Sparse Coding for Third-order Super-symmetric Tensor Descriptors with Application to Texture Recognition

Piotr Koniusz
National ICT Australia (NICTA),
Canberra Research Laboratory
piotr.koniusz@nicta.com.au

Anoop Cherian
ARC Centre of Excellence for Robotic Vision,
Australian National University, Canberra
anoop.cherian@anu.edu.au

Abstract

Super-symmetric tensors – a higher-order extension of scatter matrices – are becoming increasingly popular in machine learning and computer vision for modeling data statistics, co-occurrences, or even as visual descriptors. They were shown recently to outperform second-order approaches [18], however, the size of these tensors are exponential in the data dimensionality, which is a significant concern. In this paper, we study third-order super-symmetric tensor descriptors in the context of dictionary learning and sparse coding. For this purpose, we propose a novel non-linear third-order texture descriptor. Our goal is to approximate these tensors as sparse conic combinations of atoms from a learned dictionary. Apart from the significant benefits to tensor compression that this framework offers, our experiments demonstrate that the sparse coefficients produced by this scheme lead to better aggregation of high-dimensional data and showcase superior performance on two common computer vision tasks compared to the state of the art.

1. Introduction

Recent times have witnessed an increasing trend in several machine learning and computer vision applications to use rich representations that capture the inherent structure and statistics of the data. A few notable such representations are histograms, strings, covariances, trees, and graphs. The goal of this paper is to study a new class of structured data descriptors – third-order super-symmetric tensors – in the context of sparse coding and dictionary learning. Tensors are often used to capture higher-order moments of data distributions such as the covariance, skewness, or kurtosis and have been used as data descriptors in several computer vision applications. In region covariances [36], a covariance matrix – computed on multi-modal features from an image region – is used as a descriptor for the region and is useful for object tracking, retrieval, texture recognition, and video analysis [36, 39, 30, 21, 10]. Given bag-of-words histograms or local descriptor vectors from an image, a second-order co-occurrence pooling of these vectors captures the occurrences of two features together in an image and is recently shown to provide superior performance in semantic segmentation and visual concept detection, compared to their first-order counterparts [5, 18]. A natural extension of the idea is to use higher-order pooling operators, an extensive experimental analysis of which is provided in [18]. Their paper shows that pooling using third-order super-symmetric tensors can significantly improve upon the second-order descriptors, *e.g.*, by more than 7% MAP on the challenging PASCAL VOC07 dataset.

However, given that the size of the tensors increases exponentially against the dimensionality of their first-order counterpart, efficient representations are extremely important for applications that use these higher-order descriptors. To this end, the goal of this paper is to study these descriptors in the classical dictionary learning and sparse coding setting [27]. Using the properties of super-symmetric tensors, we formulate a novel optimization objective (Sections 6, 7) in which each third-order data tensor is approximated by a sparse non-negative linear combination of positive semi-definite matrices. Although our objective is non-convex – typical to several dictionary learning algorithms – we show that our objective is convex in each variable, thus allowing a block-coordinate descent scheme for the optimization. Experiments (Section 8) on the PASCAL VOC07 dataset show that the compressed coefficient vectors produced by our sparse coding preserve the discriminative properties of the original tensors and provide robust tensor compression and aggregation. Inspired by the merits of third-order pooling proposed in [18], we further introduce a novel tensor descriptor for texture recognition via the linearization of explicitly defined RBF kernels, and show that sparse coding of these novel tensors performs better than the state-of-the-art descriptors used for texture recognition.

In summary, our contributions are: i) a novel third-order texture descriptor[1], ii) novel third-order tensor dictionary learning and sparse coding, and iii) theoretical guarantees on the existence of the sparse decomposition.

[1] Note that, third-order descriptors can be easily aggregated from CNN features similar to [9]; however, the choice of input features is complementary to our main goal - theoretical foundations of third-order tensor coding.

2. Related Work

Dictionary learning and sparse coding [8, 27] methods have significantly contributed to improving the performance of numerous applications in computer vision and image processing. While these algorithms assume a Euclidean vectorial representation of the data, there have been extensions to other data descriptors such as tensors, especially symmetric positive definite matrices [6, 11, 34]. These extensions typically use non-Euclidean geometries defined by a similarity measure. Popular choices of such measures for positive definite matrices are the log-determinant divergence [34], the log-Euclidean metric [1], and the affine invariant Riemannian metric [28]. However, the third-order tensors considered in this paper are neither positive definite nor there are any standard similarity measures known, apart from the Euclidean distance. Thus, extending these prior methods to our setting is infeasible, demanding novel formulations.

Third-order tensors have been used for various tasks. Spatio-temporal third-order tensor on video data for activity analysis and gesture recognition is proposed in [15] . Non-negative factorization is applied to third-order tensors in [33] for image denoising. Multi-linear algebraic methods for tensor subspace learning are surveyed in [24]. Tensor textures are used in the context of texture rendering in computer vision applications in [38]. Similar to eigenfaces for face recognition, multi-linear algebra based techniques for face recognition use third-order tensors in [37]. However, these applications work with a single tensor; the objective of this paper is to learn the underlying structure of a large collection of such tensors generated from visual data using the framework of dictionary learning and sparse coding, which to the best of our knowledge is a novel proposition.

In addition to the dictionary learning framework, we also introduce an image descriptor for textures. While we are not aware of any previous works that propose third-order tensors as image region descriptors, the most similar methods to our approach are i) third-order probability matrices for image splicing detection [42] and ii) third-order global image descriptors that aggregates SIFT vectors into autocorrelation tensors [18]. In contrast, our descriptor, is assembled from elementary signals such as intensity and its first- and second-order derivatives, analogous to covariance descriptors [36], but demonstrating superior accuracy. The formulation chosen by us for texture recognition also differs from prior works such as kernel descriptors [2] and convolutional kernel networks [25].

3. Notations

Before we proceed, we briefly review our notations next. We use bold-face upper-case calligraphic and regular letters for third-order tensors and matrices, respectively, bold-face lower-case letters for vectors and normal fonts for scalars. Each second-order tensor along the third mode of a third-order tensor is called a *slice*. Using Matlab style notation, the s-th slice of $\boldsymbol{\mathcal{X}}$ is given by $\boldsymbol{\mathcal{X}}_{:,:,s}$. The operation $\uparrow\otimes$ stands for an outer product of a second-order tensor with a vector. For example, $\boldsymbol{\mathcal{Y}} = \mathbf{Y}\uparrow\otimes\mathbf{y}$ produces a third-order tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{d_1\times d_2\times d_3}$ from a matrix $\mathbf{Y} \in \mathbb{R}^{d_1\times d_2}$ and a vector $\mathbf{y} \in \mathbb{R}^{d_3}$, where the s-th slice of $\boldsymbol{\mathcal{Y}}$ is given by $\mathbf{Y}y_s$, y_s being the s-th dimension of $\mathbf{y}$. $\mathcal{I}_N$ stands for the index set $\{1, 2, ..., N\}$. We denote the space of $d \times d$ positive semi-definite matrices as $\mathcal{S}_+^d \subset \mathbb{R}^{d\times d}$, the space of super-symmetric tensors of dimension $d \times d \times d$ as $\mathfrak{S}^d \subset \mathbb{R}^{d\times d\times d}$ and the space of tensors $\mathbb{R}^{d\times\ldots\times d}$ with r modes as $\mathbb{R}^{\times_r d}$. Going by the standard terminology in higher-order tensor factorization literature [7, 20], we define a *core tensor* as the analogous of the singular value matrix produced by SVD in the second-order case. A core tensor need not be diagonal in general and depends on the chosen decomposition.

4. Background

In this section, we review super-symmetric tensors and their properties, followed by a brief exposition of the method described in [18] for generating tensor descriptors for an image. The latter will come useful when introducing our new texture descriptors.

4.1. Third-Order Super-Symmetric Tensors

We define a super-symmetric tensor descriptor as follows:

Definition 1. *Suppose $\mathbf{x}_n \in \mathbb{R}_+^d, \forall n \in \mathcal{I}_N$ represents data vectors from an image, then a third-order super-symmetric tensor (TOSST) descriptor $\boldsymbol{\mathcal{X}} \in \mathfrak{S}^d$ of these data vectors is given by:*

$$\boldsymbol{\mathcal{X}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n\mathbf{x}_n^T)\uparrow\otimes\mathbf{x}_n. \tag{1}$$

In an object recognition setting, the data vectors are usually SIFT descriptors, Gabor filters, or other primitives extracted from the image, while the tensor descriptor is obtained as the third-order autocorrelation tensor via applying (1).

The following properties of the TOSST descriptor are useful in the sequel.

Proposition 1. *For a TOSST descriptor $\boldsymbol{\mathcal{X}} \in \mathfrak{S}^d$, we have:*

1. *Super-Symmetry: $\boldsymbol{\mathcal{X}}_{i,j,k} = \boldsymbol{\mathcal{X}}_{\Pi(i,j,k)}$ for indexes i, j, k and their permutation given by Π, $\forall\Pi$.*
2. *Every slice is positive semi-definite, that is, $\boldsymbol{\mathcal{X}}_{:,:,s} \in \mathcal{S}_+^d, \quad \forall s \in \mathcal{I}_d$.*
3. *Indefiniteness,* i.e.*, under a CP decomposition [20], it can have positive, negative, or zero entries in its core-tensor – equivalent of eigenvalues in matrix case.*

Proof. The first two properties can be proved directly from (1). To prove the last property, note that for a TOSST tensor $\boldsymbol{\mathcal{X}}$ and some $\mathbf{z} \in \mathbb{R}^d, \mathbf{z} \neq 0$, we have $((\boldsymbol{\mathcal{X}} \otimes_1 \mathbf{z}) \otimes_2 \mathbf{z}) \otimes_3 \mathbf{z} = \sum_{s=1}^{d} z_s(\mathbf{z}^T \mathbf{X}_s \mathbf{z})$ where $\mathbf{X}_s$ is the s-th slice of $\boldsymbol{\mathcal{X}}$. While $\mathbf{z}^T \mathbf{X}_s \mathbf{z} \geq 0$, the tensor product could be negative for $\mathbf{z} < 0$. In the above, $\otimes_i$ denotes tensor product in the i-th mode [16]. □

Due to the indefiniteness of the tensors, we cannot use some of the well-known distances on the manifold of SPD matrices [1, 28]. Thus, we restrict to using the Euclidean distance for the derivations and experiments in this paper.

Among several properties of tensors, one that is important and is typically preserved by tensor decompositions is the *tensor rank* defined below.

Definition 2 (Tensor Rank)**.** *Given a tensor $\boldsymbol{\mathcal{X}} \in \mathfrak{S}^d$, its* tensor rank $\mathrm{TRank}(\boldsymbol{\mathcal{X}})$ *is defined as the minimum p such that $\boldsymbol{\mathcal{X}}_{:,:,s} \in \mathrm{Span}\,(M_1, M_2, ..., M_p)$ for all $s \in \mathcal{I}_d$, where each $M_i \in \mathcal{S}_+^d$ is rank-one.*

4.2. Third-Order Global Image Descriptor

In what follows, we outline a global image descriptor for the object category recognition from [18] used in our experiments on the PASCAL VOC07 dataset. This descriptor is a specific example of the TOSST approach. To the unfamiliar reader, we also illustrate below, some preliminary results for this descriptor to highlight its robustness. The simplicity of the approach and good results it produces, motivate us to develop a third-order descriptor for textures (Section 5) as a natural extension of second-order region covariance descriptors (RCD), which are very popular for texture recognition. Furthermore, the high dimensionality of TOSST descriptors provides a strong motivation to extend the classical dictionary learning and sparse coding algorithm to generate compact tensor representations (Section 6).

The Third-order Global Image Descriptor is based on SIFT descriptors [23] aggregated into a third-order autocorrelation tensor. This step precedes applying Higher Order Singular Value Decomposition [20, 16] for whitening via Power Normalisation [18] performed on the core tensor. The following steps represent this tensor descriptor:

$$\boldsymbol{\mathcal{V}}_i = \uparrow \otimes_r \mathbf{v}_i, \ \forall i \in \mathcal{I} \tag{2}$$

$$\bar{\boldsymbol{\mathcal{V}}} = \underset{i \in \mathcal{I}}{\mathrm{Avg}}(\boldsymbol{\mathcal{V}}_i) \tag{3}$$

$$(\boldsymbol{\mathcal{E}}; \boldsymbol{A}) = \mathrm{HOSVD}(\bar{\boldsymbol{\mathcal{V}}}) \tag{4}$$

$$\hat{\boldsymbol{\mathcal{E}}} = \mathrm{Sgn}(\boldsymbol{\mathcal{E}})\,|\boldsymbol{\mathcal{E}}|^{\gamma_e} \tag{5}$$

$$\hat{\boldsymbol{\mathcal{V}}} = ((\hat{\boldsymbol{\mathcal{E}}} \otimes_1 \boldsymbol{A}) ...) \otimes_r \boldsymbol{A} \tag{6}$$

$$\boldsymbol{\mathcal{X}} = \mathrm{Sgn}(\hat{\boldsymbol{\mathcal{V}}})\,|\hat{\boldsymbol{\mathcal{V}}}|^{\gamma_c} \tag{7}$$

Equations (2, 3) assemble a higher-order autocorrelation tensor $\bar{\boldsymbol{\mathcal{V}}}$ per image. First, the outer product $\uparrow \otimes_r$ of order r [20, 16, 18] is applied to the local image descriptors $\mathbf{v}_i \in \mathbb{R}^d$ from an index set $\mathcal{I}$ *i.e.*, indexes of SIFT vectors from an image. This results in $|\mathcal{I}|$ autocorrelation matrices $(\uparrow \otimes_2 \mathbf{v}_i \triangleq \mathbf{v}_i \mathbf{v}_i^T)$ or third-order tensors $\boldsymbol{\mathcal{V}}_i \in \mathfrak{S}^d : \boldsymbol{\mathcal{V}}_i = (\uparrow \otimes_3 \mathbf{v}_i \triangleq (\mathbf{v}_i \mathbf{v}_i^T) \uparrow \otimes \mathbf{v}_i)$ of rank one if $r = 2$ or $r = 3$, respectively. These rank-one tensors are then averaged by Avg, resulting in tensor $\bar{\boldsymbol{\mathcal{V}}} \in \mathfrak{S}^d$ that describes globally the image and could be used as a training sample. However, practical image representations have to deal with the so-called burstiness which is "*the property that a given visual element appears more times in an image than a statistically independent model would predict*" [13].

Power Normalization (PN) [3, 29, 13] is known to suppress burstiness. Therefore, equations (4-6) and (7) apply two-stage pooling with the eigenvalue- and coefficient-wise PN respectively. In equation (4), operator $\mathrm{HOSVD} : \mathfrak{S}^d \to \left(\mathbb{R}^{\times_r d}, \mathbb{R}^{d \times d}\right)$ decomposes tensor $\bar{\boldsymbol{\mathcal{V}}}$ into a core tensor $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{\times_r d}$ with eigenvalues and an orthonormal factor matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, which can be interpreted as the principal components in r modes. PN is then applied element-wise by equation (5) to the eigenvalues of $\boldsymbol{\mathcal{E}}$ to even out their contributions. Tensor $\hat{\boldsymbol{\mathcal{V}}} \in \mathbb{R}^{\times_r d}$ is then assembled in equation (6) by the tensor product $\otimes_i$ in the i-th mode [16]. Then, the coefficient-wise PN acting on $\hat{\boldsymbol{\mathcal{V}}}$ produces tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\times_r d}$ in equation (7).

In our experiments, we use $r = 3$ and i) $\boldsymbol{\mathcal{X}} = \bar{\boldsymbol{\mathcal{V}}}$ when PN is disabled, *e.g.*, $\gamma_e = \gamma_c = 1$, or ii) apply the above HOSVD approach if $0 < \gamma_e < 1 \wedge 0 < \gamma_c \leq 1$. Therefore, the final descriptor $\boldsymbol{\mathcal{X}}$ in equation (7) represents an image by a TOSST which consists entirely of SPD matrix slices, if no PN is used. Preliminary results in Table 1 show that the third-order descriptor outperforms the second-order approach.

	PASCAL VOC07	Caltech 101	Flowers 102
second-order	54.0%	78.5%	83.1%
third-order	**62.7%**	**83.9%**	**89.0%**

Table 1: Evaluation of the Third-order Global Image Descriptor.

5. TOSST Texture Descriptors

Many region descriptors for texture recognition are described in the literature [5, 21, 36]. Their construction generally consists of the following steps:

i) For an image $\mathbf{I}$ and a region $\mathcal{R}$ in it, extract feature statistics from the region. If (x, y) represent pixel coordinates in $\mathcal{R}$, then a feature vector from $\mathcal{R}$ at (x, y) is given as:

$$\boldsymbol{\phi}_{xy}(\mathbf{I}) = \left[\tfrac{x-x_0}{w-1}, \tfrac{y-y_0}{h-1}, I_{xy}, \tfrac{\partial I_{xy}}{\partial x}, \tfrac{\partial I_{xy}}{\partial y}, \tfrac{\partial^2 I_{xy}}{\partial x^2}, \tfrac{\partial^2 I_{xy}}{\partial y^2}\right]^T, \tag{8}$$

where (x_0, y_0), w and h are the coordinate origin, width, and height of $\mathcal{R}$, respectively.

ii) Given feature vectors $\boldsymbol{\phi}_{xy}$, compute a covariance matrix $\boldsymbol{\Phi}(\mathbf{I}, \mathcal{R}) = \frac{1}{|\mathcal{R}|-1} \sum_{(x,y) \in \mathcal{R}} \left(\boldsymbol{\phi}_{xy}(\mathbf{I}) - \bar{\boldsymbol{\phi}}\right) \left(\boldsymbol{\phi}_{xy}(\mathbf{I}) - \bar{\boldsymbol{\phi}}\right)^T$,

where $\bar{\phi}$ is the mean over $\phi_{xy}(\mathbf{I})$ from region $\mathcal{R}$ [36, 21]. Alternatively, one can compute an autocorrelation matrix $\boldsymbol{\Phi}(\mathbf{I},\mathcal{R})=\frac{1}{|\mathcal{R}|}\sum_{(x,y)\in\mathcal{R}}\boldsymbol{\phi}_{xy}(\mathbf{I})\boldsymbol{\phi}_{xy}(\mathbf{I})^T$ as proposed in [5, 18].

In this work, we make modifications to the steps listed above. Instead of using linear representations for $\boldsymbol{\phi}_{xy}(\mathbf{I})$, we use RBF kernels that are known to improve classification performance. The concatenated statistics in step (i) can be seen as linear features forming linear kernel K_{lin}:

$$K_{lin}((x,y,\mathbf{I}^a),(x',y',\mathbf{I}^b)) = \boldsymbol{\phi}_{xy}(\mathbf{I}^a)^T\boldsymbol{\phi}_{x'y'}(\mathbf{I}^b). \quad (9)$$

We go a step further and define the following sum kernel K_{rbf} composed of $\tau=7$ RBF kernels G:

$$K_{rbf}\left((x,y,\mathbf{I}^a),\left(x',y',\mathbf{I}^b\right)\right)=\sum_{i=1}^{\tau}G_{\sigma_i}(\phi^i_{xy}(\mathbf{I}^a)-\phi^i_{x'y'}(\mathbf{I}^b)), \quad (10)$$

where $\phi^i_{xy}(\mathbf{I})$ is the i-th feature in equation (8) while $G_{\sigma_i}(\mathbf{u}-\mathbf{u}') = \exp(-\|\mathbf{u}-\mathbf{u}'\|^2/2\sigma_i^2)$ is the so-called relative compatibility kernel that measures the compatibility between features of type i in each of the image regions.

The next step involves linearization of Gaussian kernels G_{σ_i} for $i=1,...,\tau$ to obtain feature maps that expresses the kernel K_{rbf} by the dot-product. In what follows, we use a fast approximation method based on probability product kernels [12]. Specifically, we employ the inner product for the d'-dimensional isotropic Gaussian distribution:

$$G_\sigma\left(\mathbf{u}-\mathbf{u}'\right)=\left(\frac{2}{\pi\sigma^2}\right)^{\frac{d'}{2}}\int_{\boldsymbol{\zeta}\in\mathbb{R}^{d'}}G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta})\,G_{\sigma/\sqrt{2}}(\mathbf{u}'-\boldsymbol{\zeta})\,\mathrm{d}\boldsymbol{\zeta}. \quad (11)$$

Equation (11) is then approximated by replacing the integral with the sum over Z pivots $\boldsymbol{\zeta}_1,...,\boldsymbol{\zeta}_Z$:

$$G_\sigma(\mathbf{u}-\mathbf{u}') \approx \left\langle\sqrt{w}\boldsymbol{\phi}(\mathbf{u}),\sqrt{w}\boldsymbol{\phi}(\mathbf{u}')\right\rangle,$$
$$\boldsymbol{\phi}(\mathbf{u}) = \left[G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta}_1),...,G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta}_Z)\right]^T. \quad (12)$$

A weighting constant w relates to the width of the rectangle in the numerical integration and is factored out by the ℓ_2 normalization:

$$G_\sigma(\mathbf{u}-\mathbf{u}') = \frac{G_\sigma(\mathbf{u}-\mathbf{u}')}{G_\sigma(\mathbf{u}-\mathbf{u})G_\sigma(\mathbf{u}'-\mathbf{u}')} \approx \quad (13)$$
$$\left\langle\frac{\sqrt{w}\boldsymbol{\phi}(\mathbf{u})}{\|\sqrt{w}\boldsymbol{\phi}(\mathbf{u})\|_2},\frac{\sqrt{w}\boldsymbol{\phi}(\mathbf{u}')}{\|\sqrt{w}\boldsymbol{\phi}(\mathbf{u}')\|_2}\right\rangle=\left\langle\frac{\boldsymbol{\phi}(\mathbf{u})}{\|\boldsymbol{\phi}(\mathbf{u})\|_2},\frac{\boldsymbol{\phi}(\mathbf{u}')}{\|\boldsymbol{\phi}(\mathbf{u}')\|_2}\right\rangle.$$

The task of linearizing each G_{σ_i} in equation (10) is trivial as these kernels use one-dimensional variables ($d'=1$). Thus, we uniformly sample domains of each variable and use Z=5. With this tool at hand, we rewrite kernel K_{rbf} as:

$$K_{rbf}((x,y,\mathbf{I}^a),(x',y',\mathbf{I}^b)) \approx \left\langle\boldsymbol{v}^a_{xy},\boldsymbol{v}^b_{x'y'}\right\rangle, \quad (14)$$

where vector $\boldsymbol{v}^a_{xy}$ is composed by τ sub-vectors $\frac{\phi^i_{xy,\sigma_i}}{\|\phi^i_{xy,\sigma_i}\|_2}$ and each sub-vector $i=1,...,\tau$ is a result of linearization by equations (11-13). We use a third-order aggregation of $\boldsymbol{v}$ according to equation (1) to generate our TOSST descriptor for textures. For comparisons, we also aggregate $\boldsymbol{\phi}$ from equation (8) into a third-order linear descriptor. See Appendix A for the derivation of the third-order aggregation procedure.

6. Problem Formulation

Suppose we are given data tensors $\boldsymbol{\mathcal{X}}_n, n\in\mathcal{I}_N$, each $\boldsymbol{\mathcal{X}}_n\in\mathfrak{S}^d$. We want to learn a tensor dictionary $\overline{\boldsymbol{\mathcal{B}}}$ with atoms $\boldsymbol{\mathcal{B}}_1,\boldsymbol{\mathcal{B}}_2,...,\boldsymbol{\mathcal{B}}_K$, where each $\boldsymbol{\mathcal{B}}_k\in\mathfrak{S}^d$ consists of d-slices. Let the s-th slice of the k-th atom be given by $\mathbf{B}^s_k$ where $s\in\mathcal{I}_d$, $k\in\mathcal{I}_K$ and each $\mathbf{B}^s_k\in\mathcal{S}^d_+$. Then, the problem of dictionary learning and sparse coding can be formulated as follows for sparse coefficient vectors $\boldsymbol{\alpha}^n\in\mathbb{R}^K, n\in\mathcal{I}_N$:

$$\underset{\overline{\boldsymbol{\mathcal{B}}}\ \boldsymbol{\alpha}^1,...,\boldsymbol{\alpha}^N}{\arg\min}\sum_{n=1}^{N}\left\|\boldsymbol{\mathcal{X}}_n-\sum_{k=1}^{K}\boldsymbol{\mathcal{B}}_k\alpha^n_k\right\|^2_F+\lambda\|\boldsymbol{\alpha}^n\|_1. \quad (15)$$

Note that in the above formulation, third-order dictionary atoms $\boldsymbol{\mathcal{B}}_k$ are multiplied by scalars α^n_k. For K atoms, each of size d^3, there are Kd^3 parameters to be learned in the dictionary learning process. An obvious consequence of this reconstruction process is that we require $N\gg K$ to prevent overfitting. To circumvent this bottleneck and work in the regime $dN\gg K$, the reconstruction process is amended such that every dictionary atom is represented by an outer product of a second-order symmetric positive semi-definite matrix $\mathbf{B}_k$ and a vector $\mathbf{b}_k$. Such a decomposition/approximation will reduce the number of parameters from Kd^3 to $K(d^2+d)$. Using this strategy, we re-define dictionary $\overline{\boldsymbol{\mathcal{B}}}$ to be represented by atom pairs $\overline{\boldsymbol{\mathcal{B}}}\equiv\{(\mathbf{B}_k,\mathbf{b}_k)\}_{k=1}^K$ such that $\boldsymbol{\mathcal{B}}_k=\mathbf{B}_k\uparrow\otimes\mathbf{b}_k$. Note that the tensor rank of $\boldsymbol{\mathcal{B}}_k$ under this new representation is equal to the $\mathrm{Rank}(\mathbf{B}_k)$ as formally stated below.

Proposition 2 (Atom Rank). *Let* $\boldsymbol{\mathcal{B}}=\mathbf{B}\uparrow\otimes\mathbf{b}$ *and* $\|\mathbf{b}\|_1\neq 0$, *then* $\mathrm{TRank}(\boldsymbol{\mathcal{B}})=\mathrm{Rank}(\mathbf{B})$.

Proof. Rank of $\boldsymbol{\mathcal{B}}$ is the smallest p satisfying $(\mathbf{B}\uparrow\otimes\mathbf{b})_{:,:,s}=\mathbf{B}\cdot b_s\in\mathrm{Span}\,(M_1,M_2,...,M_p)\,,\ \forall s\in\mathcal{I}_d$, where each $M_i\in\mathcal{S}^d_+$ is rank-one. The smallest p satisfying $\mathbf{B}\in\mathrm{Span}\,(M_1,M_2,...,M_p)$ is the same, since multiplication of a matrix by any non-zero scalar ($\mathbf{B}\cdot b_s$, $b_s\neq 0$) will not change its eigenvectors, which constitute the spanning set. □

Using the above idea, we can rewrite (15) as follows:

$$\underset{\substack{\overline{\boldsymbol{\mathcal{B}}}\\ \boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N}}{\arg\min} \sum_{n=1}^{N} \left\| \boldsymbol{\mathcal{X}}_n - \sum_{k=1}^{K} \left(\mathbf{B}_k \uparrow \otimes \mathbf{b}_k\right) \alpha_k^n \right\|_F^2 + \lambda \left\|\boldsymbol{\alpha}^n\right\|_1. \tag{16}$$

Introducing optimization variables $\boldsymbol{\beta}_k^n$ and rewriting the loss function by taking the slices out, we rewrite (16) as:

$$\underset{\substack{\overline{\boldsymbol{\mathcal{B}}}\\ \boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N}}{\arg\min} \sum_{n=1}^{N} \sum_{s=1}^{S} \left\| \mathbf{X}_n^s - \sum_{k=1}^{K} \mathbf{B}_k \beta_k^{s,n} \right\|_F^2 + \lambda \left\|\boldsymbol{\alpha}^n\right\|_1,$$
$$\text{subject to } \boldsymbol{\beta}_k^n = \mathbf{b}_k \alpha_k^n, \quad \forall k \in \mathcal{I}_K \text{ and } n \in \mathcal{I}_N. \tag{17}$$

We may rewrite the equality constraints in (17) as a proximity constraint in the objective function (using a regularization parameter γ), and constrain the sparse coefficients in $\boldsymbol{\alpha}_n$ to be non-negative, as each slice is positive semi-definite. This is a standard technique used in optimization, commonly referred to as *variable splitting*. We can also normalize the dictionary atoms $\mathbf{B}_k$ to be positive semi-definite and of unit-trace for better numerical stability. In that case, vector atoms $\mathbf{b}_k$ can be constrained to be non-negative (or even better, $\boldsymbol{\beta}_k^n \geq 0$ instead of $\boldsymbol{\alpha}_n \geq 0$ and $\mathbf{b}_k \geq 0$). Introducing chosen constraints, we can rewrite our tensor dictionary learning and sparse coding formulation:

$$\underset{\substack{\overline{\boldsymbol{\mathcal{B}}}\\ \boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N \geq 0}}{\arg\min} \sum_{n=1}^{N} \sum_{s=1}^{S} \left\| \mathbf{X}_n^s - \sum_{k=1}^{K} \mathbf{B}_k \beta_k^{s,n} \right\|_F^2 + \gamma \left\|\boldsymbol{\beta}^{s,n} - \mathbf{b}^s \odot \boldsymbol{\alpha}^n\right\|_2^2 + \lambda \left\|\boldsymbol{\alpha}^n\right\|_1,$$
$$\text{subject to } \mathbf{B}_k \succcurlyeq 0,\ \operatorname{Tr}(\mathbf{B}_k) \leq 1,\ \left\|\mathbf{b}_k\right\|_1 \leq 1,\ k \in \mathcal{I}_K. \tag{18}$$

In the above equation, operator $\odot$ is an element-wise multiplication between vectors. Coefficients γ and λ control the quality of the proximity and sparsity of $\boldsymbol{\alpha}_n$, respectively. The formulation (18) is convex in each variable and can be solved efficiently using block coordinate descent. Note that, we may use a richer characterization of the loss by replacing the Frobenius norm with one based on the Riemannian geometry of positive definite matrices [6]; however, in that case the convexity in each variable will be lost.

Remark 1 (Non-symmetric Tensors)**.** *Note that non-symmetric $\boldsymbol{\mathcal{X}}_n \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ can be coded if the positive semi-definite constraint on $\mathbf{B}_k$ in (18) is dropped, such that $\mathbf{B}_k \in \mathbb{R}^{d_1 \times d_2}$ and $\mathbf{b}_k \in \mathbb{R}^{d_3}$. Other non-negative constraints can also be removed. While this case is a straightforward extension of our formulation, a thorough investigation into it is left as future work.*

Optimization

We propose a block-coordinate descent scheme to solve (18), in which each variable is solved alternately, while keeping other variables fixed. Due to the convexity of each sub-problem, we observe convergence to a local minimum empirically, but a theoretical convergence analysis is difficult due to the non-smoothness of the problem. Initial values of $\mathbf{B}_k$, $\mathbf{b}_k$, and $\boldsymbol{\alpha}^n$ are chosen randomly from the uniform distribution within a prespecified range. Vectors $\boldsymbol{\beta}^{s,n}$ are initialized with the Lasso algorithm. Next, we solve four separate convex sub-problems resulting in updates of $\mathbf{B}_k$, $\mathbf{b}_k$, $\boldsymbol{\beta}^{s,n}$, and $\boldsymbol{\alpha}^n$. For learning second-order matrices $\mathbf{B}_k$, we employ the PQN solver [32] which lets us implement and apply projections into the SPD cone, handle the trace norm or even the rank of matrices (if this constraint is used). Similarly, the projected gradient is used to keep $\mathbf{b}_k$ inside the simplex $\|\mathbf{b}_k\|_1 \leq 1$. For the remaining sub-problems, we use the L-BFGS-B solver [4] which enables us to handle simple box constraints *e.g.*, we split coding for $\boldsymbol{\alpha}^n$ into two non-negative sub-problems $\boldsymbol{\alpha}_+^n$ and

Algorithm 1: Third-order Dictionary Learning and Sparse Coding.

Data: N data tensors $\overline{\boldsymbol{\mathcal{X}}} \equiv \{\boldsymbol{\mathcal{X}}_1, ..., \boldsymbol{\mathcal{X}}_N\}$, proximity and sparsity constants γ and λ, stepsize η, K dictionary atoms $\overline{\boldsymbol{\mathcal{B}}} \equiv \{(\mathbf{B}_k, \mathbf{b}_k)\}_{k=1}^K$ if coding, otherwise $LearnDict = true$

Result: N sparse coeffs. $\overline{\boldsymbol{\alpha}} \equiv \{\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^N\}$, K atoms $\overline{\boldsymbol{\mathcal{B}}} \equiv \{(\mathbf{B}_k, \mathbf{b}_k)\}_{k=1}^K$ if $LearnDict = true$

Initialization:

if $LearnDict$ **then**

- Uniformly sample slices from $\overline{\boldsymbol{\mathcal{X}}}$ and fill $\mathbf{B}_k, \forall k \in \mathcal{I}_K$
- Uniformly sample values in $\left[-\frac{1}{K}; \frac{1}{K}\right]$ to init. $\mathbf{b}_k, \forall k \in \mathcal{I}_K$

end

- Vectorize slices $\mathbf{X}_n^s, \forall s \in \mathcal{I}_S, \forall n \in \mathcal{I}_N$, and atoms $\mathbf{B}_k$, $\forall k \in \mathcal{I}_K$, solve Lasso problem to fill $\boldsymbol{\beta}^{s,n}, \forall s \in \mathcal{I}_S, \forall n \in \mathcal{I}_N$
- Uniformly sample values in $\left[-\frac{1}{K\lambda}; \frac{1}{K\lambda}\right]$ to fill $\boldsymbol{\alpha}^n, \forall n \in \mathcal{I}_N$
- $Objective^{(0)} = 0,\ t = 1$

Main loop:

while $\neg\, Converged$ **do**

if $LearnDict$ **then**

- $\mathbf{B}_k^{(t+1)} = \Pi_{SPD}\left(\mathbf{B}_k^{(t)} - \eta \left.\frac{\partial f}{\partial \mathbf{B}_k}\right|_{\mathbf{B}_k^{(t)}}\right), \forall k \in \mathcal{I}_K$, where f is the cost from (18), projection $\Pi_{SPD}(\mathbf{B}) = \frac{\mathbf{B}^*}{\operatorname{Tr}(\mathbf{B}^*)}$, $\mathbf{B}^* = \boldsymbol{U} \max(\boldsymbol{\lambda}^*, 0) \boldsymbol{V}^T$ for $(\boldsymbol{U}, \boldsymbol{\lambda}^*, \boldsymbol{V}) = SVD(\mathbf{B})$
- $\mathbf{b}_k^{(t+1)} = \Pi_+\left(\mathbf{b}_k^{(t)} - \eta \left.\frac{\partial f}{\partial \mathbf{b}_k}\right|_{\mathbf{b}_k^{(t)}}\right), \forall k \in \mathcal{I}_K$, and $\Pi_+(\mathbf{b}) = \max(\mathbf{b}, 0)$ if Π_+ is used

end

- $\boldsymbol{\beta}^{s,n,(t+1)} = \Pi_+\left(\boldsymbol{\beta}^{s,n,(t)} - \eta \left.\frac{\partial f}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta}_k^{s,n,(t)}}\right), \forall s \in \mathcal{I}_S, \forall n \in \mathcal{I}_N$, use of Π_+ is optional
- $\boldsymbol{\alpha}_n^{(t+1)} = \Pi_+\left(\boldsymbol{\alpha}_n^{(t)} - \eta \left.\frac{\partial f}{\partial \boldsymbol{\alpha}}\right|_{\boldsymbol{\alpha}_n^{(t)}}\right), \forall n \in \mathcal{I}_N$, use of Π_+ is optional
- $Objective^{(t)} = f\left(\overline{\boldsymbol{\mathcal{X}}}, \overline{\mathbf{B}}^{(t)}, \overline{\mathbf{b}}^{(t)}, \overline{\boldsymbol{\alpha}}^{(t)}\right)$, where $\overline{\mathbf{B}}^{(t)}$ and $\overline{\mathbf{b}}^{(t)}$ are K matrix and vector atoms
- $Converged = EvaluateStoppingCriteria\left(t, Objective^{(t)}, Objective^{(t-1)}\right)$

end

$\boldsymbol{\alpha}_-^n$ (unless $\boldsymbol{\alpha}^n \geq 0$) by applying box constraints $\boldsymbol{\alpha}_+^n \geq 0$ and $\boldsymbol{\alpha}_-^n \geq 0$. Finally, we obtain $\boldsymbol{\alpha}^n = \boldsymbol{\alpha}_+^n - \boldsymbol{\alpha}_-^n$ after minimization. For the sparse coding phase (no dictionary learning), we fix $\{(\mathbf{B}_k, \mathbf{b}_k)\}_{k=1}^K$. The algorithm proceeds as above, but without updates of $\mathbf{B}_k$ and $\mathbf{b}_k$. Algorithm 1 shows the important steps of our dictionary learning and sparse coding.

7. Theory

In this section, we analyze the theoretical properties of our sparse coding formulation. We establish that under an approximation setting, a sparse coding of the data tensor exists in some dictionary for every data point. We also provide bounds on the quality of this approximation and its tensor rank. Note that the exact sparse recovery is not mandatory in image classification [41]. Thus, we skip this line of investigation. The following lemma comes useful in the sequel.

Lemma 1 ([35]). *Let* $\mathbf{Y} = \sum_{i=1}^m \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T$, *where each* $\boldsymbol{\phi}_i \in \mathbb{R}^d$ *and* $m = \Omega(d^2)$. *Then, there exist an* $\boldsymbol{\alpha} \in \mathbb{R}_+^m$ *and* $\epsilon \in (0,1)$ *such that:*

$$\mathbf{Y} \preceq \sum_{i=1}^m \alpha_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T \preceq (1+\epsilon)\mathbf{Y}, \tag{19}$$

where $\boldsymbol{\alpha}$ *has* $\mathcal{O}(d\log(d)/\epsilon^2)$ *non-zeros.*

Theorem 1. *Let* $\boldsymbol{\mathcal{X}} = \sum_{i=1}^m (\boldsymbol{\phi}_i \boldsymbol{\phi}_i^T)\uparrow \otimes \boldsymbol{\phi}_i$, *then there exist a second-order dictionary* $\mathbf{B}$ *with* md *atoms and a sparse coefficient vector* $\boldsymbol{\beta} \in \mathbb{R}_+^{md}$ *with* $\mathcal{O}(d^2\log(d)/\epsilon^2)$ *non-zeros, such that for* $\epsilon \in (0,1)$,

$$I)\ \widetilde{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^m (\boldsymbol{\phi}_i \boldsymbol{\phi}_i^T)\uparrow \otimes \bar{\boldsymbol{\beta}}^i \quad \text{and} \quad II)\ \left\| \boldsymbol{\mathcal{X}} - \widetilde{\boldsymbol{\mathcal{X}}} \right\|_F \leq \epsilon \sum_{s=1}^d \|\mathbf{X}^s\|_F,$$

where $\mathbf{X}^s$ *is the* s*-th slice of* $\boldsymbol{\mathcal{X}}$ *and* $\bar{\boldsymbol{\beta}}^i \in \mathbb{R}^d$.

Proof. To prove I: each slice $\mathbf{X}_s = \sum_{i=1}^m (\boldsymbol{\phi}_i \boldsymbol{\phi}_i^T)\phi_i^s$, where ϕ_i^s is the s-th dimension of $\boldsymbol{\phi}_i$. Let $\boldsymbol{\phi}_i' = \boldsymbol{\phi}_i \sqrt{\phi_i^s}$, then $\mathbf{X}^s = \sum_{i=1}^m \boldsymbol{\phi}_i' \boldsymbol{\phi}_i'^T$. If the slices are to be treated independently (as in (18)), then to each slice we can apply Lemma 1 that results in sparse coefficient vectors $\bar{\boldsymbol{\beta}}^i$ having $\mathcal{O}(d\log(d)/\epsilon^2)$ non-zeros. Extending this result to all the d-slices, we obtain the result.
To prove II: substituting $\widetilde{\boldsymbol{\mathcal{X}}}$ with its sparse approximation and using the upper-bound in (19) for each slice, the result follows. □

Theorem 2 (Approximation quality via Tensor Rank). *Let* $\widetilde{\boldsymbol{\mathcal{X}}}$ *be the sparse approximation to* $\boldsymbol{\mathcal{X}} \in \mathfrak{S}^d$ *obtained by solving* (17) *using a dictionary* $\mathbf{B}$ *and if* $\mathrm{Rank}(\mathbf{B})$ *represent the maximum of the rank of any atom, then:*

$$\mathrm{TRank}(\widetilde{\boldsymbol{\mathcal{X}}}) \leq \min\left(\left| \bigcup_{s=1}^d \mathrm{Supp}(\mathbf{X}^s) \right| \mathrm{Rank}(\mathbf{B}),\ d^2 \right), \tag{20}$$

where $\mathrm{Supp}(\mathbf{X}^s)$ *is the support set produced by* (17) *for slice* $\mathbf{X}^s$.

Proof. Rewriting $\widetilde{\boldsymbol{\mathcal{X}}}$ in terms of its sparse tensor approximation, followed by applying Proposition 2 and Definition 2, the proof directly follows. Note that the maximum tensor rank of $\widetilde{\boldsymbol{\mathcal{X}}}$ is d^2. □

Theorem 2 gives a bound on the rank of approximation of a tensor $\boldsymbol{\mathcal{X}}$ by $\widetilde{\boldsymbol{\mathcal{X}}}$ (which is a linear combination of atoms). Knowing rank of $\boldsymbol{\mathcal{X}}$ and $\widetilde{\boldsymbol{\mathcal{X}}}$ helps assess the quality of approximation (beyond the Euclidean norm between $\boldsymbol{\mathcal{X}}$ and $\widetilde{\boldsymbol{\mathcal{X}}}$). This is useful in experiments with imposed *Rank-R* constraints ($R = 1, 2, ...$) on atoms $\mathbf{B}_k$ and helps measure how rank constraints on $\mathbf{B}_k$ impact the approximation.

(a) (b) (c) (d) (e)

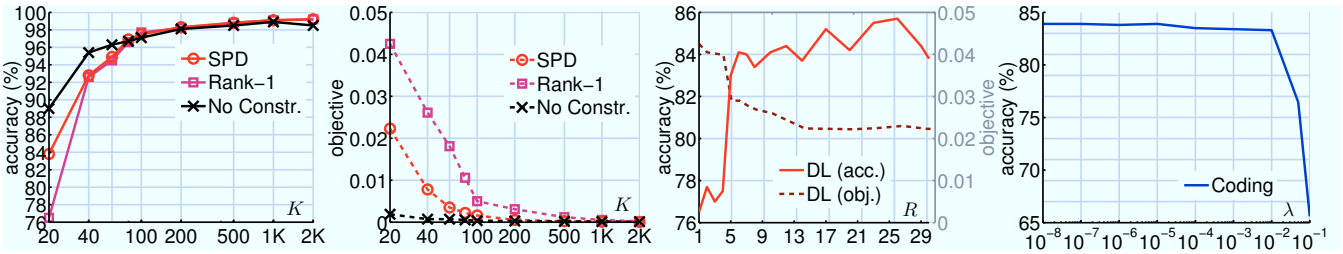
Figure 1: Examples of textures in 1(a) and 1(b) show 4 images from 4 different classes of the Brodatz dataset. Note the high interclass similarity between the top and bottom images. Figures 1(c)-1(e) show 2 samples per class per column to illustrate UIUC materials. Note high interclass variations (in contrast to Brodatz).

8. Experiments

In this section, we present experiments demonstrating the usefulness of our framework. As second-order region covariance descriptors (RCD) are the most similar tensor descriptors to our TOSST descriptor, we decided to evaluate our performance on applications of RCDs, specifically for texture recognition. In the sequel, we evaluate our novel TOSST texture descriptor and compare to the state of the art on two texture benchmarks, namely the Brodatz textures [26] and the UIUC materials [22]. Moreover, experiments illustrating behaviour of our dictionary learning are provided. We also demonstrate the adequacy of our framework to compression of third-order global image descriptors [18] on the challenging PASCAL VOC07 dataset.

8.1. Datasets

The Brodatz dataset illustrated in Figures 1(a)-1(b) contains 111 different textures each one represented by a single 640×640 image. We follow the standard protocol, *i.e.*, each texture image is subdivided into 49 overlapping blocks of size 160×160. For training, 24 randomly selected blocks are used per class, while the rest is used for testing. We use 10 data splits. The UIUC materials dataset illustrated in Figures 1(c)-1(e) contains 18 subcategories of materials taken



(a) (b) (c) (d)

Figure 2: Parameter sensitivity on Brodatz textures: 2(a) and 2(b) show classification accuracy and objective values against various dictionary sizes respectively, 2(c) and 2(d) show accuracy with fixed $K = 20$, but varying the atom rank R, and sparsity regularization λ.

in the wild from four general categories *e.g.*, bark, fabric, construction materials, and outer coat of animals. Each subcategory contains 12 images taken at various scales. We apply a leave-one-out evaluation protocol [22]. The PASCAL VOC07 set consists of 5011 images for training and 4952 for testing, 20 classes of objects of varied nature *e.g.*, human, cat, chair, train, bottle, etc.

8.2. Experimental Setup

TOSST descriptors. We evaluate the following variants of the TOSST descriptor on the Brodatz dataset: linear descriptor with first- and -second order derivatives as in (8, 9) (i) but without luminance, and ii) with luminance. RBF descriptor as in (8, 10) (iii) without luminance, and iv) with luminance. Moreover, v) is a variant based on (iv) that uses the opponent color cues for evaluations on the UIUC materials set [22]. Removing luminance for (i) and (iii) helps emphasize the benefit of RBF over the linear formulation. First though, we investigate our dictionary learning.

Dictionary learning. We evaluate several variants of our dictionary learning approach on the Brodatz dataset. We use the RBF descriptor without the luminance cue (iii) to prevent saturation in performance. We apply patch size 40 with stride 20 to further lower computational complexity, sacrificing performance slightly. This results in 49 TOSST descriptors per block. We do not apply any whitening, thus preserving the SPD structure of the TOSST slices.

Figures 2(a) and 2(b) plot the accuracy and the dictionary learning objective against an increasing size K of the dictionary. We analyze three different regularization schemes on the second-order dictionary atoms in (18), namely (1) with only SPD constraints, (2) with SPD and low-rank constraints, and (3) without SPD but $\mathrm{TRank}(\boldsymbol{\mathcal{B}}_k) = \mathrm{Rank}(\mathbf{B}_k) \leq d$. When we enforce $\mathrm{Rank}(\mathbf{B}_k) \leq R < d$ for $k = 1, ..., K$ to obtain low-rank atoms $\boldsymbol{\mathcal{B}}_k$ by scheme (2), the convexity w.r.t. $\mathbf{B}_k$ is lost because the optimization domain in this case is the non-convex boundary of the PD cone. However, the formulation (18) for schemes (1) and (3) is convex w.r.t. $\mathbf{B}_k$. The plots highlight that for lower dictionary sizes, variants (1) and (2) perform worse than (3) which exhibits the best performance due to a very low objective. However, the effects of overfitting start emerging for larger K. The *SPD* constraint appears to offer the best trade-off resulting in a good performance for both small and large K. In Figure 2(c), we further evaluate the performance for a *Rank-R* constraint given $K = 20$ atoms. The plot highlights that imposing the low-rank constraint on atoms may benefit classification. Note that for $15 \leq R \leq 30$, the classification accuracy fluctuates up to 2%, which is beyond the standard deviation error ($\leq 0.4\%$) despite any significant fluctuation to the objective in this range. This suggests that imposing some structural constraints on dictionary atoms is an important step in dictionary learning. Lastly, Figure 2(d) shows the classification accuracy for the coding and pooling steps w.r.t. λ controlling sparsity and suggests that the texture classification favors low sparsity.

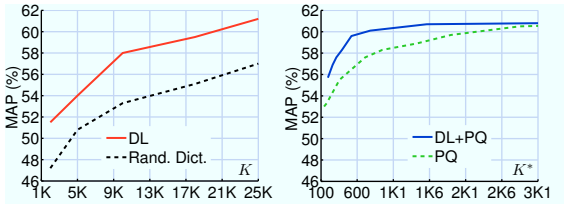
8.3. Comparison to the State of the Art on Textures

In this experiment, descriptor variants were whitened as in [18]. In each block, we extract 225 TOSST descriptors with patch size 20 and stride 10, apply our dictionary learning given $K = 2000$, followed by the sparse coding, and perform pooling as in [19] prior to classification with SVM. Table 2 demonstrates our results which highlight that the RBF variant outperforms the linear descriptor. This is even more notable when the luminance cue is deliberately removed from descriptors to degrade their expressiveness. Table 2 also demonstrates state-of-the-art results. With $99.9 \pm 0.08\%$ accuracy, our method is the strongest performer. Additionally, Table 2 provides our results on the UIUC materials obtained with descriptor variant (v) (described above) and $K = 4000$ dictionary atoms. We used TOSST descriptors

	(i) linear, no lum.	(ii) linear, lum.	(iii) RBF, no lum.	(iv) RBF, lum.	(v) RBF, lum., opp.
dataset	Brodatz	Brodatz	Brodatz	Brodatz	UIUC materials
ten. size d	6	7	30	35	45
accuracy	$93.9 \pm 0.2\%$	$99.4 \pm 0.1\%$	$99.4 \pm 0.2\%$	$\mathbf{99.9 \pm 0.08\%}$	$\mathbf{58.0 \pm 4.3\%}$[2]

Brodatz	UIUC materials
ELBCM 98.72% [31]	SD 43.5% [22]
L^2ECM 97.9% [21]	CDL $52.3 \pm 4.3\%$ [40]
RC 97.7% [36]	RSR $52.8 \pm 5.1\%$ [11]

Table 2: Evaluations of the proposed TOSST descriptor (left) and comparisons to the state of the art (right).



(a) (b)

Figure 3: Impact of K in the signature compression on PASCAL VOC07. Dictionary learning (*DL*) in 3(a) outperforms *(Random Dict.)* formed by atoms sampled from a distribution of DL. In 3(b), DL merged with Product Quantization (*DL+PQ*) is compared to Product Quantization (*PQ*) w.r.t. compressed signature size K^*.

with patch size 40, stride 20 and obtained $58.0 \pm 4.3\%$[2] which outperforms other methods. Table 3 shows the compression rates achieved with our coding prior to pooling of sparse features [19], which could not be performed directly on *e.g.*, Product Quantized codes [14].

8.4. Signature Compression on PASCAL VOC07

In this section, we evaluate our method on the PASCAL VOC07 dataset. We use the setup in [18] to generate Third-order Global Image Descriptors detailed in Section 4.2 with the goal of compressing them by our sparse coding framework. In detail, we use SIFT vectors extracted from gray-scale images with radii $12, 16, 24, 32$, and stride $4, 6, 8, 10$, reduce SIFT size to $90D$, append SPM codes of size $11D$ [18], and obtain the global image signatures with the baseline score of 61.3% MAP. Next, we learn dictionaries using our setup as in equation (18) and apply our sparse coding to reduce signature size from 176851 (upper simplex of TOSST) to sizes from $2K$ to $25K$. The goal is to regain the baseline score. Figure 3(a) shows that a learned dictionary *(DL)* of size 25K yields 61.2% MAP at $7\times$ compression rate. A random dictionary is about 4.3% worse. For completeness, Figure 3(b) shows results for Product Quantization *(PQ)* [14]. PQ compressor requires partial decompression (*e.g.*, in SGD-based SVM) unlike sparse codes. In the extreme case, it reduces the bit-rate of each dimension . It is complementary to our dictionary learning. Combined approach (*DL+PQ*) outperforms PQ by up to 3% MAP.

Complexity. On a 4-core CPU and using Matlab, our sparse coding takes about $3.2s$ using $d = 30$ and a dictionary with $2K$ atoms. Dictionary Learning converges in about 50 iterations. We have observed that about $9\times$ speedup is possible for SGD. The sparse coding sub-problem includes solving for the variables $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ alternately – which are non-negative (ℓ_1 constrained) least squares problems; each alternating iteration of which costs $\mathcal{O}(S(K^2S + K^3))$ time for K atoms and S slices. As for dictionary learning, each dictionary update takes $\mathcal{O}(KS^3)$ time.

[2]Note that we use for simplicity a single scale/size descriptor. With multiple scales/sizes, this result will improve further as UIUC materials exhibit large intraclass scale variations.

9. Conclusions

We presented a novel formulation and an efficient algorithm for sparse coding third-order tensors using a learned dictionary consisting of both first- and second-order atoms. Our experiments demonstrate that our scheme leads to significant compression of the input tensors, while not sacrificing accuracy. Further, we proposed a novel tensor descriptor for texture recognition, which when sparse-coded by our approach, achieves state-of-the-art accuracy on two benchmark datasets. Our approach is general and useful for a variety of other applications, *e.g.*, action recognition [17].

Appendix

A. Derivation of the Third-order Aggregation.

Rising K_{rbf} to the power r and applying the outer product $\uparrow\otimes_r$ of order r to $\boldsymbol{v}_{xy}$ yield a higher-order tensor of rank one denoted as $\boldsymbol{\mathcal{V}}_{xy}$:

$$K^r_{rbf}((x,y,\mathbf{I}^a),(x^{'},y^{'},\mathbf{I}^b)) \approx \left\langle \boldsymbol{\mathcal{V}}^a_{xy}, \boldsymbol{\mathcal{V}}^b_{x^{'}y^{'}} \right\rangle,\ \boldsymbol{\mathcal{V}}^i_{xy} = \uparrow\otimes_r \boldsymbol{v}^i_{xy},$$

where $i \in \{a,b\}$. Next, tensors $\boldsymbol{\mathcal{V}}_{xy}$ are aggregated over image regions $\boldsymbol{\mathcal{R}}^a$ and $\boldsymbol{\mathcal{R}}^b$ as:

$$\sum_{(x,y)\in\boldsymbol{\mathcal{R}}^a} K^r_{rbf}((x,y,\mathbf{I}^a),(x^{'},y^{'},\mathbf{I}^b)) \approx \left\langle \bar{\boldsymbol{\mathcal{V}}}^a, \bar{\boldsymbol{\mathcal{V}}}^b \right\rangle,$$

$$\text{where } \bar{\boldsymbol{\mathcal{V}}}^i = \operatorname*{Avg}_{(x,y)\in\boldsymbol{\mathcal{R}}^i} \boldsymbol{\mathcal{V}}^i_{xy}, \tag{21}$$

$$\text{and } (x^{'},y^{'}) \in \boldsymbol{\mathcal{R}}^b : x^{'} = x - x^a_0 + x^b_0 \wedge y^{'} = y - y^a_0 + y^b_0. \tag{22}$$

The aggregation step in (21) is analogous to (2-3) and step (ii) in Section 5. We shift the origin x^a_0 to x^b_0 and obtain $x^{'}$ in $\boldsymbol{\mathcal{R}}^b$ corresponding to x in $\boldsymbol{\mathcal{R}}^a$. Next, higher-order auto-correlation tensors $\bar{\boldsymbol{\mathcal{V}}}$ are formed, which are whitened using (4-7). Our experiments use $r = 3$. Using dictionary learning and sparse coding, we generate mid-level features, one per region. Given a test image and a set of overlapping regions, we perform pooling [19] over such mid-level features to obtain a descriptor that is then used in a classifier.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the ARC through the ICT Centre of Excellence program. AC is funded by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

	Brodatz		UIUC	PASCAL VOC07
d	30	35	45	101
sig. size	4960	7770	16215	176851
K	100–2000	2000	4000	25000
compr.	49.6×–2.48×	3.88×	4.05×	7.07×

Table 3: Compression rates (*sig. size* – number of unique coefficients per tensor, d – side dimension, K – vocabulary size).

References

[1] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic resonance in medicine*, 56(2):411–421, 2006. 2, 3

[2] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. *CVPR*, 2011. 2

[3] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. *ICIP*, 2005. 3

[4] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995. 5

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. *ECCV*, 2012. 1, 3, 4

[6] A. Cherian and S. Sra. Riemannian sparse coding for positive definite matrices. *ECCV*, 2014. 2, 5

[7] P. Comon. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, 2014. 2

[8] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *TIP*, 15(12):3736–3745, 2006. 2

[9] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. *ECCV*, 2014. 1

[10] K. Guo, P. Ishwar, and J. Konrad. Action recognition from video using feature covariance matrices. *TIP*, 22(6), 2013. 1

[11] M. Harandi, C. Sanderson, R. Hartley, and B. Lovell. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. *ECCV*, 2012. 2, 7

[12] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *JMLR*, 5:819–844, 2004. 4

[13] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. *CVPR*, 2009. 3

[14] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011. 8

[15] T.-K. Kim, K.-Y. K. Wong, and R. Cipolla. Tensor canonical correlation analysis for action classification. *CVPR*, 2007. 2

[16] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. 3

[17] P. Koniusz, A. Cherian, and F. Porikli. Tensor representations via kernel linearization for action recognition from 3D skeletons. *ArXiv, http://arxiv.org/abs/1604.00239*, 2016. 8

[18] P. Koniusz, F. Yan, P. Gosselin, and K. Mikolajczyk. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *PAMI*, 2016. 1, 2, 3, 4, 6, 7, 8

[19] P. Koniusz, F. Yan, and K. Mikolajczyk. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *CVIU*, 2012. 7, 8

[20] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Analysis and Applications*, 21:1253–1278, 2000. 2, 3

[21] P. Li and Q. Wang. Local log-Euclidean covariance matrix (L^2 ECM) for image representation and its applications. *ECCV*, 2012. 1, 3, 4, 7

[22] Z. Liao, J. Rock, Y. Wang, and D. Forsyth. Non-parametric filtering for geometric detail extraction and material representation. *CVPR*, 2013. 6, 7

[23] D. G. Lowe. Object recognition from local scale-invariant features. *CVPR*, 1999. 3

[24] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540–1551, 2011. 2

[25] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. *NIPS*, 2014. 2

[26] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996. 6

[27] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997. 1, 2

[28] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *IJCV*, 66(1):41–66, 2006. 2, 3

[29] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *ECCV*, 2010. 3

[30] F. Porikli and O. Tuzel. Covariance tracker. *CVPR*, 2006. 1

[31] A. Romero, M. Y. Terán, M. Gouiffès, and L. Lacassagne. Enhanced local binary covariance matrices (ELBCM) for texture analysis and object tracking. *MIRAGE*, pages 10:1–10:8, 2013. 7

[32] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. *AISTATS*, 2009. 5

[33] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. *ICML*, 2005. 2

[34] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos. Tensor sparse coding for region covariances. *ECCV*, 2010. 2

[35] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. 6

[36] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. *ECCV*, 2006. 1, 2, 3, 4, 7

[37] M. A. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *ECCV*, 2002. 2

[38] M. A. Vasilescu and D. Terzopoulos. Tensortextures: multilinear image-based rendering. *ACM Transactions on Graphics*, 23(3):336–342, 2004. 2

[39] Q. Wang, F. Chen, and W. Xu. Tracking by third-order tensor representation. *Trans. Systems, Man, and Cybernetics*, 41(2):385–396, 2011. 1

[40] R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. *CVPR*, 2012. 7

[41] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. *CVPR*, 2009. 6

[42] X. Zhao, S. Wang, S. Li, and J. Li. A comprehensive study on third order statistical features for image splicing detection. *Digital Forensics and Watermarking*, pages 243–256, 2012. 2