Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Online Dictionary Learning on Symmetric Positive Definite Manifolds with Vision Applications

Shengping Zhang[1,3], **Shiva Kasiviswanathan**[2], **Pong C Yuen**[3] and **Mehrtash Harandi**[4]
[1]School of Computer Science and Technology, Harbin Institute of Technology at Weihai, China
[2]General Electric Global Research, United States
[3]Department of Computer Science, Hong Kong Baptist University, Hong Kong
[4]NICTA and Australian National University, Australia
s.zhang@hit.edu.cn, kasivisw@gmail.com, pcyuen@comp.hkbu.edu.hk, mehrtash.harandi@nicta.com.au

Abstract

Symmetric Positive Definite (SPD) matrices in the form of region covariances are considered rich descriptors for images and videos. Recent studies suggest that exploiting the Riemannian geometry of the SPD manifolds could lead to improved performances for vision applications. For tasks involving processing large-scale and dynamic data in computer vision, the underlying model is required to progressively and efficiently adapt itself to the new and unseen observations. Motivated by these requirements, this paper studies the problem of online dictionary learning on the SPD manifolds. We make use of the Stein divergence to recast the problem of online dictionary learning on the manifolds to a problem in Reproducing Kernel Hilbert Spaces, for which, we develop efficient algorithms by taking into account the geometric structure of the SPD manifolds. To our best knowledge, our work is the first study that provides a solution for online dictionary learning on the SPD manifolds. Empirical results on both large-scale image classification task and dynamic video processing tasks validate the superior performance of our approach as compared to several state-of-the-art algorithms.

Introduction

In computer vision, our main focus area here, the need for adaptive online methods arises in countless applications that process large-scale and dynamic data (Ross et al. 2008; Mairal et al. 2010; Lu, Shi, and Jia 2013; Wang, Wang, and Yeung 2013; Zhang et al. 2013b; Xu et al. 2013). Structured descriptors in the form of SPD matrices (e.g., covariance descriptors) are becoming pervasive to describe images and videos (Porikli, Tuzel, and Meer 2006; Zhang et al. 2008; Harandi et al. 2012; Hu et al. 2012; Guo, Ishwar, and Konrad 2013; Harandi, Salzmann, and Porikli 2014). Despite their wide applications and appealing properties, SPD matrices which lie on a special type of Riemannian manifold (henceforth, referred to as SPD manifold) are difficult to analyze (Harandi et al. 2012; Cherian and Sra 2014).

This paper addresses the problem of online dictionary learning on the SPD manifolds. We consider the following two problems: (i) *Coding:* Given a matrix $\boldsymbol{X}$ and a set of matrices $\mathbb{D} = \{\boldsymbol{D}_i\}_{i=1}^k$, where $\boldsymbol{X}$ and $\boldsymbol{D}_i$ are all SPD matrices, how can $\boldsymbol{X}$ be approximated by a combination of matrices in $\mathbb{D}$? (ii) *Dictionary Learning:* Given a set of SPD matrices $\mathbb{X} = \{\boldsymbol{X}_i\}_{i=1}^t$ arriving in a steaming fashion, how can a set $\mathbb{D}$ be learned in an online fashion to "represent" $\mathbb{X}$ accurately? We provide efficient approaches for tackling these problems that take into account the geometric structure of the SPD manifolds.

One of the difficulties in dealing with SPD matrices is that they do not naturally lie in a linear space. To overcome this, we follow the approach of (Harandi et al. 2012) and use a special type of the matrix Bregman divergence, namely the Stein divergence (Sra 2012) to recast our problems on the SPD manifolds as problems in Reproducing Kernel Hilbert Spaces (RKHS). As explained later, this mapping preserves most properties of the original manifold geometry.

For coding, we propose an efficient (sparse) coding algorithm in RKHS based on the method of accelerated proximal gradient (Nesterov 1983; Beck and Teboulle 2009). This leads to a (provably) faster algorithm for coding in RKHS than previous known approaches which were based on techniques such as OMP (Nguyen et al. 2012) and Feature-Sign Search (Gao, Tsang, and Chia 2010).

The problem of dictionary learning is to estimate a collection of basis vectors over which a given data collection can be accurately reconstructed, often with sparse encodings. This is a well-investigated problem over Euclidean spaces (Olshausen and Field 1997; Aharon, Elad, and Bruckstein 2006; Mairal et al. 2010). However, a limitation with traditional dictionary learning algorithms, e.g., (Olshausen and Field 1997; Aharon, Elad, and Bruckstein 2006) is the requirement to have the whole dataset available beforehand to minimize the learning cost. Hence, these methods cannot efficiently deal with very large data sets or data that dynamically vary over time, such as video sequences. In the last few years, online dictionary learning has been extensively studied in the Euclidean spaces (Mairal et al. 2010; Kasiviswanathan et al. 2012). However, to our best knowledge, there is no prior work on online dictionary learning on the SPD manifolds. Our dictionary learning algorithm is unique in the sense that it uses a *geometric gradient descent* approach and at any given timepoint it only consumes the incoming data together with seen observations over a short time window. In contrast to existing studies for dictionary learning in RKHS such as (Harandi et al. 2012; Nguyen et al. 2012), which rely on access to the whole

dataset and other approximations for dictionary learning, our approach is significantly more efficient and does not approximate the resulting optimization problems.

We apply our proposed approach on three commonly encountered vision challenges of *background subtraction* (Sheikh and Shah 2005; Narayana, Hanson, and Learned-Miller 2012; Haines and Xiang 2014), *visual tracking* (Ross et al. 2008; Bao et al. 2012; Wang, Wang, and Yeung 2013), and *digit recognition* (Lecun et al. 1998; Yang et al. 2011; Lu, Shi, and Jia 2013). These applications require the use of online learning method either due to the applications themselves (e.g., updating the model to handle dynamic background for background subtraction or to overcome the appearance variances for visual tracking), or to reduce the processing time as in the case of digit recognition. For these applications, we empirically compare and contrast our proposed approach against several relevant known state-of-the-art techniques.

Related Work

The decomposition of an image or video signal using a few atoms of a trained dictionary has been shown to deliver notable results for various visual tasks (Mairal et al. 2010; Lu, Shi, and Jia 2013; Xu et al. 2013; Zhang et al. 2013a; Lan, Ma, and Yuen 2014). Significant progress has been made in developing the theory of sparse coding and dictionary learning (batch or online) for Euclidean vector spaces (Elad 2010; Mairal et al. 2010; Kasiviswanathan et al. 2012) and for high dimensional feature space mapped by implicit kernel functions (Gao, Tsang, and Chia 2010; Nguyen et al. 2012). Similar problems on the manifold of SPD matrices have also received recent attention (Sra and Cherian 2011; Harandi et al. 2012; Ho, Xie, and Vemuri 2013; Cherian and Sra 2014), but none of them considers online dictionary learning.

For the (coding) problem of representing a SPD matrix using a dictionary consisting of SPD matrices (Cherian and Sra 2014) recently proposed a formulation based on the Affine Invariant Riemannian Metric (AIRM) distance (Pennec, Fillard, and Ayache 2006). This formulation leads to a difficult noncnovex problem, which remains non-convex even if we take into account the geodesic convexity of the AIRM distance (even though experimental results of (Cherian and Sra 2014) suggest it could be better than using Stein divergences for some object recognition applications). The problem of dictionary learning remains unsolved in this Riemannian setting. For the batch dictionary learning on the SPD manifolds, (Sra and Cherian 2011) proposed to measure the similarity between SPD matrices using the Frobenius norm and formulated the sparse coding and dictionary learning problems accordingly. While solving the problems using purely Euclidean structure of SPD matrices is computationally attractive, it neglects the Riemannian structure of the SPD manifolds. Another line of research is to perform sparse coding and dictionary learning by flattening the SPD manifold using its identity (Guo, Ishwar, and Konrad 2013). Though such embedding considerably simplifies the follow up developments, the pair-wise distances are no longer adequate, which can negatively affect the performance. Along similar lines, recently (Ho, Xie, and Vemuri 2013) proposed to exploit the tangent spaces of the SPD manifold for sparse coding and dictionary learning. Since the sparse coding problem has a trivial solution in this approach, an affine constraint has to be added. While having an affine constraint along with sparse coding is beneficial in specific tasks (e.g., clustering), in general, the resulting formulation is restrictive and no longer addresses the original problem. Furthermore, working in successive tangent spaces, though common, values only as a first-order approximation to the SPD manifold at each step, and is also computationally very demanding.

In essence, the approach proposed by (Harandi et al. 2012) is the closest to our work here, where similarly RKHS embedding was used for sparse coding and dictionary learning. Nevertheless, our work here differs significantly from theirs as we are interested in online dictionary learning while they consider a batch framework. Moreover, in (Harandi et al. 2012) dictionary atoms were obtained through an approximation. Since the approximation could generate negative definite matrices, projection to the positive cone is required as a post-processing. We note that positive cone has no boundary, and hence such projection is not well-posed. In contrast, here we propose a principled and geometric approach to update dictionary atoms which guarantees positive definiteness of the solution. Also, the use of accelerated techniques for sparse coding and online dictionary update makes our approach better suited for video applications.

For each of three applications considered in this paper, there are many online algorithms (not particularly, based on dictionary learning). We mention a few relevant techniques in the experimental section.

Preliminaries

We use $[n]$ to denote the set $\{1, \ldots, n\}$. Vectors are always column vectors and are denoted by bold lower letters (e.g., $\boldsymbol{a}$). Notation a_i is used to indicate element at position the i of vector $\boldsymbol{a}$. Matrices are denoted by bold upper case letters (e.g., $\boldsymbol{A}$). The soft-thresholding operator is the following non-linear function for $a, T \in \mathbb{R}$: $\text{soft}(a, T) = \text{sign}(a) \cdot \max\{|a| - T, 0\}$. The operator soft is extended to a vector by applying it to every entry in the vector.

SPD Matrix and Stein Kernel

A $d \times d$, real SPD matrix $\boldsymbol{A}$ has the property that $\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} > 0$ for any non-zero $\boldsymbol{v} \in \mathbb{R}^d$. The space of $d \times d$ SPD matrices, denoted by $\mathcal{S}^d_{++}$, is not a vector space since, instead, $\mathcal{S}^d_{++}$ forms the interior of a convex cone in the d^2-dimensional Euclidean space. The $\mathcal{S}^d_{++}$ space is mostly studied when endowed with a Riemannian metric, thus forming a Riemannian manifold (Sra 2012). In this paper, we use Stein divergence which has variety of properties akin to AIRM[1] and importantly can be embedded into RKHS in the form of Gaussian kernel (Harandi et al. 2012). Embedding into Hilbert space allows us to design algorithms for linear spaces with manifold-valued data.

[1] A very recent paper (Harandi, Salzmann, and Hartley 2014) showed that the lengths of curves under Stein divergence and AIRM distance are tightly related.

Definition 1. *The Stein or S divergence is a symmetric member of Bregman matrix divergences and is defined as:*

$$S(\boldsymbol{A}, \boldsymbol{B}) \triangleq \operatorname{logdet}\left(\frac{\boldsymbol{A}+\boldsymbol{B}}{2}\right) - \frac{1}{2}\operatorname{logdet}(\boldsymbol{A}\boldsymbol{B}) \ .$$

For the Stein divergence, the following kernel (referred henceforth as the Stein kernel)

$$\kappa(\boldsymbol{A}, \boldsymbol{B}) = \exp\{-\beta \cdot S(\boldsymbol{A}, \boldsymbol{B})\} \ ,$$

is positive definite for certain choices of $\beta > 0$ (Sra 2012). A positive definite kernel enables us to transpose problems on $\mathcal{S}_{++}^d$ to familiar problems in RKHS.

Let $\phi : \mathcal{S}_{++}^d \to \mathcal{H}$ be the implicit mapping associated with the Stein kernel such that for any two positive definite matrices $\boldsymbol{A}, \boldsymbol{B}$, we have $\kappa(\boldsymbol{A}, \boldsymbol{B}) = \phi(\boldsymbol{A})^\top \phi(\boldsymbol{B})$. Note that since $\kappa(\boldsymbol{A}, \boldsymbol{A}) = \phi(\boldsymbol{A})^\top \phi(\boldsymbol{A}) = 1$.

Online Dictionary Learning on SPD Manifolds

In this section, we propose an online framework for dictionary learning to handle streaming SPD data matrices. Many applications on videos, such as background subtraction and visual tracking, can be formulated as online learning process where a model is first built, then a new observation received is compared with the model to make a decision and finally the model is updated using the current observation. Recognition tasks on large-scale images such as digit recognition, do not require model updates over time, but learning the model on the whole training data has high computation complexity. Therefore, online learning that processes a small set of data at each batch is extremely useful.

Problem Formulation. Let $\{\boldsymbol{X}_t \in \mathcal{S}_{++}^d, t = 1, 2, \dots\}$ denote a sequence of streaming input matrices where $\boldsymbol{X}_t$ is the SPD matrix introduced at time t. Using the input matrix stream $\{\boldsymbol{X}_t\}_{t=1,\dots,}$ we learn a dictionary of k atoms which are also SPD matrices. Let $\mathbb{D}_{t-1} = \{\boldsymbol{D}_{t-1_1}, \dots, \boldsymbol{D}_{t-1_k}\}$ represent the atoms of the dictionary at time $t-1$.

Since we operate in an online framework, we introduce a short sliding time window ω over which we maintain the history. At every timestep t, our goals are:

(a) check whether $\phi(\boldsymbol{X}_t)$ can be "well" approximated as a sparse linear combination of elements from the set $\{\phi(\boldsymbol{D}_{t-1_1}), \dots, \phi(\boldsymbol{D}_{t-1_k})\}$ (the *sparse coding* step).

(b) obtain $\boldsymbol{D}_{t_1} \dots, \boldsymbol{D}_{t_k}$ such that the elements of $\{\phi(\boldsymbol{D}_{t_1}), \dots, \phi(\boldsymbol{D}_{t_k})\}$ form a good basis to linearly construct $\phi(\boldsymbol{X}_{t-\omega+1}), \dots, \phi(\boldsymbol{X}_t)$ (the *dictionary learning step*).

In this paper, for sparse coding we use an ℓ_2-loss function along with an ℓ_1-regularization term:

$$\underset{\boldsymbol{h}\in\mathbb{R}^k}{\arg\min} \left\| \phi(\boldsymbol{X}_t) - \sum_{i=1}^{k} \phi(\boldsymbol{D}_{t-1_i}) h_i \right\|^2 + \lambda \|\boldsymbol{h}\|_1 \ , \qquad (1)$$

where λ is a regularization parameter. The ℓ_2-reconstruction error measures the quality of the approximation while the complexity is measured by the ℓ_1-norm of the optimal $\boldsymbol{h}$.

In the following, for simplicity we will assume $t \geq \omega$. We define the problem of dictionary learning at timestep t as:

$$\underset{\substack{\boldsymbol{D}_1,..,\boldsymbol{D}_k \in \mathcal{S}_{++}^d \\ \boldsymbol{h}_{t-\omega+1},..,\boldsymbol{h}_t \in \mathbb{R}^k}}{\arg\min} \sum_{\tau=t-\omega+1}^{t} \left\| \phi(\boldsymbol{X}_\tau) - \sum_{i=1}^{k} \phi(\boldsymbol{D}_i) h_{\tau_i} \right\|^2 + \lambda \|\boldsymbol{h}_\tau\|_1 \ . \qquad (2)$$

Here, h_{τ_i} is the ith entry in the vector $\boldsymbol{h}_\tau$. The solution of (2) ($\boldsymbol{D}_1, \dots, \boldsymbol{D}_k$ variables) forms the set of new dictionary atoms $\boldsymbol{D}_{t_1}, \dots, \boldsymbol{D}_{t_k}$.

Solving the Sparse Coding Problem. To efficiently solve the problem in (1), we adopt the Accelerated Proximal Gradient (APG) algorithm (Nesterov 1983; Beck and Teboulle 2009) (see the appendices for a background on the APG algorithm). To apply the APG algorithm, we need the gradient of $\|\phi(\boldsymbol{X}_t) - \sum_{i=1}^k \phi(\boldsymbol{D}_{t-1_i}) h_i\|^2$ (with respect to $\boldsymbol{h}$) at various iterates. This gradient equals, $-2\sum_{i=1}^k \kappa(\boldsymbol{D}_{t-1_i}, \boldsymbol{X}_t) + 2\boldsymbol{K}\boldsymbol{h}$, where $\boldsymbol{K}$ denotes a $k \times k$ matrix with (i, j)th entry being $\kappa(\boldsymbol{D}_{t-1_i}, \boldsymbol{D}_{t-1_j})$. Let L denote the Lipschitz constant of the above gradient function (which can be set using the largest eigenvalue of the Hessian of the loss function). Algorithm 1 details the resulting procedure for solving (1). Subscript (p) denotes the value of a variable at the pth iteration of the procedure.

Algorithm 1: : Kernel sparse coding (1) using APG

Input: New observance $\boldsymbol{X}_t \in \mathcal{S}_{++}^d$, previous dictionary atoms $\boldsymbol{D}_{t-1_1}, \dots, \boldsymbol{D}_{t-1_k}$

Initialize $\boldsymbol{b}_{(1)} = \boldsymbol{h}_{(0)} \leftarrow \boldsymbol{0}$, $q_{(1)} \leftarrow 1$, and $p \leftarrow 1$

while *not(converge)* **do**

$\quad \boldsymbol{h}_{(p)} \leftarrow \text{soft}\left(\boldsymbol{b}_{(p)} + \frac{2\sum_{i=1}^k \kappa(\boldsymbol{D}_{t-1_i}, \boldsymbol{X}_t) - 2\boldsymbol{K}\boldsymbol{b}_{(p)}}{L}, \frac{\lambda}{2}\right)$

$\quad q_{(p+1)} \leftarrow \frac{1+\sqrt{1+4q_{(p)}^2}}{2}$

$\quad \boldsymbol{b}_{(p+1)} \leftarrow \boldsymbol{h}_{(p)} + \frac{q_{(p)}-1}{q_{(p+1)}}(\boldsymbol{h}_{(p)} - \boldsymbol{h}_{(p-1)})$

$\quad p \leftarrow p + 1$

end

Return $\boldsymbol{h}_{(p)}$

By the guarantees of the APG algorithm, within $T = O(\sqrt{L/\epsilon})$ iterations, $\boldsymbol{h}_{(T)}$ is such that $\|\boldsymbol{h}_{(T)} - \boldsymbol{h}^\star\| \leq \epsilon$, where $\boldsymbol{h}^\star$ is the minimizer of (1).

Updating the Dictionary. Our algorithm for solving (2) is based on block-coordinate descent. We optimize cyclically over individual $\boldsymbol{D}_i$ and h_{τ_i} ($i \in [k], \tau \in \{t-\omega+1, \dots, t\}$) variables while keeping all other variables fixed. This iterative optimization is reminiscent of the K-SVD approach for dictionary learning in Euclidean spaces (Elad 2010).

Updating $\boldsymbol{h}_{t-\omega+1}, \dots, \boldsymbol{h}_t$: Let $\boldsymbol{h}_\tau = (h_{\tau_1}, \dots, h_{\tau_k})$ for $\tau \in \{t-\omega+1, \dots, t\}$. We derive the updates for $\boldsymbol{h}_{\tau_i}$. Holding all variables except h_{τ_i} fixed, the dictionary learning problem (2) can be reduced to:

$$\min_{h_{\tau_i}\in\mathbb{R}} \|\boldsymbol{Z}_\tau - \phi(\boldsymbol{D}_i) h_{\tau_i}\|^2 + \lambda |h_{\tau_i}| \ , \qquad (3)$$

where $\boldsymbol{Z}_\tau = \phi(\boldsymbol{X}_\tau) - \sum_{j\neq i} \phi(\boldsymbol{D}_j) h_{\tau_j}$. $\boldsymbol{Z}_\tau$ is never evaluated explicitly, instead the algorithm only needs

$$\boldsymbol{Z}_\tau^\top \phi(\boldsymbol{D}_i) = \kappa(\boldsymbol{X}_\tau, \boldsymbol{D}_i) - \sum_{j\neq i} h_{\tau_j} \kappa(\boldsymbol{D}_j, \boldsymbol{D}_i) \ .$$

To finish note that $\|\boldsymbol{Z}_\tau - \phi(\boldsymbol{D}_i)h_{\tau_i}\|^2 = \|h_{\tau_i} - \boldsymbol{Z}_\tau^\top \phi(\boldsymbol{D}_i)\|^2$ (we used the fact that $\phi(\boldsymbol{D}_i)^\top \phi(\boldsymbol{D}_i) = 1$). This implies that (3) is equivalent to $\arg\min_{h_{\tau_i}\in\mathbb{R}} \|h_{\tau_i} - \boldsymbol{Z}_\tau^\top \phi(\boldsymbol{D}_i)\|^2 + \lambda|h_{\tau_i}|$. It is well-known that the minimizer of this optimization problem is given by the soft-thresholding operator, soft$(\boldsymbol{Z}_\tau^\top \phi(\boldsymbol{D}_i), \lambda/2)$ (Bach et al. 2012).

Updating $\boldsymbol{D}_i$: Holding all variables except $\boldsymbol{D}_i$ fixed, the dictionary learning problem (2) can be reduced to:

$$\arg\min_{\boldsymbol{D}_i\in\mathcal{S}^d_{++}} \sum_{\tau=t-\omega+1}^{t} \|\boldsymbol{Z}_\tau - \phi(\boldsymbol{D}_i)h_{\tau_i}\|^2 \ .$$

A simple manipulation (utilizing $\phi(\boldsymbol{D}_i)^\top \phi(\boldsymbol{D}_i) = 1$) shows that the above optimization is equivalent to

$$\arg\min_{\boldsymbol{D}_i\in\mathcal{S}^d_{++}} \sum_{\tau=t-\omega+1}^{t} \|\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}\|^2.$$

Let $F(\boldsymbol{D}_i) = \sum_{\tau=t-\omega+1}^{t} \|\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}\|^2$. We minimize $F(\boldsymbol{D}_i)$ by adopting the geometric gradient descent technique (Bonnabel 2013; Pennec, Fillard, and Ayache 2006). To apply this technique, the gradient of $F(\boldsymbol{D}_i)$ with respect to $\boldsymbol{D}_i$ (denoted by $\nabla_{\boldsymbol{D}_i} F$) is required. The Lemma 2 in the appendices derives $\nabla_{\boldsymbol{D}_i} F$.

The geometric gradient descent minimizes $F(\boldsymbol{D}_i)$ by iteratively updating,

$$\boldsymbol{D}_{i_{(p+1)}} = \boldsymbol{D}_{i_{(p)}}^{1/2} \exp\Big(-\alpha \boldsymbol{D}_{i_{(p)}}^{-1/2} \left(\nabla_{\boldsymbol{D}_{i_{(p)}}} F\right) \boldsymbol{D}_{i_{(p)}}^{-1/2} \Big) \boldsymbol{D}_{i_{(p)}}^{1/2} \ ,$$

where $\boldsymbol{D}_{i_{(p)}}$ is the value of $\boldsymbol{D}_i$ in the pth iteration and α is the step size. Here, $\exp(\cdot)$ represents the matrix exponential. The gradient descent step ensures that all iterates remain in the SPD manifold, and in practice is known to converge quickly (Pennec, Fillard, and Ayache 2006), as was also evidenced in our experiments.

Putting it all Together. Our entire approach is outlined in Algorithm 2. In our applications, the reconstruction error of $\boldsymbol{h}^*$ can be used to make a decision (to classify a pixel as background or foreground, or to identify the best target candidate for tracking, etc.). Because of the non-convexity of the dictionary learning problem, unfortunately we can not guarantee any global convergence of our approach.[2] With regards to the running time of Algorithms 1 and 2, computing the kernel function ($\kappa(\boldsymbol{A}, \boldsymbol{B})$) is the most expensive calculation, taking $O(d^3)$ time when using standard matrix computation algorithms.

Algorithm 2: : Online Framework for Sparse Coding and Dictionary Learning on SPD Manifold

Input: SPD matrices $\boldsymbol{X}_t, \boldsymbol{X}_{t-1}, \ldots, \boldsymbol{X}_{t-\omega+1}$ and atoms of previous dictionary $\mathbb{D}_t = \{\boldsymbol{D}_{t-1_1}, \ldots, \boldsymbol{D}_{t-1_k}\}$

Decision (Sparse Coding) Step: (solve Equation (1))
Use Algorithm 1 to obtain:
$\boldsymbol{h}^* \leftarrow \arg\min_{\boldsymbol{h}} \|\phi(\boldsymbol{X}_t) - \sum_{i=1}^k \phi(\boldsymbol{D}_{t-1_i})h_i\|^2 + \lambda\|\boldsymbol{h}\|_1$
Take a decision based on $\|\phi(\boldsymbol{X}_t) - \sum_{i=1}^k \phi(\boldsymbol{D}_{t-1_i})h_i^*\|$

Online Dictionary Learning Step: (solve Equation (2))
Initialize $\boldsymbol{D}_{t_i} \leftarrow \boldsymbol{D}_{t-1_i}\ \forall i \in [k]$ (warm restart)
while *not(converge)* **do**
  for $i = 1$ **to** k **do**
    Updating h_{τ_i} ***(ith element of*** $\boldsymbol{h}_\tau$***):***
    for $\tau = t - \omega + 1$ **to** t **do**
      $h_{\tau_i} \leftarrow$ soft $\Big(\kappa(\boldsymbol{X}_t, \boldsymbol{D}_{t_i}) - \sum_{j=1,j\neq i}^k h_{\tau_j}\kappa(\boldsymbol{D}_{t_j}, \boldsymbol{D}_{t_i}), \frac{\lambda}{2}\Big)$
    end
    Updating $\boldsymbol{D}_{t_i}$ ***(ith atom of dictionary):***
    $\boldsymbol{D}_{i_{(0)}} \leftarrow \boldsymbol{D}_{t_i}$
    $p \leftarrow 0$
    while *not(converge)* **do**
      $\boldsymbol{D}_{i_{(p+1)}} \leftarrow \boldsymbol{D}_{i_{(p)}}^{1/2} \exp\Big(-\alpha \boldsymbol{D}_{i_{(p)}}^{-1/2} (\nabla_{\boldsymbol{D}_{i_{(p)}}} F)\, \boldsymbol{D}_{i_{(p)}}^{-1/2}\Big) \boldsymbol{D}_{i_{(p)}}^{1/2}$
      $p \leftarrow p + 1$
    end
    $\boldsymbol{D}_{t_i} = \boldsymbol{D}_{i_{(p)}}$ (value at convergence)
  end
end
Return $\mathbb{D}_t = \{\boldsymbol{D}_{t_1}, \ldots, \boldsymbol{D}_{t_k}\}$

Experiments

In this section, we present results of applying the proposed online dictionary learning framework to three vision applications. We compute a covariance matrix from each image block. Let $\boldsymbol{F} = \big[\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_m\big]$ be a $d \times m$ matrix, obtained by stacking m independent feature vectors $\boldsymbol{f}_i \in \mathbb{R}^d$ from an $r \times r$ image block (e.g., each observation could correspond to one pixel in the image block). In our experiments, a feature vector $\boldsymbol{f}_i$ at location (x, y) is constructed as $(x, y, R, G, B, |\partial I/\partial x|, |\partial I/\partial y|)^\top$, where R, G, B are red, green, blue color components at (x, y), and the last two entries capture the first order gray gradients. These feature vectors give rise to a 7×7 covariance matrix for each image block. In the following, let $l(\boldsymbol{X}_t, \mathbb{D}_{t-1})$ denote the sparse coding loss function defined by (1).

The values of algorithmic parameters used in our experiments are: $k = 5$, $\lambda = 0.01$, $T = 5$, $\beta = 0.1$, and $\alpha = 10^{-7}$. These parameter were tuned and then fixed for all experiments. All experiments were done on a PC with 2.83GHz CPU and 6GB memory. Codes of all compared methods were downloaded from respective authors' websites.

Background Subtraction

Given an video (image sequence) the aim of background subtraction (or foreground detection) is to identify foreground objects in each frame of the sequence. In doing so, a background subtraction system usually contains three components: (i) *background modeling*—builds an initial background model for every pixel using training data, (ii) *background subtraction*—a pixel in the query frame is compared against its background model and classified as background or foreground accordingly, and (iii) *background update*—the detected background pixels in a query frame are used

[2] The theory for online dictionary learning developed for Euclidean spaces, see e..g., (Mairal et al. 2010; Kasiviswanathan et al. 2012), do not seem to extend to the case when the dictionaries are learnt in RKHS using kernel functions.

Method	*WavingTrees*	*Fountain*	*WaterSurface*	*Curtain*	*Campus*	*Lobby*	*TimeOfDay*	*LightSwitch*	*Hall*	*Escalator*
JKDE	83.96	50.06	77.64	85.95	80.42	78.06	73.41	10.40	73.49	53.75
AKDE	86.17	68.14	**87.56**	**94.88**	89.49	**83.17**	71.35	16.74	77.91	43.40
DPMM	94.42	67.55	84.87	84.68	86.23	68.22	79.50	17.90	71.85	29.61
GOSUS	82.86	86.19	83.76	93.53	85.53	80.94	65.83	13.68	79.29	66.23
Our Approach (off-line)	83.00	76.62	79.22	88.29	81.40	58.06	80.36	33.78	64.12	65.49
Our Approach	**95.22**	**90.46**	85.32	93.09	**93.01**	81.82	**88.02**	**63.68**	**87.08**	**82.63**

Table 1: Background Subtraction: F-measure (%) on ten test sequences. The second-to-last row lists the results of our approach when the dictionary is not updated after the training step.

to further update their background models. In our setup, let $\boldsymbol{X}_t$ be the covariance descriptor extracted from an image block centered around a pixel in frame t. Let $\mathbb{D}_{t-1}$ be a background dictionary such that the loss function $l(\boldsymbol{X}_t, \mathbb{D}_{t-1})$ is "small" for background pixels and large otherwise. Then by thresholding the loss function, one can classify this pixel as background or foreground. If the pixel is marked as background, then we use $\boldsymbol{X}_t$ to update its background dictionary $\mathbb{D}_{t-1}$ to obtain $\mathbb{D}_t$ for detection in the next frame (at the start dictionaries are initialized using information from training frames). The benefit of dictionary update is that it makes the trained background models to adapt to new observations, which is extremely useful to overcome challenges such as illumination variation or dynamic backgrounds.

We compared our approach to several state-of-the-art methods, including Joint domain-range Kernel Density Estimate (JKDE) (Sheikh and Shah 2005), Adaptive Kernel Density Estimate with hybrid feature (AKDE) (Narayana, Hanson, and Learned-Miller 2012), Grassmannian Online Subspace Updates with Structured-sparsity (GOSUS) (Xu et al. 2013), and Dirichlet Process Mixture Models (DPMM) (Haines and Xiang 2014). We used 10 sequences from two popular benchmark datasets: Wallflower[3] and I2R[4] for comparison, which contain a variety of challenges such as dynamic backgrounds (*WavingTrees*, *Fountain*, *WaterSurface*, *Curtain*, *Campus*), illumination changes (*Lobby*, *TimeOfDay*, *LightSwitch*, and presence of multiple foreground objects (*Hall* and *Escalator*). For each video sequence, we used the first 200 frames not containing the foreground objects for training and the remaining frames were used for detection (testing). For each pixel, the threshold used for detection was set to the maximum reconstruction error on the training frames plus a bias which is chosen from the set $\{0, 0.01, 0.02, 0.03, 0.04, 0.05\}$. The block size $r = 8$, and as mentioned earlier, we set $\omega = 1$.

The F-measure is used for the quantitative evaluation, and is defined as the harmonic mean of the recall and precision. Table 1 demonstrates that the performance of our approach is significantly superior to other studied methods. Our approach achieves the best performance on seven sequences and the second best performanceon two sequences. To validate the benefit of the (online) dictionary update step, we also tried a dictionary-based approach without the dictionary updates. As seen from Table 1, when the dictionary is not updated, the performance decreases significantly.

We also present qualitative evaluation of compared algorithms. Figure 1 shows background subtraction results on some example images from each of the tested sequences. We can visually see that our approach performs better than other compared approaches on background subtraction. It should be emphasized that our approach detects some background pixels along the contour of the foreground object as foreground. This is because the covariance matrix is computed from an image block and not on a single pixel. When detecting a pixel, its neighboring pixels can affect its detection result. However, considering that accurately detecting the foreground contour is not important for most video applications, this drawback is negligible. Also in the second-to-last row, we see visually the benefits of updating the dictionary.

Visual Tracking

Visual tracking is the task of continuously inferring the *state* (i.e., center position and size) of a tracked object in an image sequence. Let $\boldsymbol{x}_{t-1}$ be the state of the object in frame $t-1$. We use the particle filter based tracking framework (Ross et al. 2008; Hu et al. 2012; Bao et al. 2012; Wang, Wang, and Yeung 2013) that employs a set of particles $\boldsymbol{x}_t^1, \ldots, \boldsymbol{x}_t^N$ with individual weights $\omega_t^1, \ldots, \omega_t^N$ to approximate the true posterior state distribution of $\boldsymbol{x}_t$ in frame t. As is common, we sample particles (which are referred to as the *target candidates*) from a Gaussian distribution with $\boldsymbol{x}_{t-1}$ as its mean and a pre-defined variance. Let $\boldsymbol{X}_t^i$ be the covariance matrix extracted from ith target candidate at time t. Assume that a dictionary $\mathbb{D}_{t-1}$ is maintained such that if the candidate $\boldsymbol{X}_t^i$ is very similar to the tracked target, then the loss function $l(\boldsymbol{X}_t^i, \mathbb{D}_{t-1})$ is "small". We then compute the target candidate weight as $\omega_t^i = \exp(-l(\boldsymbol{X}_t^i, \mathbb{D}_{t-1}))$. The candidate with the largest weight is chosen as the tracking result and its covariance matrix is further used to update the dictionary $\mathbb{D}_{t-1}$ to obtain $\mathbb{D}_t$. As the appearance of the tracked target changes over time, dictionary update it is critical to capture target appearances at different poses or illumination conditions.

We compared our approach with four state-of-the-art methods, Incremental Visual Tracking (IVT) (Ross et al. 2008), Sparse Representation Tracking (SRT) (Bao et al. 2012), Online Non-negative Dictionary Learning (ONNDL) (Wang, Wang, and Yeung 2013), and Log-Euclidean Riemannian Subspace (LogERS) (Hu et al. 2012). A total of 12 widely used test sequences from (Ross et al. 2008; Bao et al. 2012) were chosen for evaluating tracking perfor-

[3]http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm

[4]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

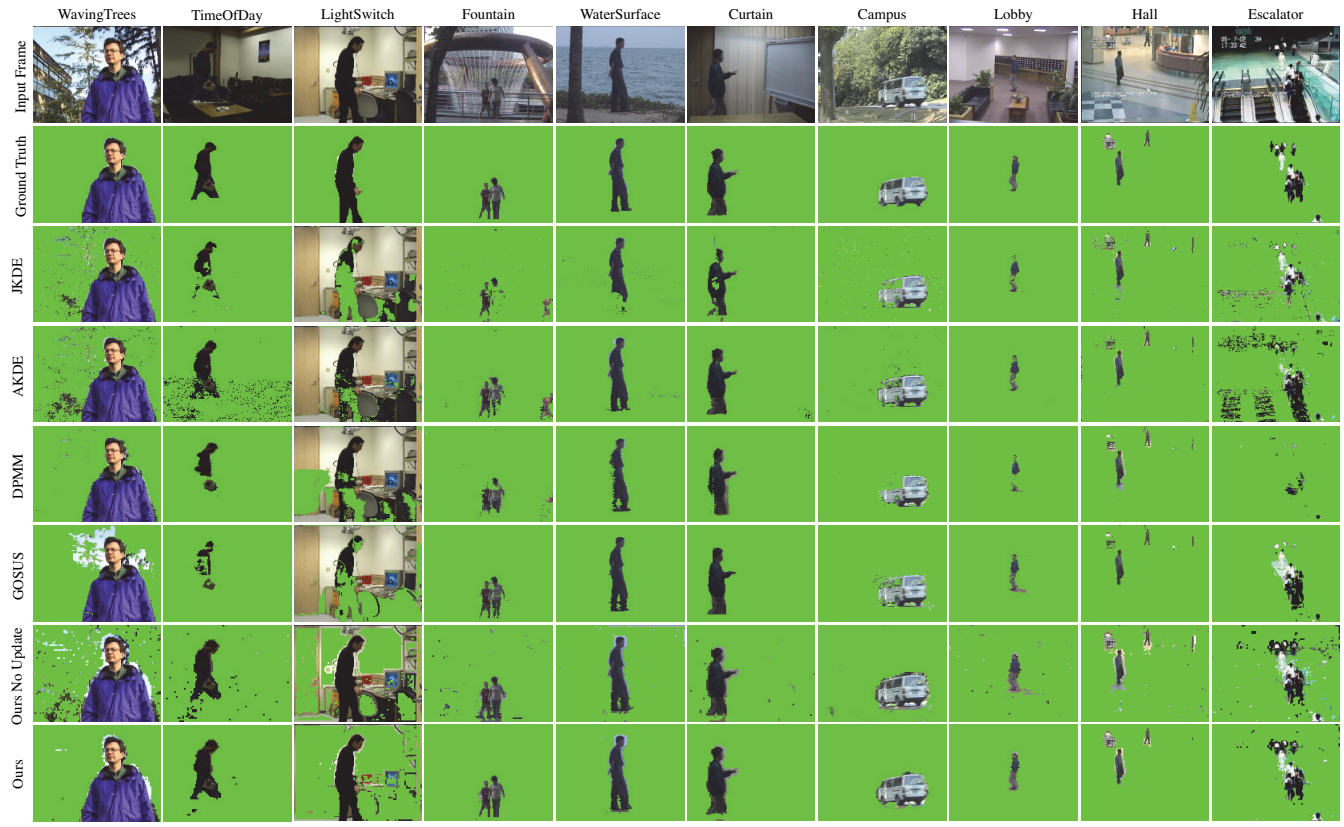

Figure 1: Qualitative comparison of compared algorithms on 10 sequences. The second-to-last row shows the results of our approach without the online dictionary update.

Method	*face*	*face2*	*car*	*david1*	*david2*	*shopping*	*PETS*	*walk*	*singer*	*woman*	*sylv*	*transformer*
LogERS	22.75	6.43	5.28	13.41	8.07	6.84	3.89	28.58	18.18	114.67	7.36	28.56
ONNDL	6.26	5.91	2.16	8.26	**3.94**	**2.32**	2.61	8.13	14.16	5.38	4.94	55.52
SRT	34.09	12.80	1.90	58.90	81.91	49.26	4.93	107.36	73.58	129.75	29.29	163.27
IVT	11.77	7.72	**1.87**	90.83	6.50	49.10	3.28	134.19	5.06	131.19	51.80	150.46
Our Approach	**3.99**	**3.06**	1.92	**4.50**	4.79	2.68	**2.40**	**4.94**	**3.29**	**3.53**	**2.49**	**5.54**

Table 2: Visual Tracking: Average center-position errors (in pixels) on 12 test sequences.

Method	*face*	*face2*	*car*	*david1*	*david2*	*shopping*	*PETS*	*walk*	*singer*	*woman*	*sylv*	*transformer*
LogERS	71.26	91.19	99.52	62.20	64.25	69.96	47.06	27.48	31.41	11.25	89.57	46.67
ONNDL	98.88	99.66	100.00	89.60	92.61	99.80	**99.76**	96.72	24.79	80.18	96.38	49.19
SRT	58.99	54.04	100.00	10.80	23.48	36.80	71.84	12.54	24.79	16.22	43.00	22.58
IVT	79.78	96.97	100.00	30.80	86.30	39.00	99.76	24.48	32.19	14.71	53.37	27.42
Our Approach	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.75	**99.85**	**100.00**	**99.09**	**100.00**	**100.00**

Table 3: Visual Tracking: Success rates (in percentage) on 12 test sequences.

mance, which contain various tracking challenges such as partial occlusion, illumination changes, and pose changes. For each sequence, we used the first 20 frames with manually labeled states of the tracked target for training and the remaining frames for tracking. Similar to (Hu et al. 2012), we resized each target candidate into a 32×32 image, and then divided it into a 4×4 grid (therefore, the block size $r = 8$). For each grid, a dictionary was learned using the training data. When a new test frame arrives, for each target candidate, the extracted covariance descriptor at each grid position is reconstructed using the corresponding dictionary. The sum of reconstruction errors over all sixteen grids is used to compute the weight of the target candidate, and the candidate with the largest weight is chosen as the current tracking result. The covariance descriptors extracted from the tracking result are used to update their corresponding dictionaries. Here again, $\omega = 1$.

For each video frame, the bounding box B obtained by a

tracker (tracking algorithm) was compared with the ground-truth bounding box B_t. The tracker is considered successful if the overlap percentage $area(B \cap B_t)/area(B \cup B_t) > 50\%$. The center-position error is defined as the Euclidean distance (in pixels) between the centers of B and B_t. The performance of a tracker on a sequence is evaluated using success rate (the proportion of frames where the tracker is successful) and average central-position error over all frames used for testing.

As shown in Tables 2 and 3, our approach achieves the lowest average center-position errors on nine sequences and the highest success rates on eleven sequences. We also show some examples of tracking results in Figures 2 and 3. The tracked targets in the *face*, *face2*, *walk*, and *woman* sequences are partially occluded by some objects. The *car*, *david1*, and *singer* sequences have illumination changes. On the *david2*, *sylv*, and *transformer* sequences, the challenge is the pose changes. The similar background-foreground objects appearing in the *shopping* and *PETS* sequences also cause many trackers to fail. As seen from these examples, our approach achieves better performance than other trackers when facing all of these challenges.

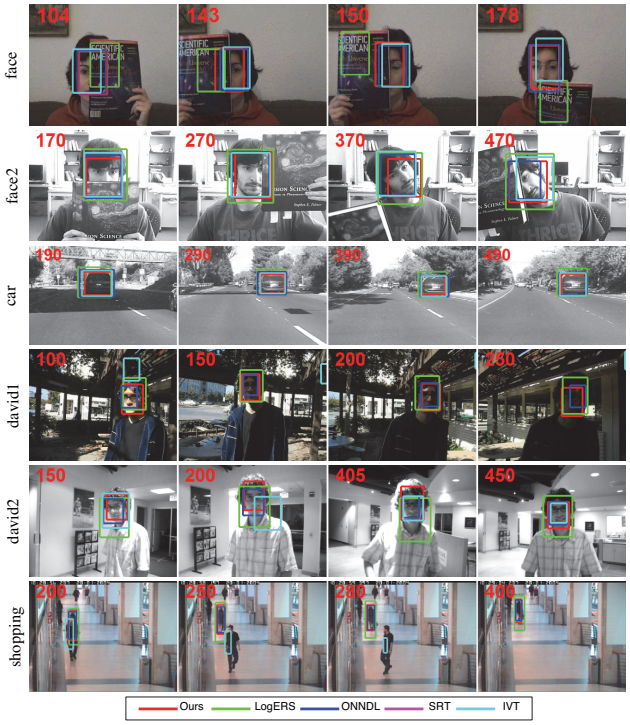

Figure 2: Examples of tracking results on six test sequences: *face*, *face2*, *car*, *david1*, *david2*, and *shopping*.

In term of running speeds, the LogERS, ONNDL, SRT, IVT, and our approach achieve an average of 4.10, 0.121, 0.9869, 6.78 and 0.93 frames/second, respectively. Our approach is only slower compared to the poorly performing trackers. Although our approach is still not quite real-time (as the code is in Matlab), we believe we can achieve near real-time performance with further code optimization and utilizing better hardware.

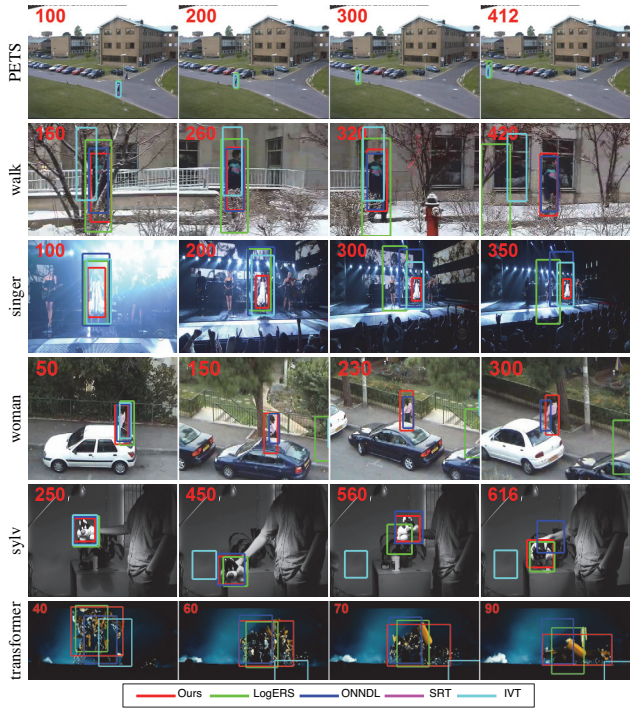

Figure 3: Examples of tracking results on six test sequences: *PETS*, *walk*, *singer*, *woman*, *sylv*, and *transformer*.

Digit Recognition

Digit recognition is an important visual pattern recognition problem that is used in many systems involving document processing. But this task typically requires a large sized training data. Off-line training on the whole training data has both high computation and memory requirements. Therefore, an online learning approach, that processes a small set of training data at each iteration, is more practical. We chose two handwritten digit datasets MNIST (Lecun et al. 1998) and USPS[5] for testing. The MNIST dataset has 60000 training and 10000 testing images, whereas the USPS dataset has 7291 training and 2007 testing images. As in (Lu, Shi, and Jia 2013), we added random outliers to both the training and testing images and then colorize them to greatly increase the versatility and complexity (given that the digits were all in grayscale originally). We resize each image to 15×15 pixels. We learn separate dictionaries for each digit (so ten in total) using covariance matrices extracted from the training data. In the testing phase, given a digit image, we first compute covariance matrix from it, and then reconstruct the covariance matrix over the ten learned dictionaries and choose the one with smallest reconstruction error as recognition result.

We compared our online dictionary learning method with settings of $\omega = 500$ and $\omega =$ size of training data, to other dictionary learning methods (using pixel intensities as features) including online robust dictionary learning

[5]http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html

(ORDL) (Lu, Shi, and Jia 2013), KSVD (Aharon, Elad, and Bruckstein 2006), and Online Euclidean Dictionary Learning (OEDL) (Mairal et al. 2010). Table 4 lists the recognition errors, from which we can see among dictionary learning techniques our approach has significantly lower recognition errors. Also, not surprisingly, larger values of ω helps. But this improvement comes at the cost of increased processing time, as on the MNIST dataset to learn a dictionary for a digit our approach with $\omega = 500$ and $\omega = 60000$ (size of training data) takes 240 and 3000 seconds, respectively.

Dataset	Our Approach (ω = training data size)	Our Approach ($\omega = 500$)	ORDL	KSVD	OEDL
MNIST	15.4	18.9	22.7	39.2	34.3
USPS	23.6	25.1	29.4	45.3	42.5

Table 4: Error rate (%) for the digit recognition task using different methods on the MNIST and USPS datasets.

Conclusion

We proposed an integrated framework for sparse coding and dictionary learning on the SPD manifolds, which benefits from both the appealing properties of SPD matrices as feature descriptors and the notion of sparsity for inference. Our proposed method takes into account the geometric structure of these manifolds and updates the dictionary in an online fashion. This is in contrast to existing studies which merely consider batch dictionary learning. The effectiveness of the proposed approach was extensively evaluated on three important vision applications that operate on large-scale dynamic data.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (No. 61300111), the China Postdoctoral Science Foundation (No. 2014M550192), Hong Kong Scholars Program (XJ2013030) and RGC grant 212313.

Appendices

Accelerated Proximal Gradient Algorithm

Consider the following unconstrained minimization problem

$$\min_{\boldsymbol{a}} \mathcal{F}(\boldsymbol{a}) + \mathcal{G}(\boldsymbol{a}), \tag{4}$$

where $\mathcal{F} : \mathbb{R}^N \to \mathbb{R}$ is a differentiable convex function and its gradient is Lipschitz continuous with parameter L[6], and $\mathcal{G} : \mathbb{R}^N \to \mathbb{R}$ is a non-smooth but convex function.

Below, we describe a version of the Accelerated Proximal Gradient (APG) algorithm for solving (4) as presented by (Beck and Teboulle 2009). Initialize $\boldsymbol{b}_{(1)} = \boldsymbol{a}_{(0)} = \mathbf{0}$, $q_{(1)} = 1$, and $p = 1$, the APG algorithm repeats the following steps until convergence:

$$\boldsymbol{a}_{(p)} = \text{argmin}_{\boldsymbol{a}} \left\{ \frac{L}{2} \left\| \boldsymbol{a} - \boldsymbol{b}_{(p)} + \frac{\frac{\partial \mathcal{F}}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{b}_{(p)}}}{L} \right\|_2^2 + \mathcal{G}(\boldsymbol{a}) \right\},$$

$$q_{(p+1)} = \frac{1 + \sqrt{1 + 4q_{(p)}^2}}{2},$$

$$\boldsymbol{b}_{(p+1)} = \boldsymbol{a}_{(p)} + \frac{q_{(p)} - 1}{q_{(p+1)}} (\boldsymbol{a}_{(p)} - \boldsymbol{a}_{(p-1)}).$$

Convergence of the APG Algorithm. If $\{\boldsymbol{a}_{(p)}\}$ is the sequence generated by the above procedure. Then within $T = O(\sqrt{L/\epsilon})$ iterations, $\boldsymbol{a}_{(T)}$ is such that $\|\boldsymbol{a}_{(T)} - \boldsymbol{a}^\star\| \leq \epsilon$, where $\boldsymbol{a}^\star$ is the minimizer of (4).

Evaluation of Gradient for Dictionary Update

Lemma 2. *The gradient of* $F(\boldsymbol{D}_i) = \sum_{\tau=t-\omega+1}^{t} \|\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}\|^2$ *with respect to* $\boldsymbol{D}_i$ *is:*

$$2\beta \sum_{\tau=t-\omega+1}^{t} h_{\tau_i} \kappa(\boldsymbol{X}_\tau, \boldsymbol{D}_i) \left((\boldsymbol{X}_\tau + \boldsymbol{D}_i)^{-1} - \frac{\boldsymbol{D}_i^{-1}}{2} \right)$$
$$- 2\beta \sum_{\tau=t-\omega+1}^{t} \sum_{j \neq i} h_{\tau_i} h_{\tau_j} \kappa(\boldsymbol{D}_j, \boldsymbol{D}_i) \left((\boldsymbol{D}_j + \boldsymbol{D}_i)^{-1} - \frac{\boldsymbol{D}_i^{-1}}{2} \right).$$

Proof. We can rewrite $\|\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}\|^2$ as $(\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i})^\top (\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i})$. Now,

$$(\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i})^\top (\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}) = \phi(\boldsymbol{D}_i)^\top \phi(\boldsymbol{D}_i)$$
$$- (\boldsymbol{Z}_\tau h_{\tau_i})^\top \phi(\boldsymbol{D}_i) - \phi(\boldsymbol{D}_i)^\top (\boldsymbol{Z}_\tau h_{\tau_i}) + (\boldsymbol{Z}_\tau h_{\tau_i})^\top (\boldsymbol{Z}_\tau h_{\tau_i}).$$

For the purposes of evaluating the gradient (w.r.t. to $\boldsymbol{D}_i$) only the $(\boldsymbol{Z}_\tau h_{\tau_i})^\top \phi(\boldsymbol{D}_i)$ and $\phi(\boldsymbol{D}_i)^\top (\boldsymbol{Z}_\tau h_{\tau_i})$ terms matter (as $\phi(\boldsymbol{D}_i)^\top \phi(\boldsymbol{D}_i) = 1$). In the following, we use F to denote $F(\boldsymbol{D}_i)$. The gradient of F with respect to $\boldsymbol{D}_i$ is:

$$\frac{\partial F}{\partial \boldsymbol{D}_i} = \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} \|\phi(\boldsymbol{D}_i) - \boldsymbol{Z}_\tau h_{\tau_i}\|^2$$
$$= \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} \left(-(\boldsymbol{Z}_\tau h_{\tau_i})^\top \phi(\boldsymbol{D}_i) - \phi(\boldsymbol{D}_i)^\top (\boldsymbol{Z}_\tau h_{\tau_i}) \right)$$
$$= -2 \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} h_{\tau_i} \boldsymbol{Z}_\tau^\top \phi(\boldsymbol{D}_i)$$
$$= -2 \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} h_{\tau_i} \left(\kappa(\boldsymbol{X}_\tau, \boldsymbol{D}_i) - \sum_{j \neq i} h_{\tau_j} \kappa(\boldsymbol{D}_j, \boldsymbol{D}_i) \right). \tag{5}$$

The first part of the gradient of (5) is:

$$-2 \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} h_{\tau_i} \kappa(\boldsymbol{X}_\tau, \boldsymbol{D}_i)$$
$$= 2\beta \sum_{\tau=t-\omega+1}^{t} h_{\tau_i} \kappa(\boldsymbol{X}_\tau, \boldsymbol{D}_i) \left((\boldsymbol{X}_\tau + \boldsymbol{D}_i)^{-1} - \frac{\boldsymbol{D}_i^{-1}}{2} \right). \tag{6}$$

The second part of the gradient of (5) is:

$$2 \frac{\partial}{\partial \boldsymbol{D}_i} \sum_{\tau=t-\omega+1}^{t} \sum_{j \neq i} h_{\tau_i} h_{\tau_j} \kappa(\boldsymbol{D}_j, \boldsymbol{D}_i)$$
$$= -2\beta \sum_{\tau=t-\omega+1}^{t} \sum_{j \neq i} h_{\tau_i} h_{\tau_j} \kappa(\boldsymbol{D}_j, \boldsymbol{D}_i) \left((\boldsymbol{D}_j + \boldsymbol{D}_i)^{-1} - \frac{\boldsymbol{D}_i^{-1}}{2} \right). \tag{7}$$

Adding (6) and (7) proves the claimed bound. □

[6]The gradient of a function $\mathcal{F}$ is Lipschitz continuous with parameter L if $\|\frac{\partial \mathcal{F}}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{x}} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{y}}\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$.

References

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

Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning* 4(1):1–106.

Bao, C.; Wu, Y.; Ling, H.; and Ji, H. 2012. Real time robust L1 tracker using accelerated proximal gradient approach. In *CVPR*, 1830–1837.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Bonnabel, S. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control* 58(9):2217–2229.

Cherian, A., and Sra, S. 2014. Riemannian sparse coding for positive definite matrices. In *ECCV*, 299–314.

Elad, M. 2010. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer.

Gao, S.; Tsang, I. W.-H.; and Chia, L.-T. 2010. Kernel sparse representation for image classification and face recognition. In *ECCV*, 1–14.

Guo, K.; Ishwar, P.; and Konrad, J. 2013. Action recognition from video using feature covariance matrices. *IEEE Transactions on Image Processing* 22(6):2479–2494.

Haines, T. S., and Xiang, T. 2014. Background subtraction with dirichlet process mixture models. *TPAMI* 36(4):670–683.

Harandi, M.; Sanderson, C.; Hartley, R.; and Lovell, B. 2012. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *ECCV*, 216–229.

Harandi, M. T.; Salzmann, M.; and Hartley, R. 2014. From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices. In *ECCV*, 17–32.

Harandi, M. T.; Salzmann, M.; and Porikli, F. M. 2014. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *CVPR*, 1003–1010.

Ho, J.; Xie, Y.; and Vemuri, B. 2013. On a nonlinear generalization of sparse coding and dictionary learning. In *ICML*, 1480–1488.

Hu, W.; Li, X.; Luo, W.; Zhang, X.; Maybank, S.; and Zhang, Z. 2012. Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model. *TPAMI* 34(12):2420–2440.

Kasiviswanathan, S.; Wang, H.; Banerjee, A.; and Melville, P. 2012. Online L1-dictionary learning with application to novel document detection. In *NIPS*, 2267–2275.

Lan, X.; Ma, A. J.; and Yuen, P. C. 2014. Multi-cue visual tracking using robust feature-level fusion based on joint sparse representation. In *CVPR*, 1194–1201.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient- based learning applied to document recognition. *IEEE Transactions on Automatic Control* 86(11):2278–2324.

Lu, C.; Shi, J.; and Jia, J. 2013. Online robust dictionary learning. In *CVPR*, 415–422.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2010. Online learning for matrix factorization and sparse coding. *JMLR* 11:19–60.

Narayana, M.; Hanson, A.; and Learned-Miller, E. 2012. Background modeling using adaptive pixelwise kernel variances in a hybrid feature space. In *CVPR*, 2104–2111.

Nesterov, Y. 1983. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady* 27(2):372–376.

Nguyen, H.; Patel, V.; Nasrabadi, N.; and Chellappa, R. 2012. Kernel dictionary learning. In *ICASSP*, 2021–2024.

Olshausen, B., and Field, D. 1997. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research* 37(23):3311–3325.

Pennec, X.; Fillard, P.; and Ayache, N. 2006. A riemannian framework for tensor computing. *IJCV* 66(1):41–66.

Porikli, F.; Tuzel, O.; and Meer, P. 2006. Covariance tracking using model update based on lie algebra. In *CVPR*, 728–735.

Ross, D.; Lim, J.; Lin, R.; and Yang, M. 2008. Incremental learning for robust visual tracking. *IJCV* 77:125–141.

Sheikh, Y., and Shah, M. 2005. Bayesian modeling of dynamic scenes for object detection. *TPAMI* 27(11):1778–1792.

Sra, S., and Cherian, A. 2011. Generalized dictionary learning for symmetric positive definite matrices with application to nearest neighbor retrieval. In *ECML*. Springer. 318–332.

Sra, S. 2012. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *NIPS*, 144–152.

Wang, N.; Wang, J.; and Yeung, D.-Y. 2013. Online robust non-negative dictionary learning for visual tracking. In *ICCV*, 657–664.

Xu, J.; Ithapu, V. K.; Mukherjee, L.; Rehg, J. M.; and Singh, V. 2013. GOSUS: Grassmannian Online Subspace Updates with Structured-sparsity. In *ICCV*, 3376–3383.

Yang, M.; Zhang, L.; Fengb, X.; and Zhang, D. 2011. Fisher discrimination dictionary learning for sparse representation. In *ICCV*, 543–550.

Zhang, S.; Yao, H.; Liu, S.; Chen, X.; and Gao, W. 2008. A covariance-based method for dynamic background subtraction. *ICPR* 3141–3144.

Zhang, S.; Yao, H.; Sun, X.; and Lu, X. 2013a. Sparse coding based visual tracking: Review and experimental comparison. *Pattern Recognition* 46(7):1772–1788.

Zhang, S.; Yao, H.; Zhou, H.; Sun, X.; and Liu, S. 2013b. Robust visual tracking based on online learning sparse representation. *Neurocomputing* 100:31–40.