Deep Metric Learning for Visual Tracking

Junlin Hu, Jiwen Lu, *Senior Member, IEEE*, and Yap-Peng Tan, *Senior Member, IEEE*

***Abstract*— In this paper, we propose a deep metric learning (DML) approach for robust visual tracking under the particle filter framework. Unlike most existing appearance-based visual trackers, which use hand-crafted similarity metrics, our DML tracker learns a nonlinear distance metric to classify the target object and background regions using a feed-forward neural network architecture. Since there are usually large variations in visual objects caused by varying deformations, illuminations, occlusions, motions, rotations, scales, and cluttered backgrounds, conventional linear similarity metrics cannot work well in such scenarios. To address this, our proposed DML tracker first learns a set of hierarchical nonlinear transformations in the feed-forward neural network to project both the template and particles into the same feature space where the intra-class variations of positive training pairs are minimized and the interclass variations of negative training pairs are maximized simultaneously. Then, the candidate that is most similar to the template in the learned deep network is identified as the true target. Experiments on the benchmark data set including 51 challenging videos show that our DML tracker achieves a very competitive performance with the state-of-the-art trackers.**

***Index Terms*— Deep learning, metric learning, visual tracking.**

I. Introduction

VISUAL tracking is one of the most important topics in computer vision due to its wide potential applications, such as motion analysis, video surveillance, and human–computer interaction. While extensive efforts have been devoted and a large number of trackers have been developed over the past two decades [1]–[19], generic visual object tracking still remains a challenging problem in computer vision due to large appearance variations in visual objects such as varying deformations, illuminations, out-of-plane rotations, occlusions, and cluttered backgrounds (see more detailed discussions in [20] and [21]).

Most existing visual tracking systems usually employ hand-crafted similarity metrics for template matching, such as the Euclidean distance [5], Matusita metric [22], Bhattacharyya coefficient [23], and Kullback–Leibler [24] and information-theoretic divergences [25]. However, predefined similarity metrics are not precise enough to measure the target distribution in the feature space because there are usually large variations on visual objects so that there are significant appearance differences on visual objects in successive frames. To address this problem, several works have been done to learn a discriminative similarity measure using advanced metric learning approaches for visual tracking [8], [15], [26]–[28]. For example, Jiang *et al.* [28] proposed a sparsity-regularized metric learning method, Li *et al.* [8] introduced an online reservoir metric learning method for appearance-based visual tracking, and Wu *et al.* [15] presented a metric-learning-based structural appearance model (MLSAM) for structure object representation and matching, respectively. However, these methods mainly learn a Mahalanobis distance metric for object matching between successive frames. While discriminative information can be exploited, these approaches usually learn a single linear transformation to project data points into another feature space such that they may not be powerful enough to exploit the nonlinear relationship of data. To address this nonlinearity problem, the kernel method is often adopted to map data points into a high-dimensional subspace, and then conventional metric learning methods are employed to learn a distance metric in this high-dimensional subspace. Unfortunately, these kernel-based methods cannot explicitly achieve the nonlinear mapping function in most cases such that they suffer from the scalability problem.

Generally, the distribution of objects is usually in a nonlinear manifold due to various variations such as deformations, illuminations, and occlusions, and hence, it is desirable to employ nonlinear discriminative methods to exploit such information. In this work, we introduce a new deep metric learning (DML) approach for robust visual tracking under the particle filter framework. Unlike existing metric-learning-based visual tracking approaches, the proposed DML tracker can explicitly learn several hierarchical nonlinear transformations to map data points into another subspace via a feed-forward neural network architecture so that these nonlinear transformations are explicitly solved by maximizing the interclass variations of negative pairs and minimizing the intra-class variations of positive pairs simultaneously. Fig. 1 shows the main procedure of our tracking approach. Experiments on 51 challenging videos show the effectiveness of our proposed tracking approach.

II. Related Work

In this section, we briefly review some representative works on visual tracking and deep learning, respectively.

Manuscript received January 21, 2015; revised May 2, 2015 and June 29, 2015; accepted August 31, 2015. Date of publication September 11, 2015; date of current version October 27, 2016. This work was supported by the Rapid-Rich Object Search Laboratory through the National Research Foundation, Singapore, under its Interactive Digital Media within the Strategic Research Programme. This paper was recommended by Associate Editor P. Salembier. *(Corresponding author: Jiwen Lu.)*

J. Hu and Y.-P. Tan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: jhu007@e.ntu.edu.sg; eyptan@ntu.edu.sg).

J. Lu is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: elujiwen@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2477936

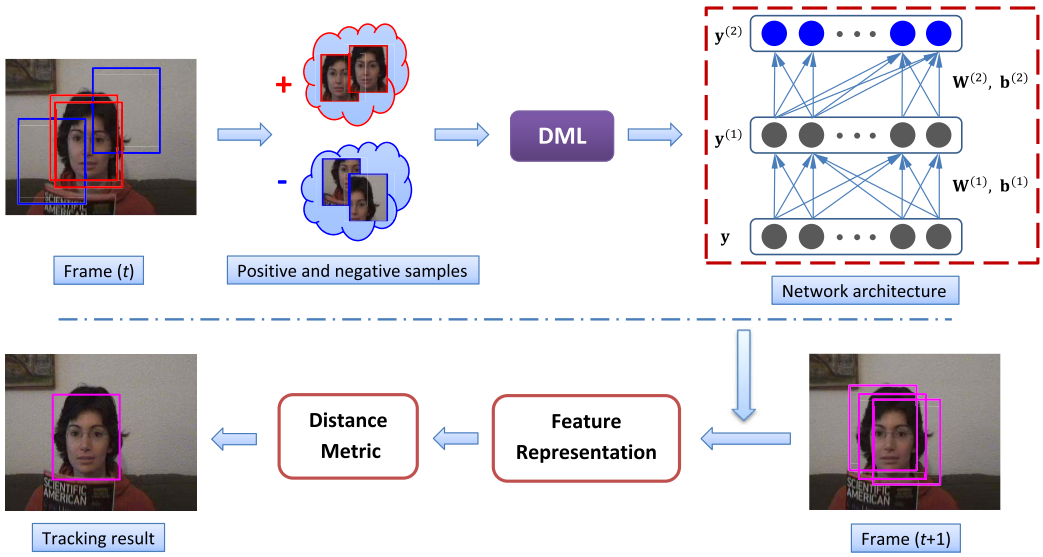


Fig. 1. Main procedure of the proposed DML tracker. Our tracker first samples a training set consisting of positive (red rectangle) and negative (blue rectangle) samples from the tth frame to learn a set of hierarchical nonlinear transformations in a deep network architecture. When the $(t+1)$th frame comes, our tracker maps the candidates (particles) sampled from this frame into the top most level of the network by using hierarchical nonlinear transformations to measure the confidence of all samples. Lastly, the candidate particle that has the maximum confidence to template is determined as the tracked object in this frame.

A. Visual Tracking

Numerous visual tracking methods have been proposed over the past decades and they can be roughly categorized into two classes: 1) generative and 2) discriminative. Generative trackers [1]–[3], [5], [7], [10], [29], [30] learn an appearance model to represent the target object and search the best candidate that has the maximal similarity score with the templates. Representative trackers in this category include incremental visual tracking (IVT) [3], ℓ_1 tracker (L1T) [5], multitask tracker (MTT) [31], and least soft-threshold squares tracker (LSST). The IVT [3] uses incremental principal component analysis (PCA) to model the object appearance changes in videos. L1T [5] represents the object, using a sparse linear combination of numerous trivial templates and several object templates, but the computational complexity of L1T is high. To address this, many extensions of L1T have been proposed to improve the tracking performance. For example, MTT models the interdependencies of sampled particles under the sparse representation framework, and LSST [10] models the error term with the Gaussian–Laplacian distribution based on a linear regression algorithm in L1T. Recently, Wang *et al.* [30] introduced an online nonnegative dictionary learning method to update the object templates. For this category, sparse representation and dictionary-learning-based tracking methods [5], [10], [30], [31] have achieved encouraging results for visual tracking.

Unlike generative trackers, discriminative trackers [4], [9], [32]–[37] formulate visual tracking as a binary classification task that aims to discriminate the target object from the surrounding backgrounds, and most of the trackers in this category employ the tracking-by-detection framework. State-of-the-art discriminative trackers are multiple instance learning (MIL) tracker [4], compressive tracker (CT) [9], online discriminative feature selection (ODFS) [35], Struck [33], circulant structure with kernel (CSK) [36], kernelized correlation filter (KCF) [19], and structure-preserving object tracker (SPOT) [37]. The MIL tracker uses an online MIL method to classify the positive and negative bags. The CT adopts a sparse learning model to compress features from the foreground targets and backgrounds. The ODFS tracker directly computes the classifier score by using a supervised learning method to improve the MIL tracker. The Struck tracker utilizes a kernelized structured output support vector machine in the tracking-by-detection framework for adaptive visual tracking. The CSK tracker explores the circulant structure of the kernel matrix for fast tracking and detection with dense sampling strategy. The SPOT is a model-free tracker, which incorporates spatial constraints between multiple objects or multiple parts in single-object tracking. In this paper, we present a DML approach from a perspective of distance metric learning with a feed-forward neural network for visual object tracking by exploiting the merits of both generative and discriminative trackers.

B. Deep Learning

In recent years, deep learning has received much attention in the research field of machine learning and computer vision due to its excellent performance in learning hierarchical feature representations directly from raw data, and many deep learning approaches have been introduced in [38]–[42]. Recent advances in visual analysis have shown that deep learning has been successfully employed in many computer vision applications such as human action recognition [42], object recognition [39], image classification [43], and face verification [44]. A number of deep learning approaches have been proposed in feature engineering, and representative

methods include deep belief network [38], stacked denoising autoencoder (SDAE) [45], and deep convolutional neural networks (CNN) [43], [46]. However, most of them aim to learn feature representations rather than distance metric with a deep architecture. To address this, several metric learning methods using deep neural network architecture have been proposed recently for face verification tasks such as deep nonlinear metric learning via independent subspace analysis (DNLML-ISA) [47] and discriminative DML (DDML) [48]. Specifically, DNLML-ISA learns a nonlinear distance metric using the stacked independent subspace analysis network, and DDML explicitly obtains the nonlinear distance metric by utilizing a large margin criterion at the top most layer of a deep neural network.

Deep learning has also been exploited in visual tracking in recent years. For example, Nowlan and Platt [49] utilized CNN for hand tracking and identifying whether the hand is open or closed. Fan *et al.* [50] broadened the common use of CNN by a multipath strategy to alleviate the drift problem in human tracking, where they extracted spatial and temporal features for specific object. Jin *et al.* [51] employed the CNN to obtain multidimensional feature vector, and then the feature vector was fed into a radial basis function network to produce confidence map to address tracking problems. Li *et al.* [17], [18] proposed using CNN to learn discriminative features for robust visual tracking. Wang and Yeung [11] proposed a deep learning tracker (DLT) to learn deep compact image representations for object tracking. While encouraging performance can be obtained, DLT requires a very large auxiliary data set to learn offline feature representations using SDAE [45]. This auxiliary data set may be inconsistent to the objects captured online so that the learned features are not adaptive to the objects. Moreover, the offline training is time consuming. Unlike these deep-learning-based trackers, in this paper, we present a DML approach to learn a set of hierarchical nonlinear transformations for visual tracking. Our tracker achieves a competitive performance with the state-of-the-art trackers on 51 challenging videos.

III. Particle Filter for Visual Tracking

The particle filter [3], [11], [52] is a Bayesian sequential importance sampling technique for visual tracking. It estimates the posterior distribution of state variables to characterize a dynamic system based on a sequence of observations. There are two steps in the particle filter framework: 1) prediction and 2) update. Let $\mathbf{s}_t$ and $\mathbf{y}_t$ be the latent state variable and the observation variable of an object at time t, respectively. The aim of visual object tracking is to predict the target state variable $\mathbf{s}_t$, given all available observations $\mathbf{y}_{1:t-1} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}\}$ up to the time $t-1$, employing the following maximum *a posteriori* estimation:

$$\begin{aligned}\hat{\mathbf{s}}_t &= \underset{\mathbf{s}_t^i}{\operatorname{argmax}}\ p\left(\mathbf{s}_t^i|\mathbf{y}_{1:t-1}\right) \\ &= \underset{\mathbf{s}_t^i}{\operatorname{argmax}}\ \int p\left(\mathbf{s}_t^i|\mathbf{s}_{t-1}\right) p(\mathbf{s}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{s}_{t-1} \end{aligned} \tag{1}$$

where $\mathbf{s}_t^i$ is the ith sample (or particle) of the state $\mathbf{s}_t$.

When a new observation $\mathbf{y}_t$ is available at time t, the posterior distribution of the state variable is recursively updated according to the Bayes rule as

$$p(\mathbf{s}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{s}_t)\, p(\mathbf{s}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}. \tag{2}$$

In the particle filter framework, the true posterior state distribution $p(\mathbf{s}_t|\mathbf{y}_{1:t})$ in (2) can be approximated by a finite set of particles $\mathbf{S}_t = \{\mathbf{s}_t^i\}_{i=1}^N$ with the corresponding importance weights $\mathbf{w}_t = \{w_t^i\}_{i=1}^N$, where N is the number of particles and the sum of vector $\mathbf{w}_t$ is 1. The particles $\{\mathbf{s}_t^i\}_{i=1}^N$ are sampled from an importance distribution $q(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{y}_{1:t})$ and weights $\mathbf{w}_t$ are updated as follows:

$$w_t^i = w_{t-1}^i \frac{p\left(\mathbf{y}_t|\mathbf{s}_t^i\right) p\left(\mathbf{s}_t^i|\mathbf{s}_{t-1}^i\right)}{q(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{y}_{1:t})}. \tag{3}$$

For simplicity, the importance distribution $q(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{y}_{1:t})$ is assumed to follow a first-order Markov process and the state transitional probability is $p(\mathbf{s}_t|\mathbf{s}_{t-1})$. In this case, the weight w_t^i in (3) is updated as $w_t^i = w_{t-1}^i p(\mathbf{y}_t|\mathbf{s}_t^i)$, where $p(\mathbf{y}_t|\mathbf{s}_t)$ is the observation likelihood to reflect the similarity of an observed particle $\mathbf{y}_t$ and the object templates.

A. Dynamical Model

For the visual tracking task, the state variable $\mathbf{s}_t$ is often represented as six affine transformation parameters: 1) *horizontal translation*; 2) *vertical translation*; 3) *scale*; 4) *angle*; 5) *aspect ratio*; and 6) *skewness* at time t, respectively. The state transition distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is modeled by a zero-mean Gaussian distribution, and the six parameters of $\mathbf{s}_t$ are assumed to be independent. Namely, $p(\mathbf{s}_t|\mathbf{s}_{t-1}) = \mathcal{N}(\mathbf{s}_t; \mathbf{s}_{t-1}, \Delta)$, here Δ is a diagonal covariance matrix with the variances of these six parameters in diagonal direction.

B. Observation Model

For each frame in a target video, the tracked result is taken to be the particle with the largest weight. Therefore, finding a good observation model $p(\mathbf{y}_t|\mathbf{s}_t)$ is a key issue under the framework of particle filter. In this paper, we utilize a new observation model by training a deep neural network that can learn several hierarchical nonlinear transformations to map both the template and particles into the same feature subspace to calculate their similarity (or likelihood) as follows:

$$p(\mathbf{y}_t|\mathbf{s}_t) = \frac{1}{\Gamma} \exp\left(-\gamma\, d_f^2(\mathbf{y}_t, \mathbf{m}_t)\right) \tag{4}$$

where Γ is a normalization factor, γ is a constant that controls the shape of the Gaussian kernel (e.g., $\gamma = 0.01$), $\mathbf{m}_t$ is the object template updated by a certain scheme, and $d_f^2(\mathbf{y}_t, \mathbf{m}_t)$ denotes the distance under the nonlinear distance metric learned by our proposed DML tracker in the top most layer of a deep neural network, which will be detailed in the following section.

IV. DML Tracker

In this section, we first present the formulation and optimization of the proposed DML tracker and then introduce its implementation details for visual tracking.

A. Deep Metric Learning

Conventional metric learning methods mainly learn the Mahalanobis distance metric by seeking a linear transformation with certain constraints, which are not powerful enough to capture the nonlinear relationship of data points. To address this nonlinearity issue, the kernel trick is widely utilized to implicitly project data points into a high dimensional subspace, and then common metric learning methods are employed to obtain a favorable distance metric in the transformed subspace. While these kernel-based methods can provide a feasible solution, they still suffer from the scalability problem because it is difficult to obtain the explicit nonlinear mapping functions. To address both the nonlinearity and scalability problems, we present a new DML approach to learn several hierarchical nonlinear transformations by using a feed-forward neural network architecture.

1) Data Preparation: The training data consist of N_p positive samples and N_n negative samples. For positive samples, we sample N_p image patches around the target within a radius of a few pixels (e.g., two pixels) by following a zero-mean Gaussian distribution, where the diagonal covariance matrix of six parameters is diag([1, 1, 0, 0, 0, 0]), and resize them into the same size (e.g., 32×32). Each image patch is further flattened into a column vector. Following the similar procedure, the N_n negative samples are sampled far away from the target object by following a zero-mean Gaussian distribution with the diagonal covariance matrix diag($[\omega_{\text{target}}, h_{\text{target}}, 0, 0, 0, 0]$), where ω_{target} and h_{target} are the width and height of the target object (or image patch). Hence, some negative samples may contain both the background and parts of the target object. Having obtained the training set, we randomly make up $\mathcal{P}$ positive pairs and $\mathcal{N}$ negative pairs to learn the parameters of our deep network. Each positive pair consists of two randomly selected positive samples. And each negative pair contains one randomly selected positive sample and one randomly picked negative sample.

2) Network Architecture: As shown in the top right corner of Fig. 1, we first construct a deep neural network to learn the nonlinear representation of a sample $\mathbf{y}$ by passing it through multiple layers of nonlinear transformations. Assume that there are $\mathcal{K}+1$ layers in our designed network and $r^{(k)}$ units in the kth layer, where $k = 1, 2, \ldots, \mathcal{K}$. Given a sample (or particle) $\mathbf{y} \in \mathbb{R}^{r^{(0)}}$ with a size of $r^{(0)}$, its output from the first layer is $\mathbf{y}^{(1)} = \varphi(\mathbf{W}^{(1)}\mathbf{y} + \mathbf{b}^{(1)}) \in \mathbb{R}^{r^{(1)}}$, where $\mathbf{W}^{(1)} \in \mathbb{R}^{r^{(1)} \times r^{(0)}}$ is a projection matrix of the first layer[1]; $\mathbf{b}^{(1)} \in \mathbb{R}^{r^{(1)}}$ is a bias vector; and $\varphi : \mathbb{R} \mapsto \mathbb{R}$ is a nonlinear activation function operated on componentwise, e.g., *sigmoid* and *tanh* functions. Then, we use the output of the first layer $\mathbf{y}^{(1)}$ as the input of the second layer. Following this procedure, the output of the kth ($k > 1$) layer can be given as:

$$\mathbf{y}^{(k)} = \varphi(\mathbf{W}^{(k)}\mathbf{y}^{(k-1)} + \mathbf{b}^{(k)}) \in \mathbb{R}^{r^{(k)}} \tag{5}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^{r^{(k)} \times r^{(k-1)}}$, $\mathbf{b}^{(k)} \in \mathbb{R}^{r^{(k)}}$, and φ are the projection, bias, and activation function of the kth layer, respectively. Finally, the output of the top most layer of network can be represented as

$$f(\mathbf{y}) = \mathbf{y}^{(\mathcal{K})} = \varphi(\mathbf{W}^{(\mathcal{K})}\mathbf{y}^{(\mathcal{K}-1)} + \mathbf{b}^{(\mathcal{K})}) \in \mathbb{R}^{r^{(\mathcal{K})}} \tag{6}$$

where the mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^{r^{(\mathcal{K})}}$ is a parametric nonlinear function which is jointly determined by the parameters $\{\mathbf{W}^{(k)}\}_{k=1}^{\mathcal{K}}$ and $\{\mathbf{b}^{(k)}\}_{k=1}^{\mathcal{K}}$.

Given a pair of particles $\mathbf{y}_i$ and $\mathbf{y}_j$, we pass them into $\mathcal{K}$ hierarchical nonlinear transformations and represent them as $f(\mathbf{y}_i) = \mathbf{y}_i^{(\mathcal{K})}$ and $f(\mathbf{y}_j) = \mathbf{y}_j^{(\mathcal{K})}$ at the top most layer, respectively. Then, their similarity is measured by calculating the squared Euclidean distance between the representations $f(\mathbf{y}_i)$ and $f(\mathbf{y}_j)$ in the designed deep neural network

$$d_f^2(\mathbf{y}_i, \mathbf{y}_j) = \|f(\mathbf{y}_i) - f(\mathbf{y}_j)\|_2^2. \tag{7}$$

3) Formulation: To learn the parameters $\{\mathbf{W}^{(k)}\}_{k=1}^{\mathcal{K}}$ and $\{\mathbf{b}^{(k)}\}_{k=1}^{\mathcal{K}}$ in the designed network, we adopt the margin fisher analysis criterion [53] at the top most layer of all the training samples and formulate the proposed DML approach as the following optimization problem:

$$\begin{aligned}\min_f \mathcal{O} = \frac{1}{\mathcal{P}} \sum_{\ell_{ij}=1} d_f^2(\mathbf{y}_i, \mathbf{y}_j) - \frac{\alpha}{\mathcal{N}} \sum_{\ell_{ij}=-1} d_f^2(\mathbf{y}_i, \mathbf{y}_j) \\ + \beta \sum_{k=1}^{\mathcal{K}} \left(\|\mathbf{W}^{(k)}\|_F^2 + \|\mathbf{b}^{(k)}\|_2^2\right)\end{aligned} \tag{8}$$

where the first term measures the intra-class compactness of positive pairs, the second term defines the interclass separability of negative pairs, and the last term is the regularization. α is a positive parameter to balance the importance between intra-class compactness and interclass separability; β ($\beta > 0$) is a regularization parameter; and ℓ_{ij} is the pairwise label of a pair of samples $\mathbf{y}_i$ and $\mathbf{y}_j$, which is set as $\ell_{ij} = 1$ if $\mathbf{y}_i$ and $\mathbf{y}_j$ are from a positive pair (i.e., both particles $\mathbf{y}_i$ and $\mathbf{y}_j$ are sampled from the same target object or positive samples) and -1 if they are from a negative pair (i.e., one particle is sampled from the positive sample and another is drawn from the negative sample). The operation $\|\cdot\|_F$ means the Frobenius norm of a matrix, and $\mathcal{P}$ and $\mathcal{N}$ denote the total number of positive pairs and negative pairs in the training data, respectively. Specifically, the DML aims to seek an optimal nonlinear mapping f by minimizing the intra-class variations of positive training pairs and maximizing the interclass variations of negative training pairs in the transformed subspace for utilizing more discriminative information.

4) Optimization: To our best knowledge, the optimization problem in (8) is not convex and it is difficult to obtain the closed-form solution directly. To solve this problem, we use the gradient-descent-based method to solve the parameters $\{\mathbf{W}^{(k)}, \mathbf{b}^{(k)}\}_{k=1}^{\mathcal{K}}$. With some algebraic simplification,

[1]Previous studies have shown that learning a Mahalanobis distance metric is equivalent to seeking a linear transformation to map each sample into a low-dimensional subspace, where the Euclidean distance of two samples in the transformed space is equal to the Mahalanobis distance, namely

$$\begin{aligned}d_{\mathbf{W}}^2(\mathbf{y}_i, \mathbf{y}_j) &= \|\mathbf{W}\mathbf{y}_i - \mathbf{W}\mathbf{y}_j\|_2^2 = (\mathbf{y}_i - \mathbf{y}_j)^T\mathbf{W}^T\mathbf{W}(\mathbf{y}_i - \mathbf{y}_j) \\ &= (\mathbf{y}_i - \mathbf{y}_j)^T\mathbf{M}(\mathbf{y}_i - \mathbf{y}_j) = d_{\mathbf{M}}^2(\mathbf{y}_i, \mathbf{y}_j)\end{aligned}$$

and the positive semidefinite matrix $\mathbf{M} = \mathbf{W}^T\mathbf{W}$ is the learned Mahalanobis distance metric in the original space.

the derivative of the objective function $\mathcal{O}$ with respect to the parameters $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$, $k = 1, 2, \ldots, \mathcal{K}$ can be calculated as follows:

$$\begin{aligned}\frac{\partial \mathcal{O}}{\partial \mathbf{W}^{(k)}} &= \frac{2}{\mathcal{P}} \sum_{\ell_{ij}=1} \left(\mathbf{G}_{ij}^{(k)} \mathbf{y}_i^{(k-1)^T} + \mathbf{G}_{ji}^{(k)} \mathbf{y}_j^{(k-1)^T}\right) \\ &\quad - \frac{2\alpha}{\mathcal{N}} \sum_{\ell_{ij}=-1} \left(\mathbf{G}_{ij}^{(k)} \mathbf{y}_i^{(k-1)^T} + \mathbf{G}_{ji}^{(k)} \mathbf{y}_j^{(k-1)^T}\right) \\ &\quad + 2\beta \mathbf{W}^{(k)} \qquad (9) \\ \frac{\partial \mathcal{O}}{\partial \mathbf{b}^{(k)}} &= \frac{2}{\mathcal{P}} \sum_{\ell_{ij}=1} \left(\mathbf{G}_{ij}^{(k)} + \mathbf{G}_{ji}^{(k)}\right) \\ &\quad - \frac{2\alpha}{\mathcal{N}} \sum_{\ell_{ij}=-1} \left(\mathbf{G}_{ij}^{(k)} + \mathbf{G}_{ji}^{(k)}\right) + 2\beta \mathbf{b}^{(k)} \qquad (10)\end{aligned}$$

where $\mathbf{y}_i^{(0)} = \mathbf{y}_i$ denotes the original input sample and the variables $\mathbf{G}_{ij}^{(\mathcal{K})}$ and $\mathbf{G}_{ji}^{(\mathcal{K})}$ for the top most layer are shown as follows:

$$\mathbf{G}_{ij}^{(\mathcal{K})} = \left(\mathbf{y}_i^{(\mathcal{K})} - \mathbf{y}_j^{(\mathcal{K})}\right) \odot \varphi'\left(\mathbf{x}_i^{(\mathcal{K})}\right) \qquad (11)$$

$$\mathbf{G}_{ji}^{(\mathcal{K})} = \left(\mathbf{y}_j^{(\mathcal{K})} - \mathbf{y}_i^{(\mathcal{K})}\right) \odot \varphi'\left(\mathbf{x}_j^{(\mathcal{K})}\right) \qquad (12)$$

and the intermediate variables $\mathbf{G}_{ij}^{(k)}$ and $\mathbf{G}_{ji}^{(k)}$ for other layers $k = 1, 2, \ldots, \mathcal{K} - 1$ are given by the following updating equations:

$$\mathbf{G}_{ij}^{(k)} = \left(\mathbf{W}^{(k+1)^T} \mathbf{G}_{ij}^{(k+1)}\right) \odot \varphi'\left(\mathbf{x}_i^{(k)}\right) \qquad (13)$$

$$\mathbf{G}_{ji}^{(k)} = \left(\mathbf{W}^{(k+1)^T} \mathbf{G}_{ji}^{(k+1)}\right) \odot \varphi'\left(\mathbf{x}_j^{(k)}\right) \qquad (14)$$

$$\mathbf{x}_i^{(k)} = \mathbf{W}^{(k)} \mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)} \qquad (15)$$

where the operation $\odot$ denotes element-wise multiplication.

Having obtained these derivatives, the parameters $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$, $k = 1, 2, \ldots, \mathcal{K}$ can be updated using the following gradient descent algorithm until convergence:

$$\mathbf{W}^{(k)} = \mathbf{W}^{(k)} - \rho \frac{\partial \mathcal{O}}{\partial \mathbf{W}^{(k)}} \qquad (16)$$

$$\mathbf{b}^{(k)} = \mathbf{b}^{(k)} - \rho \frac{\partial \mathcal{O}}{\partial \mathbf{b}^{(k)}} \qquad (17)$$

where ρ is the learning rate, which controls the convergence speed of objective function $\mathcal{O}$.

Algorithm 1 summarizes the detailed procedure of the proposed DML approach for learning parameters of the deep neural network.

Algorithm 1 DML

Input: Training data: $\mathbf{Y} = \{(\mathbf{y}_i, \mathbf{y}_j, \ell_{ij})\}$; Layers of network: $\mathcal{K} + 1$; Trade-off parameter: α; Regularization parameter: β; The learning rate: ρ; Total iterative number: T; Convergence error ε.

// Optimization procedure of DML

$\mathcal{O}_0 \leftarrow 0$;

Initializing $\{\mathbf{W}^{(k)}, \mathbf{b}^{(k)}\}_{k=1}^{\mathcal{K}}$ according to (18);

for $t = 1, 2, \cdots, T$ **do**

  for $k = 1, 2, \cdots, \mathcal{K}$ **do**

    Computing hierarchical representation $\mathbf{y}_i^{(k)}$ of each sample by using forward propagation;

  end

  // Back propagation

  for $k = \mathcal{K}, \mathcal{K} - 1, \cdots, 1$ **do**

    Obtaining $\partial \mathcal{O} / \partial \mathbf{W}^{(k)}$ and $\partial \mathcal{O} / \partial \mathbf{b}^{(k)}$ in line with (9) and (10), respectively;

  end

  // Updating parameters

  for $k = 1, 2, \cdots, \mathcal{K}$ **do**

    $\mathbf{W}^{(k)} \leftarrow \mathbf{W}^{(k)} - \rho \partial \mathcal{O} / \partial \mathbf{W}^{(k)}$;

    $\mathbf{b}^{(k)} \leftarrow \mathbf{b}^{(k)} - \rho \partial \mathcal{O} / \partial \mathbf{b}^{(k)}$;

  end

  Calculating objective $\mathcal{O}_t$ using (8);

  If $|\mathcal{O}_t - \mathcal{O}_{t-1}| < \varepsilon$, go to **Output**;

end

Output: Weights and biases: $\{\mathbf{W}^{(k)}, \mathbf{b}^{(k)}\}_{k=1}^{\mathcal{K}}$.

B. Implementation Details

This section introduces some implementation details of our DML tracker including the initialization of parameters in neural network, template update strategy, and the online tracking process.

1) Initialization: It is important to initialize parameters $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$, $1 \leq k \leq \mathcal{K}$ in our designed network for obtaining good performance. DAE [54] and its variants are usually employed to pretrain parameters of the deep neural network from a large auxiliary data set. In our experiments, we apply the initialization approach in [55], which adopts a normalized random initialization strategy to initialize $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$ of each layer. The weight $\mathbf{W}^{(k)}$ at each layer is given by a uniform distribution as

$$\mathbf{W}^{(k)} \sim U\left[-\frac{\sqrt{6}}{\sqrt{r^{(k)} + r^{(k-1)}}}, \frac{\sqrt{6}}{\sqrt{r^{(k)} + r^{(k-1)}}}\right]. \qquad (18)$$

The bias $\mathbf{b}^{(k)}$ in this layer is set as $\mathbf{0}$, and $r^{(0)}$ denotes the size of the first input layer.

2) Template Update: The template update is another important issue in visual tracking because the object usually experiences rapid appearance changes during the tracking process. If we use a fixed appearance template from the first frame, it cannot capture well the appearance variations resulting from the changes in illumination, pose, background, and occlusion. If the template is updated quickly, small errors will be introduced in each update, and the tracker drifts from the object since the errors are gradually accumulated. In our work, we use a simple update scheme with an incremental subspace learning method. Specifically, let $\mathbf{y}_t^o$ denote the observation vector corresponding to the best candidate state of the tth frame. The template $\mathbf{m}_t$ is updated from several observation vectors (e.g., $\mathbf{y}_{t'}^o, \ldots, \mathbf{y}_{t-2}^o, \mathbf{y}_{t-1}^o, \mathbf{y}_t^o$) following the mean update of the IVT [3] as:

$$\mathbf{m}_t = \frac{\theta t'}{\theta t' + \tau} \mathbf{m}_{t'} + \frac{\tau}{\theta t' + \tau} \sum_{i=0}^{\tau-1} \mathbf{y}_{t-i}^o \qquad (19)$$

where $\mathbf{m}_{t'}$ is the template updated at the t'th frame, $t' = t - \tau + 1$; the interval τ denotes the number of new observations at each update; and θ, $0 < \theta \leq 1$, is a forgetting factor that balances the contribution between old and new observations and also reduces the effect of the old observations to the resulting template $\mathbf{m}_t$. In our experiments, τ and θ are empirically set as 5 and 0.95, respectively.

3) Online Tracking Process: When a new frame arrives, we first randomly sample N particles around the current state of the tracked object according to a zero-mean Gaussian distribution under the particle filter framework. Then, the confidence (or likelihood) of each particle in (4) is calculated at the top layer of the deep network. The particle with the highest confidence to object template is selected as the tracked object. Having localized the target object, we sample positive and negative pairs to learn a set of nonlinear transformations using the proposed DML method. In our implementations, we learn the parameters of our designed network at every several frames (e.g., ten frames in the experiments).

V. Experiments

To evaluate the performance of the proposed DML tracker, we conducted visual tracking experiments on the benchmark data set [56] which contains 51 challenging video sequences. These sequences present various challenging situations such as illumination variation, partial occlusion, pose change, motion blur, background clutter, scale variation, in-plane rotation, and low resolution. Please refer to the tracking benchmark [56] for more details. We compare our DML tracker with 11 state-of-the-art trackers: 1) tracker using Gaussian processes regression (TGPR) [16]; 2) KCF [19]; 3) Struck [33]; 4) sparse collaborative model (SCM) [57] tracker; 5) CT [9]; 6) IVT [3]; 7) L1APG [58] (a powerful variant of the L1T [5]); 8) MIL [4]; 9) MTT [31]; 10) LSST [10]; and 11) DLT [11]. For a fair comparison, we used the publicly available source codes provided by either the benchmark [56] with the same parameters or the authors with their original parameters.

A. Experimental Setup

For our DML tracker, we train a deep neural network of four layers ($\mathcal{K} = 3$) with the *tanh* activation function, where the tradeoff parameter α, the regularization parameter β, and the learning rate ρ are empirically set at 0.1, 0.01, and 0.01 for all experiments, respectively, unless stated otherwise. For each video, the location of the tracked object in the first frame is manually labeled, and the number of particles is set at 600 for each frame. Then each particle is resized to 32×32 and further flattened into a 1024D vector. To reduce the training and testing time of our method, each feature vector is also reduced to 100 dimension using PCA where the projection matrix is learned from samples collected in the first frame of each video. The units of all the layers in the designed neural network are set at $[r^{(0)}, r^{(1)}, r^{(2)}, r^{(3)}] = [100, 100, 80, 80]$ for all the video sequences. For the training set (see Data Preparation in Section IV for more details), we first sampled $N_p = 20$ positive samples and $N_n = 200$ negative samples, and then generated $\mathcal{P} = 200$ positive pairs and $\mathcal{N} = 800$ negative pairs to learn the parameters of our neural network for every ten frames. In addition, the template is updated every five frames using an incremental mean update scheme proposed in [3]. Regarding the state transition distribution for six affine parameters in the particle filter, their standard deviations are set at $\text{diag}(\Delta) = [4, 4, 0.01, 0, 0.001, 0]$ for all the 51 image sequences. We conduct experiments on a PC whose hardware configuration comprises a 3.2-GHz CPU (Intel i5-3470) and an 8-GB RAM. The proposed DML tracker was implemented in the MATLAB platform, and it runs at 5.53 frames/s on average.

B. Quantitative Evaluation

1) Evaluation Metric: We adopt two widely used evaluation metrics for quantitative comparison: 1) overlapping ratio (OR) and 2) central location error (CLE). The OR metric is defined as $\text{OR} = (\text{area}(B_T \cap B_G)/\text{area}(B_T \cup B_G))$, where B_T is the bounding box of the tracked result and B_G is the ground truth bounding box for each frame. The CLE metric denotes the Euclidean distance (in pixels) between the centers of B_T and B_G. Based on the OR and CLE metrics, *success* plot and *precision* plot [36], [56] are used for evaluating the overall performance of trackers. The success plot shows the ratios of successful frames over the whole video with various thresholds densely sampled in the range $[0, 1]$, where the successful frame indicates that the OR of this tracked frame is larger than the given threshold. In addition, the area under curve (AUC) of each success plot is calculated and ranked for evaluating different trackers. The precision plot simply shows the percentage of frames whose CLE is within a range of distance thresholds (e.g., 0–50 pixels). Hence, an accurate tracker is expected to achieve a higher precision at the given low distance threshold. Following [36] and [56], the precision score at the specific threshold (i.e., 20 pixels) is chosen as the representative precision score for ranking different trackers. Lastly, we adopt the one-pass evaluation (OPE) [56] strategy for evaluating the robustness of various trackers. Specifically, we run these trackers from the ground truth location in the first frame of a test video to the end, and then report the overall tracking performance by both the success and precision plots.

2) Overall Performance: We quantitatively summarize the overall tracking performance of the 12 trackers on all the 51 videos, and Fig. 2 shows the success and precision plots. In the success plot, we make the following observations.

1) The AUC score of our DML tracker is 0.466, which is ranked fifth, where the top four trackers are TGPR (0.529), KCF (0.503), SCM (0.499), and Struck (0.474), respectively. The TGPR obtains the best tracking result; the reason may be that the TGPR exploits the auxiliary data collected from the early frames to assist the tracking decision.
2) The success rate of our tracker is consistently higher than that of Struck when the overlap threshold varies in the range [0.4, 1].
3) For deep-learning-based trackers, our DML tracker outperforms DLT (0.436) by 3.0%.

In the precision plot, our DML tracker also obtains the fifth best results across all location error thresholds

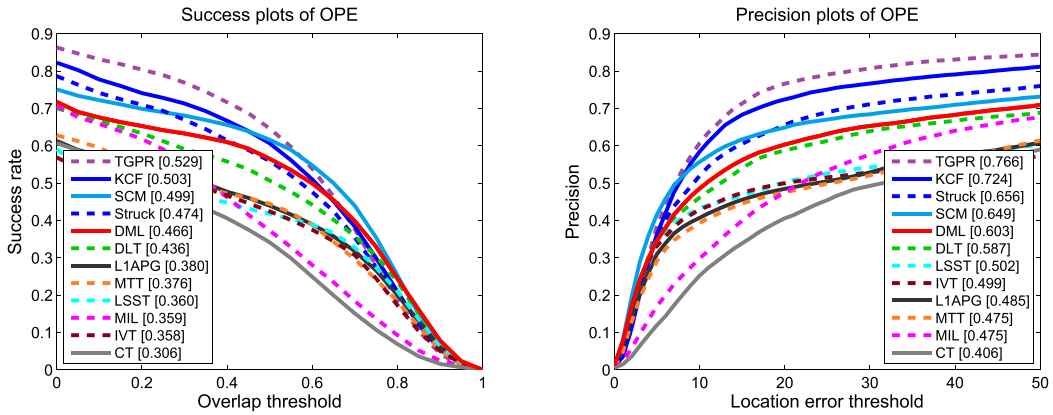


Fig. 2. Success plots and precision plots of the 12 trackers on the 51 sequences for comparing overall performance, respectively. The legend lists the performance score for each tracker. The proposed DML tracker (in red curve) is ranked fifth among these trackers in both the success and precision plots.

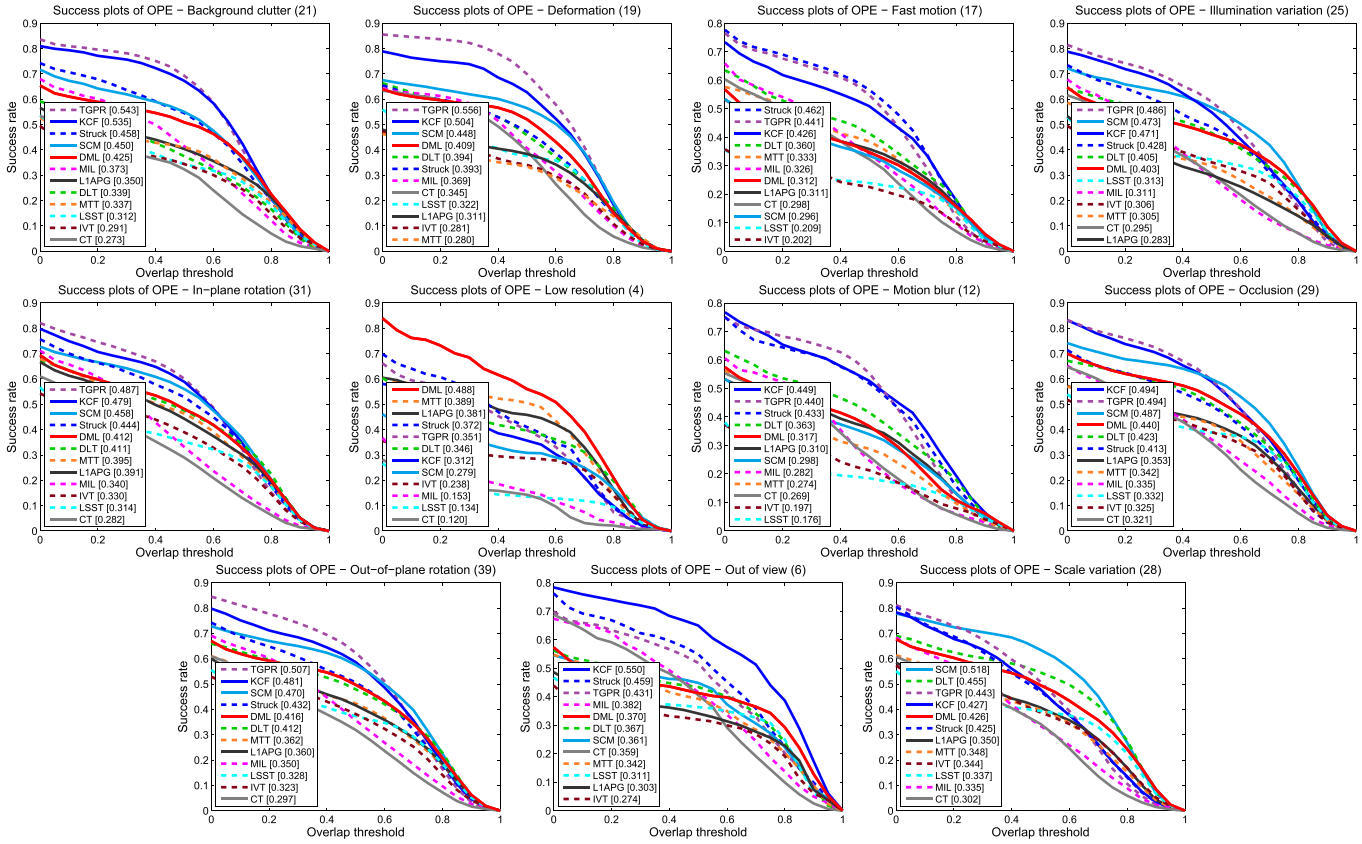


Fig. 3. Success plots of trackers on the 11 subsets for attribute-based performance analysis. The value in bracket is the number of videos in the subset.

from 0 to 50 pixels. Furthermore, we see that the SCM tracker shows a higher precision score than Struck when the distance threshold is less than 20 pixels though the Struck is ranked first at the location error threshold of 20 pixels. Hence, the precision plot is used as an auxiliary evaluation criterion for analyzing the rankings of trackers [56]. Overall, our DML tracker is comparable with these state-of-the-art trackers in both the success and precision plots.

3) Attribute-Based Performance Analysis: The benchmark data set [56] also annotates the dominant attributes of each sequence and brings about 11 subsets for analyzing the performance of trackers with regard to different challenging factors, where each subset consists of some videos sharing a unique attribute. These 11 attributes are background clutters, deformation, fast motion, illumination variation, in-plane rotation, low resolution, motion blur, occlusion, out-of-plane rotation, out of view, and scale variation. We report the success plots and precision plots of different trackers for these 11 attributes in Figs. 3 and 4, respectively. For a low-resolution subset, our DML tracker obtains an AUC of 0.488, which outperforms the second-ranked tracker MTT (0.389) by 10% in the success rate. The reason is

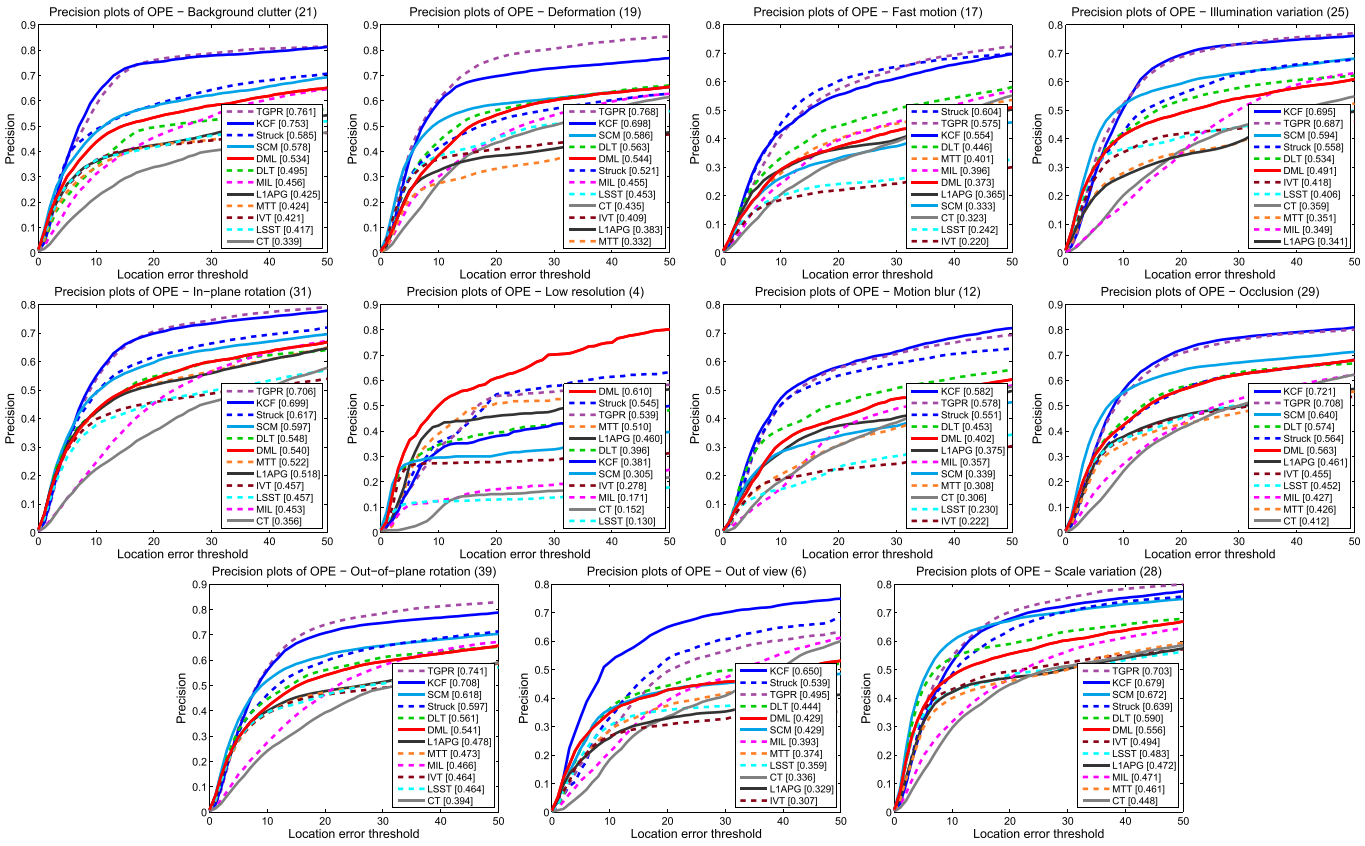


Fig. 4. Precision plots of trackers on the 11 subsets for attribute-based performance analysis. The value in bracket is the number of videos in the subset.

that the DML can learn a discriminative distance metric for template matching. For objects in low-resolution videos, it is not reliable to extract visual features to represent the target objects, and most visual tracking methods significantly degrade their performance. The DML tracker aims to preserve the underlying metric in low-resolution data for discriminative matching, and it focuses more on the similarity measure and the matching function rather than on the object representation. The DML tracker also achieves the fourth place in deformation and occlusion, and the fifth place in other six subsets in terms of the success rate. For fast motion, most of the trackers degrade their performance while Struck shows the best tracking results. This is because Struck is a dense-sampling-based tracker that usually has larger search regions than the trackers in the particle filter framework.

C. Qualitative Comparison

1) Illumination and Scale Changes: We evaluated various trackers on three sequences including *CarDark*, *Singer1*, and *Skating1* with significant illumination variations as shown in Fig. 5. In the *CarDark* sequence, the MIL, DLT, and IVT cannot correctly track the object to the last frame, and they begin to drift from frames #60 and #300 due to continuous illumination variation in appearance. The sequence *Singer1* shows both large scale variation and severe illumination change. MIL, Struck, L1APG, and MTT are not accurately tracking the object because they cannot smoothly handle the large scale variations (e.g., #120 and #310) while the DML, DLT, and SCM trackers work well in this sequence. For *Skating1*, the target object suffers from extreme illumination variations, occlusions, as well as nonrigid deformation. Most of the trackers gradually lose the target object from frames #56 and #180 during tracking, and our DML approach reliably tracks this video to frame #268 which performs slightly better than the DLT and SCM trackers. Generally, the DML tracker achieves the best overlapping rate on these videos, and it is robust to illumination and scale changes.

2) Occlusion and Pose Variation: Fig. 6 shows several sequences with heavy occlusion or long-time partial occlusion and pose variation: 1) *David3*; 2) *Walking2*; 3) *FaceOcc1*; and 4) *Liquor*, respectively. For the sequence *David3*, the MTT first loses target after frame #58, and Struck, SCM, MIL, and DLT drift away from frame #130 due to the first heavy occlusion caused by a tree. The DML tracker successfully tracks the target object through all the frames of this video. For *Walking2*, all the trackers except the MIL and LSST can successfully track the object, and the MIL and LSST lose the object at frame #230 because of similar appearance occlusion. In addition, Struck shows an imperfect performance in dealing with the scale change (e.g., #380 and #495). In the sequence *FaceOcc1*, almost all the trackers obtain favorable results. Regarding video *Liquor*, all the trackers drift from the target object in the whole tracking process. For example, most of the trackers get lost at frame #360, but the proposed DML method still solidly tracks most of the frames (#850).

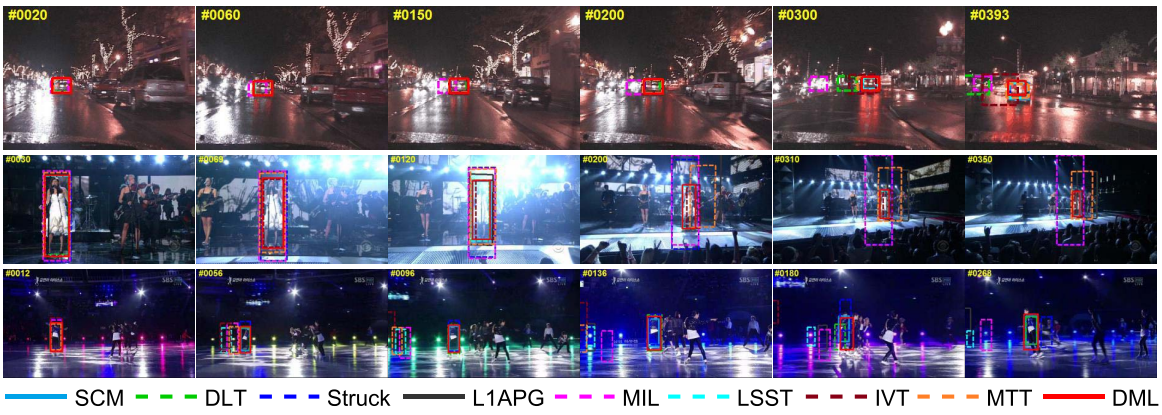


Fig. 5. Sampled tracking results of representative trackers on the sequences with illumination and scale changes. The sequences are *CarDark*, *Singer1*, and *Skating1* from top to bottom. Frame numbers are shown in the top left of each figure.

Fig. 6. Sampled tracking results of representative trackers on the sequences with heavy occlusion. The sequences are *David3*, *Walking2*, *FaceOcc1*, and *Liquor* from top to bottom. Frame numbers are shown in the top left of each figure.

The tracking results on these challenging sequences show that our DML tracker has the ability to deal with occlusion during tracking. Here we highlight an occlusion example (i.e., sequence *FaceOcc1*) to show how DML works in this case. In frame #872, a particle (face) with partial occlusion is mapped into another subspace via the learned nonlinear transformations by the DML approach for discriminative matching, under which some relevant dimensions (without occlusion) are highlighted and some irrelevant ones (with occlusion) are suppressed to ease the effect of occlusion. Thus, our DML tracker can handle partial occlusion even if it only employs the holistic template.

3) Fast Motion and Motion Blur: Fig. 7 illustrates the tracking results on the sequences *Jumping*, *CarScale*, *Ironman*, and *Soccer* with fast motion and motion blur. In the sequence *Jumping*, only Struck, MIL, and DML successfully track the target object from the first frame to the last frame, and the Struck and DML trackers achieve better results in terms of the overlap rate. For sequence *CarScale*, the IVT, DLT, and DML show better tracking performance than others at frame #170; however, they only locate partial target object after frame #200 due to large scale variation caused by fast motion. In the sequences *Ironman* and *Soccer*, the DML method fails to correctly track the target object after some frames, and other trackers also drift away the targets during tracking because the tracked objects undergo several variations such as illumination, occlusion, fast motion, and motion blur. Overall, fast motion and motion blur are two very challenging factors for the DML tracker.

4) Deformation: Fig. 8 shows the tracking results on two sequences *Bolt* and *Singer2* with nonrigid deformation. In the sequence *Bolt*, most of the trackers lose the target object from frame #30, and the proposed DML tracker can smoothly follow up this sequence to frame #200 even though this

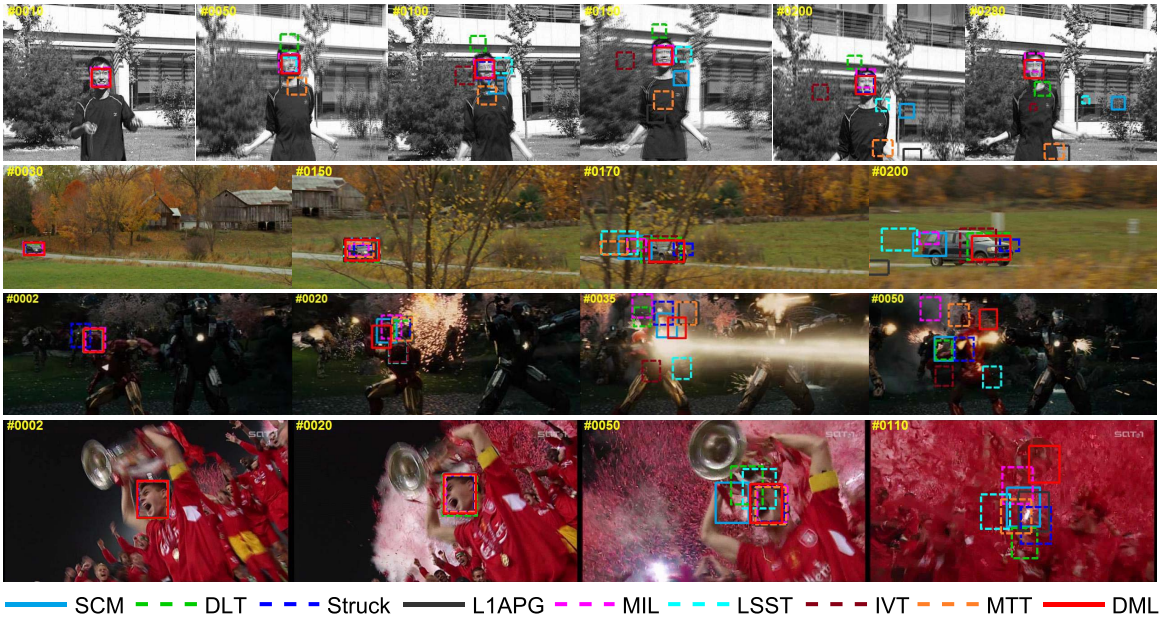


Fig. 7. Sampled tracking results of representative trackers on the sequences with fast motion and motion blur. The sequences are *Jumping*, *CarScale*, *Ironman*, and *Soccer* from top to bottom. Frame numbers are shown in the top left of each figure.

Fig. 8. Sampled tracking results of representative trackers on the sequences with nonrigid deformation. The sequences are *Bolt* and *Singer2* from top to bottom. Frame numbers are shown in the top left of each figure.

sequence undergoes large shape deformation. The sequence *Singer2* depicts that a singer goes through both changes in body shape and significant illumination variations. Obviously, three trackers (MIL, LSST, and DML) focus the target on most of the frames and the DML track shows the precise tracking performance. These tracking results show that the DML tracker can handle some deformation, the reason for which is that the DML method aims to preserve the underlying metric of data points for discriminative matching by seeking a nonlinear mapping to reduce the effect of deformation in another feature space.

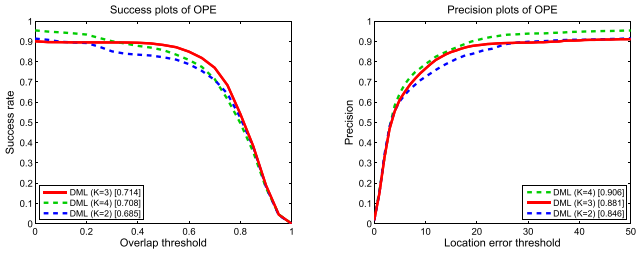


Fig. 9. Success plots and precision plots of the DML tracker with different layers of network on the ten video sequences.

D. Performance Analysis

In this section, we investigate several factors that affect the performance of the proposed DML tracker. Specifically, we selected a subset consisting of ten video sequences from the benchmark data set, and these sequences are *Car4*, *CarDark*, *Crossing*, *David3*, *Dudek*, *FaceOcc1*, *Mhyang*, *Singer1*, *Skating1*, and *Walking2*, respectively.

1) Effect of Different Layers of Network: To evaluate how the number of layers in our deep network affects the tracking performance of the proposed approach, we compare our DML tracker with other two different structures: 1) three layers ($\mathcal{K} = 2$) with units $[r^{(0)}, r^{(1)}, r^{(2)}] = [100, 100, 80]$ and 2) five layers ($\mathcal{K} = 4$) with units $[r^{(0)}, r^{(1)}, r^{(2)}, r^{(3)}, r^{(4)}] = [100, 100, 80, 80, 80]$, respectively. Fig. 9 shows the performance comparison of our method with different layers of network in terms of the success and precision plots on the ten videos. We see that the DML tracker with layers $\mathcal{K} = 4$ reports the best result in the precision plot and the second place in the success plot, and it obtains the best AUC score in the success plot when $\mathcal{K}$ is set to 3. This indicates that the performance of DML can be further improved when the number of layers is increased. However, the improvement is not significant and time-consuming. Therefore, we use the number of layers $\mathcal{K} = 3$ in the experiments.

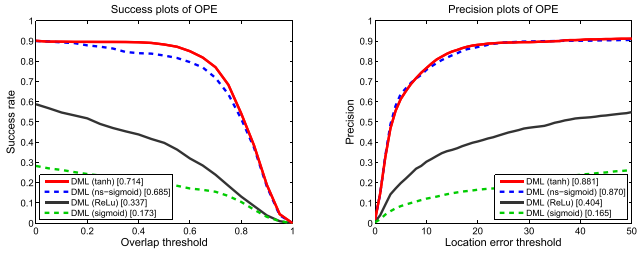


Fig. 10. Success plots and precision plots of the DML tracker with different activation functions on the ten video sequences.

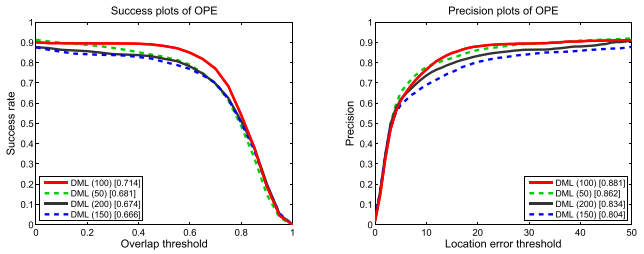


Fig. 11. Success plots and precision plots of the DML tracker with various dimensions (50, 100, 150, 200) of input sample by PCA on ten sequences.

2) Effect of Different Activation Functions: We also analyze the effect of the various nonlinear activation functions on our DML tracker. These activation functions are *tanh*, *sigmoid*, *nonsaturating sigmoid* (ns-sigmoid) [59], and *rectified linear unit* (ReLu), and they are defined as

$$\texttt{tanh:}\ \varphi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{20}$$

$$\texttt{sigmoid:}\ \varphi(x) = \frac{1}{1 + e^{-x}} \tag{21}$$

$$\texttt{ns-sigmoid:}\ x = \varphi^3(x)/3 + \varphi(x) \tag{22}$$

$$\texttt{ReLu:}\ \varphi(x) = \max\ (x, 0). \tag{23}$$

Fig. 10 gives the performance of the DML tracker with different activation functions in both the success and precision plots. We find that the DML tracker with tanh and ns-sigmoid can achieve better tracking performance than sigmoid and ReLu. There are two possible reasons.

1) The activation function sigmoid becomes flatter than tanh and ns-sigmoid far to the left or right.
2) The gradient of the ReLu goes to zero when the input data are negative, and the weights are not updated in this case.

Hence, we employ the *tanh* activation function for evaluating our tracker in the experiments.

3) Effect of Different Dimensions by PCA: To investigate the effect of different feature dimensions of our DML on the tracking performance, we tested the DML tracker with different feature dimensions (50, 100, 150, 200) reduced by PCA on these ten sequences. Fig. 11 shows the performance comparison of our DML tracker with various feature dimensions in terms of the success and precision plots on these ten videos. We see the following.

1) The DML tracker with the dimensionality of 100 shows the best results in both the precision plot and the success plot.

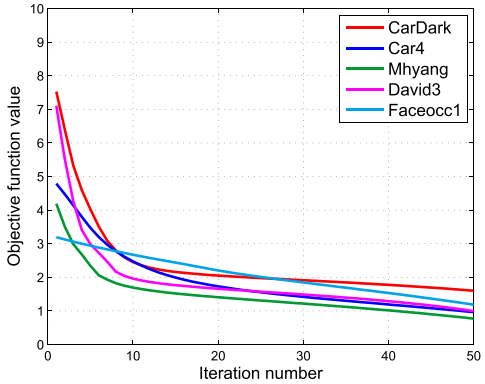


Fig. 12. Convergence curves of DML with the number of iterations on five sequences (i.e., *CarDark*, *Car4*, *Mhyang*, *David3*, and *Faceocc1*) at one update. Note that the numbers on the y-axis are the true loss.

2) The tracking performance of DML gradually reduces when the sample dimensionality is too low (e.g., 50) or too high (e.g., 200). Therefore, we fixed it at 100 in our experiments.

4) Computational Complexity and Convergence Analysis: The forward propagation (FP) and back propagation (BP) are two important procedures to our DML approach. In each iteration, the major calculation of the FP part is

$$4N \sum_{k=1}^{\mathcal{K}} \left(r^{(k-1)} + 1 + \frac{1}{2} N_\varphi \right) r^{(k)} \tag{24}$$

where N ($N = \mathcal{P} + \mathcal{N}$) is the number of sample pairs (including positive and negative pairs); $r^{(k)}$ is the number of neurons at the kth layer; $\mathcal{K}+1$ is the total number of layers; and N_φ is operations for evaluating activation function $\varphi(\cdot)$. The major operations of the BP part in each iteration is

$$Nr^{(\mathcal{K})} + 4N \sum_{k=1}^{\mathcal{K}} \left(\frac{1}{2} N_{\varphi'} + \frac{3}{2} r^{(k-1)} + 1 \right) r^{(k)} \tag{25}$$

where $N_{\varphi'}$ is operations for evaluating derivative $\varphi'(\cdot)$ of the activation function $\varphi(\cdot)$. Adding (24) and (25), we can obtain the computational complexity of the DML method for each iteration as

$$Nr^{(\mathcal{K})} + 2N \sum_{k=1}^{\mathcal{K}} (5r^{(k-1)} + N_\varphi + N_{\varphi'} + 4) r^{(k)}. \tag{26}$$

We evaluate the convergence rate of the DML approach on several sequences at one update. These sequences are *CarDark*, *Car4*, *Mhyang*, *David3*, and *Faceocc1*. Fig. 12 plots the objective function value of DML versus different numbers of iterations on several sequences at one update. We can see that the proposed DML tracker converges in 20–30 iterations.

5) Comparison With Existing Metric Learning Methods: We compared our DML with two exising metric-learning-based methods: 1) information-theoretic metric learning [60] and 2) MLSAM [15]. Fig. 13 shows the performance comparison of our DML tracker and other metric-learning-based methods in terms of the success and precision plots on the whole benchmark data set (51 videos). We see that our DML tracker obtains some better tracking performance than other

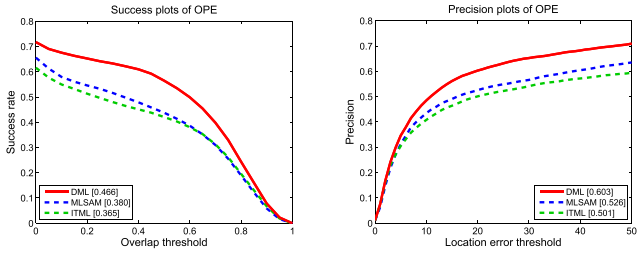


Fig. 13. Success plots and precision plots of DML and other distance metric-learning-based methods on the benchmark data set (51 videos).

two metric-learning-based methods in both the precision plot and the success plot. Hence, our DML tracker is complementary to existing distance metric-learning-based trackers.

VI. Conclusion

In this paper, we have proposed a DML approach for robust visual tracking. The proposed DML tracker can adaptively learn a set of hierarchical nonlinear transformations to map samples into a latent subspace, where the objects are projected much closer and the marginal between objects and backgrounds are maximized, so that objects can be easily separated from the background regions. Experiments on the benchmark data set including 51 challenging videos have shown that our DML tracker achieves a very competitive performance compared with 11 state-of-the-art tracking methods.

In the future, we are going to further improve the performance of our method by using other advanced procedures in the whole tracking approach. Some possible future directions include: utilizing auxiliary data for information transfer, using advanced deep architectures, and employing more advanced template update models and powerful feature representation.

References

[1] M. J. Black and A. D. Jepson, "EigenTracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vis.*, vol. 26, no. 1, pp. 63–84, Jan. 1998.

[2] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1296–1311, Oct. 2003.

[3] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, May 2008.

[4] B. Babenko, M.-H. Yang, and S. J. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 983–990.

[5] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *Proc. 12th IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 1436–1443.

[6] C. Shen, J. Kim, and H. Wang, "Generalized kernel-based visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 119–130, Jan. 2010.

[7] H. Li, C. Shen, and Q. Shi, "Real-time visual tracking using compressive sensing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1305–1312.

[8] X. Li, C. Shen, Q. Shi, A. Dick, and A. van den Hengel, "Non-sparse linear representations for visual tracking with online reservoir metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1760–1767.

[9] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 864–877.

[10] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2371–2378.

[11] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 809–817.

[12] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1822–1829.

[13] F. Yang, H. Lu, and M.-H. Yang, "Robust visual tracking via multiple kernel boosting with affinity constraints," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 242–254, Feb. 2014.

[14] Z. Xiao, H. Lu, and D. Wang, "L2-RLS-based object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1301–1309, Aug. 2014.

[15] Y. Wu, B. Ma, M. Yang, J. Zhang, and Y. Jia, "Metric learning based structural appearance model for robust visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 5, pp. 865–877, May 2014.

[16] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 188–203.

[17] H. Li, Y. Li, and F. Porikli, "DeepTrack: Learning discriminative feature representations by convolutional neural networks for visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–12.

[18] H. Li, Y. Li, and F. Porikli, "Robust online visual tracking with a single convolutional neural network," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 194–209.

[19] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[20] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. Van Den Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, pp. 58:1–58:48, Sep. 2013.

[21] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[22] G. D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun./Jul. 2004, pp. I-790–I-797.

[23] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[24] A. M. Elgammal, R. Duraiswami, and L. S. Davis, "Probabilistic tracking in joint feature-spatial spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2003, pp. I-781–I-788.

[25] P. A. Viola and W. M. Wells, III, "Alignment by maximization of mutual information," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 1995, pp. 16–23.

[26] X. Wang, G. Hua, and T. X. Han, "Discriminative tracking by metric learning," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 200–214.

[27] G. Tsagkatakis and A. Savakis, "Online distance metric learning for object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 12, pp. 1810–1821, Dec. 2011.

[28] N. Jiang, W. Liu, and Y. Wu, "Order determination and sparsity-regularized metric learning adaptive visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1956–1963.

[29] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1269–1276.

[30] N. Wang, J. Wang, and D.-Y. Yeung, "Online robust non-negative dictionary learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 657–664.

[31] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2042–2049.

[32] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. Brit. Mach. Vis. Conf.*, 2006, pp. 47–56.

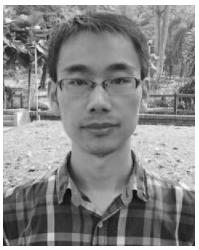
[33] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 263–270.

[34] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.

[35] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time object tracking via online discriminative feature selection," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4664–4677, Dec. 2013.

[36] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 702–715.

[37] L. Zhang and L. J. P. van der Maaten, "Preserving structure in model-free tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 756–769, Apr. 2014.
[38] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, May 2006.
[39] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[40] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[41] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 609–616.
[42] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 3361–3368.
[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.
[44] G. B. Huang, H. Lee, and E. G. Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2518–2525.
[45] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Jan. 2010.
[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[47] X. Cai, C. Wang, B. Xiao, X. Chen, and J. Zhou, "Deep nonlinear metric learning with independent subspace analysis for face verification," in *Proc. ACM Multimedia Conf.*, 2012, pp. 749–752.
[48] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1875–1882.
[49] S. Nowlan and J. Platt, "A convolutional neural network hand tracker," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 901–908.
[50] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1610–1623, Oct. 2010.
[51] J. Jin, A. Dundar, J. Bates, C. Farabet, and E. Culurciello, "Tracking with deep neural networks," in *Proc. 47th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2013, pp. 1–5.
[52] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. New York, NY, USA: Springer-Verlag, 2001.
[53] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[54] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[56] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2411–2418.
[57] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 2356–2368, May 2014.
[58] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1830–1837.
[59] G. Andrew, R. Arora, J. A. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1247–1255.
[60] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 209–216.



Junlin Hu received the B.Eng. degree from the Xi'an University of Technology, Xi'an, China, in 2008, and the M.Eng. degree from Beijing Normal University, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His current research interests include computer vision, pattern recognition, and biometrics.

Jiwen Lu (M'11–SM'15) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in electrical engineering from the Xi'an University of Technology, Xi'an, China, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore.

He is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored or co-authored over 130 scientific papers in these areas, where more than 40 papers are published in the IEEE TRANSACTIONS journals and top-tier computer vision conferences. His current research interests include computer vision, pattern recognition, and machine learning.

Dr. Lu was a recipient of the First-Prize National Scholarship and the National Outstanding Student Award from the Ministry of Education, China, in 2002 and 2003, the Best Student Paper Award from the Pattern Recognition and Machine Intelligence Association of Singapore in 2012, the Top 10% Best Paper Award from the IEEE International Workshop on Multimedia Signal Processing in 2014, and the National 1000 Young Talents Plan Program in 2015. He currently serves as an Associate Editor of *Pattern Recognition Letters*, *Neurocomputing*, and the IEEE BIOMETRICS COUNCIL NEWSLETTERS.

Yap-Peng Tan (M'97–SM'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1993, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 1995 and 1997, respectively, all in electrical engineering.

He was with Intel Corporation, Chandler, AZ, USA, and Sharp Laboratories of America, Camas, WA, USA, from 1997 to 1999. In 1999, he joined Nanyang Technological University, Singapore, where he is currently an Associate Professor and the Associate Chair (Academic) of the School of Electrical and Electronic Engineering. He is the principal inventor or co-inventor on 15 U.S. patents in the areas of image and video processing. His current research interests include image and video processing, content-based multimedia analysis, computer vision, and pattern recognition.

Dr. Tan was a recipient of an IBM Graduate Fellowship from the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, from 1995 to 1997. He served as the Chair of the Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems Society, a member of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society, and a Voting Member of the ICME Steering Committee. He is an Editorial Board Member of the IEEE TRANSACTIONS ON MULTIMEDIA, the *EURASIP Journal on Advances in Signal Processing*, and the *EURASIP Journal on Image and Video Processing*, and an Associate Editor of the *Journal of Signal Processing Systems*. He was the General Co-Chair of the 2010 IEEE International Conference on Multimedia and Expo and is the Co-Chair of the 2015 IEEE Conference on Visual Communications and Image Processing. He was a Guest Editor of special issues of several journals, including the IEEE TRANSACTIONS ON MULTIMEDIA.