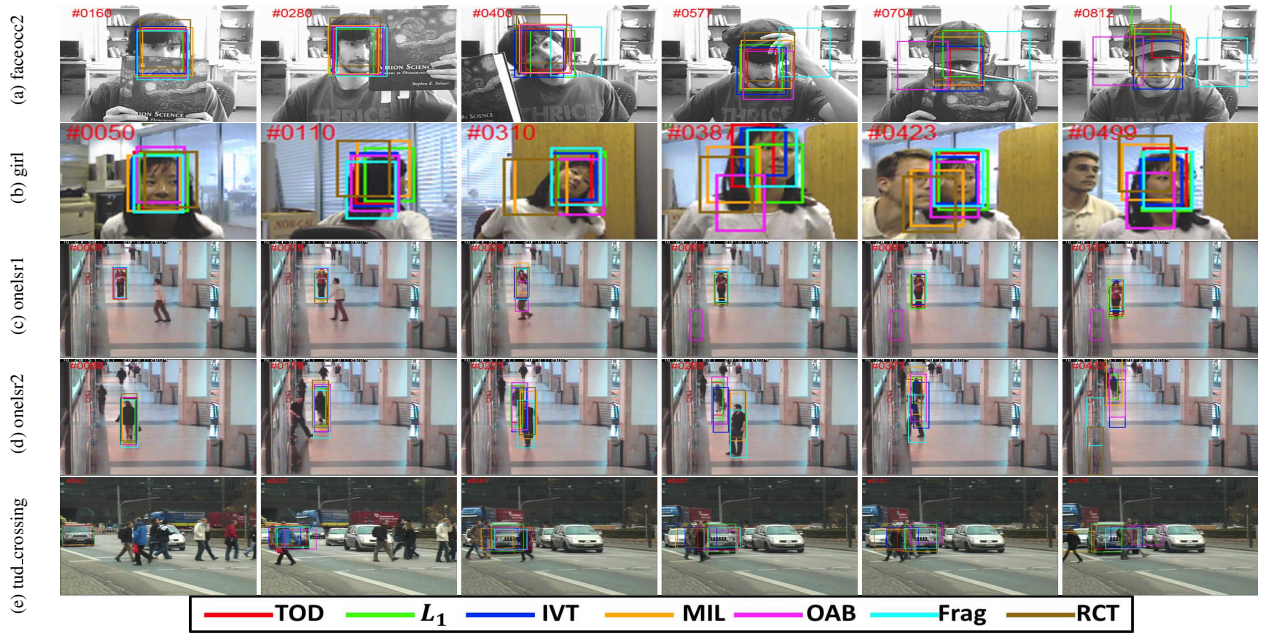

Figure 4. Tracking results of 7 methods on 5 general video sequences. Frame numbers are denoted in red and the 7 tracking results (bounding boxes) are color-coded in each frame.

inherent to occlusion (e.g. spatial contiguity), the proposed TOD is much more robust for tracking under occlusion. The structured sparse learning problem is solved using an efficient IALM method. We show that the popular L_1 tracker [14] is a special case of our formulation. Also, we extensively analyze the performance of our tracker on challenging real-world video sequences and show that it outperforms 6 state-of-the-art trackers.

Acknowledgment

This study is supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR).

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, 2009.

[3] P. Chockalingam, N. Pradeep, and S. Birchfield. Adaptive fragmentsbased tracking of non-rigid objects using level sets. In *ICCV*, 2009.

[4] T. B. Dinh, N. Vo, and G. Medioni. Context Tracker: Exploring Supporters and Distracters in Unconstrained Environments. In *CVPR*, 2011.

[5] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Trans. PAMI*, 30(2):267–282, 2008.

[6] V. Gay-Bellile, A. Bartoli, and P. Sayd. Direct estimation of nonrigid registrations with image-based self-occlusion reasoning. *IEEE Trans. PAMI*, 32(1):87–104, 2010.

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

[8] B. Han and L. Davis. On-line density-based appearance modeling for object tracking. In *ICCV*, 2005.

[9] T. Kaneko and O. Hori. Feature selection for reliable tracking using template matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 796–802, 2003.

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

[11] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. In *Technical Report UILU-ENG-09-2214, UIUC*, August 2009.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[13] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *TPAMI*, 26:810–815, 2004.

[14] X. Mei and H. Ling. Robust Visual Tracking and Vehicle Classification via Sparse Representation. *TPAMI*, 33(11):2259–2272, 2011.

[15] A. Mittal and L. S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *IJCV*, 51(3):189–203, 2003.

[16] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. RASL: Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images. *TPAMI (to appear)*, 2011.

[17] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. *IJCV*, 77(1):125–141, 2008.

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

[19] M. Yang, J. Yuan, and Y. Wu. Spatial selection for attentional visual tracking. In *CVPR*, 2007.

[20] M. Yang, Y.Wu, and G. Hua. Context-aware visual tracking. *PAMI*, 31(1):1195–1209, 2009.

[21] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13–58, Dec. 2006.

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

[23] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *ECCV*, 2012.

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

[25] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2013.