Convolutional Sparse and Low-Rank Coding-Based Image Decomposition

He Zhang, *Student Member, IEEE* and Vishal M. Patel, *Senior Member, IEEE*

***Abstract*—We propose novel convolutional sparse and low-rank coding-based methods for cartoon and texture decomposition. In our method, we first learn a set of generic filters that can efficiently represent cartoon and texture type images. Then using these learned filters, we propose two optimization frameworks to decompose a given image into cartoon and texture components - Convolutional Sparse Coding-Based Image Decomposition (CSCD) and Convolutional Low-rank Coding-Based Image Decomposition (CLCD). By working directly on the whole image, the proposed image separation algorithms do not need to divide the image into overlapping patches for leaning local dictionaries. The shift-invariance property is directly modeled into the objective function for learning filters. Extensive experiments show that the proposed methods perform favorably compared to state-of-the-art image separation methods.**

***Index Terms*—Image decomposition, convolutional coding, low-rank coding, sparse coding.**

I. Introduction

In many practical applications such as biomedical imaging, remote sensing, biometrics and astronomy, images can be modeled as superpositions of cartoon (e.g. piecewise smooth image) and texture structures (i.e. rain-component or fence) [1], [2], [3], [4], [5], [6]. For instance, in remote sensing, a synthetic aperture radar image can be modeled as a superposition of the ground reflectivity field (cartoon) with multiplicative speckle (texture) [7], [8], [9]. Similarly, in order to detect cracks on concrete structures, one can also model the image as a superposition of background texture with a crack component [10]. In these applications, a common task is to separate such an image into two individual images - one containing the cartoon part and the other containing the texture part.

In recent years, methods based on sparse representation and ℓ_1-minimization have been developed to deal with this problem. In particular, an approach called Morphological Component Analysis (MCA) was proposed in [11] for separating different geometrical components from a given image under the assumption that an image is the linear mixture of several morphological components. In this method, it is assumed that different morphological components are sufficiently distinct and that each one can be sparsely represented using a specific dictionary but not in the other ones. The performance of MCA depends on the dictionaries chosen for representing cartoon and texture components. In practice, dictionaries corresponding to the Discrete Cosine Transform (DCT) or the Discrete Sine Transform (DST) are used to represent the texture component as their atoms are oscillatory in nature and dictionaries corresponding to wavelet, curvelet, shearlet or contourlet are used to represent the piecewise smooth component as they represent geometric features such as edges well. The MCA algorithm has been very successful in separating various components in many practical applications [11], [12], [8], [13]. However, one of the limitations of this approach is that complicated textures found in many practical applications can not be modeled by DCT or DST dictionaries [14], [8]. As a result, it tends to produce a poor separation.

It has been observed that learning a dictionary directly from training samples rather than using a predetermined dictionary such as DCT or wavelet, usually leads to better representation and hence can provide improved results in many image processing and classification problems [15], [16]. One such dictionary learning-based method for image separation was proposed in [14]. However, this method only learns local dictionaries for the texture component and uses predetermined global dictionaries such as wavelet or curvelet for the cartoon component. One of the limitations of this method is that it is computationally very expensive and extremely slow [14]. Furthermore, most dictionary leaning approaches are patch-based and features learned with these methods often contain shifted versions of the same features [16]. To deal with this issue, Convolutional Sparse Coding (CSC) methods have been introduced in which shift invariance is directly modeled in the objective [17], [18], [19], [20]. CSC has been demonstrated to have important applications in a wide range of computer vision and image processing problems [21], [22], [23], [6].

Even though sparsity-based methods have been very successful in various image decomposition applications, the sparsity prior for texture can sometimes capture the edges in the cartoon part. To deal with this issue, many recent works have proposed using a low-rank prior to characterize the texture component [24], [25]. The idea is that the texture may have a globally different pattern, however, the matrix composed by suitable stacking of vectorized blocks is low-rank. Using this idea, [24] proposed a novel texture prior called Block Nuclear Norm (BNN) for image separation. Similarly, a Low Patch Rank (LPR) prior was enforced on the patches extracted from a texture component in [25] for separating an image into cartoon and texture components. However, these low-rank based methods assume that the underlying texture lies in a single low-rank subspace, which may not be true in practice since a texture often lies in a union of multiple subspaces.

He Zhang is with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, 08854 USA e-mail:he.zhang92@rutgers.edu

Vishal Patel is with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, 08854 USA e-mail:vishal.m.patel@rutgers.edu

Manuscript received 2016.

Besides sparsity and patch-rank methods, edge-preserving techniques can also be used for image separation, as the task of edge-preserving methods can be regarded as decomposing a given image into a natural image with sharp edges (cartoon) and the noise or texture component [26]. Different types of affinity relations have been explored to preserve the sharp edges [27], [26], [28]. Even though the edge-preserving methods achieve very good performance in recovering the cartoon-type images, they often fail to distinguish textures from noise [27], [26], [28]. Most recently, deep learning methods have also been adopted to address the singe image separation problem via directly learn the mapping between input image and desired target image [29], [30].

In contrast to these methods, we propose a global Convolutional Low-rank Coding (CLC) method to model the texture component by learning a set of low-rank convolutional filters directly from the training texture images. Using the CSC and CLC methods for representing cartoon and texture components, respectively, we propose two image separation methods - *Convolutional Sparse Coding-based Decomposition* (CSCD) and *Convolutional Low-rank Coding-based Decomposition* (CLCD) for *Cartoon + Texture* decomposition. Figure 1 gives an overview of the proposed image separation methods. We first learn a set of generic filters that can efficiently represent cartoon and texture type images. Then using these learned filters, we propose an optimization framework to separate a given image into cartoon and texture components. Rather than using TV-regularization as post-processing procedure to refine the recovered cartoon component discussed in [31], [11], we propose a new optimization procedure to directly include the TV-regularization into the optimization framework.

This paper makes the following contributions.

1) CSCD is proposed in which using multiple cartoon and texture training images, we first learn sparsity-based convolutional filters corresponding to these components. Then, using the learned filters, we develop an MCA type of algorithm to separate the texture and cartoon components from a given image.
2) Similar to CSC, we propose CLC to efficiently represent low-rank textures.
3) CLCD is proposed in which we use multiple texture images to learn a set of convolutional low-rank filters based on CLC and use multiple cartoon images to learn a set of sparsity-based convolutional filters. Then, we develop a modified MCA type of algorithm to separate the texture and cartoon components from a given image by replacing the sparsity prior on texture with low-rank prior.

A preliminary version of this work appeared in [31], which describes just the CSCD method for image separation. Extensive experimental evaluations, CLC and CLCD are extensions to [31].

The rest of the paper is organized as follows. In Section II, we give a brief background on sparsity-based image separation and convolutional sparse coding. Details of the proposed CSCD algorithm are given in Section III. The CLC and CLCD algorithms are introduced in Section IV. Experimental results are presented in Section V and Section VI concludes the paper with a brief summary and discussion.

II. Background

In this section, we give a brief background on sparsity-based image separation and convolutional sparse coding.

A. Image Separation

Let $\mathbf{y}$ be a lexicographically ordered vector of size N^2 corresponding to an image $Y \in \mathbb{R}^{N\times N}$. Assume that $\mathbf{y}$ is a superposition of two different images

$$\mathbf{y} = \mathbf{y}_c + \mathbf{y}_t, \tag{1}$$

where $\mathbf{y}_c$ and $\mathbf{y}_t$ are the cartoon or piecewise smooth component and the texture component of $\mathbf{y}$, respectively. We assume that $\mathbf{y}_c$ is sparse in a dictionary represented in a matrix form as $\mathbf{D}_c \in \mathbb{R}^{N^2\times M_c}$, and similarly, $\mathbf{y}_t$ is sparse in a dictionary represented in a matrix form as $\mathbf{D}_t \in \mathbb{R}^{N^2\times M_t}$. The dictionaries $\mathbf{D}_c$ and $\mathbf{D}_t$ are chosen such that they provide sparse representations of piecewise smooth and texture components, respectively. That is, we assume there are coefficient vectors $\mathbf{x}_c \in \mathbb{R}^{M_c}$ and $\mathbf{x}_t \in \mathbb{R}^{M_t}$ so that $\mathbf{y}_c = \mathbf{D}_c\mathbf{x}_c$ and $\mathbf{y}_t = \mathbf{D}_t\mathbf{x}_t$. The sparsity assumption means that when the coefficients are ordered in magnitude, they decay rapidly. Then, one can estimate the components $\mathbf{y}_c$ and $\mathbf{y}_t$ via $\mathbf{x}_c$ and $\mathbf{x}_t$ by solving the following optimization problem [11]

$$\begin{aligned}\hat{\mathbf{x}_c}, \hat{\mathbf{x}_t} = \arg\min_{\mathbf{x}_c,\mathbf{x}_t} &\frac{1}{2}\|\mathbf{y} - \mathbf{D}_c\mathbf{x}_c - \mathbf{D}_t\mathbf{x}_t\|_2^2 + \lambda_c\|\mathbf{x}_c\|_1 \\ &+ \lambda_t\|\mathbf{x}_t\|_1 + \lambda \mathrm{TV}(\mathbf{D}_c\mathbf{x}_c),\end{aligned} \tag{2}$$

where TV is the total variation (i.e. sum of the absolute variations in the image) [32] and for an N-dimensional vector $\mathbf{x}$, $\|\cdot\|_q$ denotes the ℓ_q-norm, $1 \le q < \infty$, defined as $\|\mathbf{x}\|_q = \left(\sum_{i=1}^N |x_i|^q\right)^{\frac{1}{q}}$. Here, λ_c, λ_t and λ are positive regularization parameters. The two components are the corresponding representations of the two parts and can be obtained by $\hat{\mathbf{y}_c} = \mathbf{D}_c\hat{\mathbf{x}_c}$ and $\hat{\mathbf{y}_t} = \mathbf{D}_t\hat{\mathbf{x}_t}$. Various methods have been developed in the literature to obtain the solution of (2) [11], [12].

B. Convolutional Sparse Coding

In CSC, given a set of M training samples $\{\mathbf{y}_m\}_{m=1}^M$, the objective is to learn a set of convolutional filters $\{\mathbf{d}_k\}_{i=1}^K$ by solving the following optimization problem

$$\begin{aligned}\arg\min_{\mathbf{d},\mathbf{x}} \frac{1}{2}\sum_{m=1}^M \left\|\mathbf{y}_m - \sum_{k=1}^K \mathbf{d}_k * \mathbf{x}_{m,k}\right\|_2^2 + \lambda \sum_{m=1}^M\sum_{k=1}^K \|\mathbf{x}_{m,k}\|_1 \\ \text{subject to } \|\mathbf{d}_k\|_2^2 \le 1 \quad \forall k \in \{1,\cdots,K\},\end{aligned} \tag{3}$$

where $\mathbf{x}_{m,k}$ are the sparse coefficients that approximate the data $\mathbf{y}_m$ when convolved with the corresponding filters $\mathbf{d}_k$ of fixed support. Here, $*$ represents the 2-D convolution operator and λ is a positive regularization parameter. Several methods have been proposed in the literature for solving the above optimization problem. For instance, [18] introduced a Fourier domain Alternating Direction Method of Multipliers

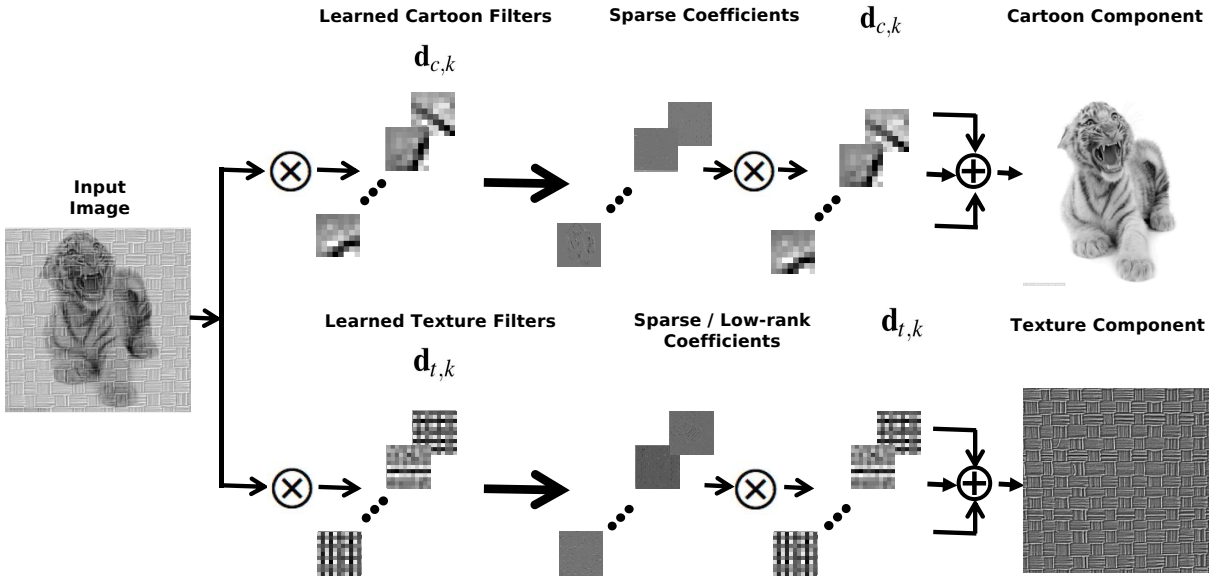


Fig. 1: An overview of the testing process of the proposed convolutional sparse and low-rank coding-based image decomposition methods. The cartoon filters $\mathbf{d}_{c,k}$ have already been learned from a set of clean cartoon images and the texture filters $\mathbf{d}_{t,k}$ have already been learned from a set of clean texture images.

(ADMM) [33] framework for solving the CSC problem. In [19] proper boundary conditions were incorporated for solving the CSC optimization problem. More recently, [20], [34], [35], [36] developed an efficient method that jointly uses the space and Fourier domains to solve the CSC problem. Most recently, [37] proposed a new convolutional sparse coding algorithm that incorporates the boundary truncation operator whose algorithm is more friendly to training with big-data. In this paper, we adapt the method proposed in [20] for learning the convolutional filters due to its simplicity and efficiency.

III. Convolutional Sparse Coding-Based Image Decomposition (CSCD)

Following the mixture model in (1), given $\mathbf{y}$ our goal is to separate it into $\mathbf{y}_c$ and $\mathbf{y}_t$. Assume that we have already learned the convolutional sparsity-based filters corresponding to $\mathbf{y}_c$ and $\mathbf{y}_t$ by solving the CSC problem (3) for the cartoon and the texture components separately. That is, we have learned $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$ and $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$ such that $\mathbf{y}_c = \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k}$ and $\mathbf{y}_t = \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k}$, where $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$ are the sparse coefficients that approximate $\mathbf{y}_c$ and $\mathbf{y}_t$ when convolved with the filters $\mathbf{d}_{c,k}$ and $\mathbf{d}_{t,k}$, respectively. We propose to estimate $\mathbf{y}_c$ and $\mathbf{y}_t$ via $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$ by solving the following CSC-based optimization problem

$$\begin{aligned}\hat{\mathbf{x}}_{c,k}, \hat{\mathbf{x}}_{t,k} = \arg \min_{\mathbf{x}_{c,k}, \mathbf{x}_{t,k}} & \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} \right\|_2^2 \\ & + \lambda_c \sum_{k=1}^{K_c} \|\mathbf{x}_{c,k}\|_1 + \lambda_t \sum_{k=1}^{K_t} \|\mathbf{x}_{t,k}\|_1 \\ & + \lambda \mathrm{TV} \left(\sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right),\end{aligned} \tag{4}$$

where λ_t, λ_c and λ are positive regularization parameters. In this paper, we adopt the anisotropic version of the TV-regularization

$$\mathrm{TV}(\mathbf{x}) = \|\mathbf{g_0} * \mathbf{x}\|_1 + \|\mathbf{g_1}, *\mathbf{x}\|_1, \tag{5}$$

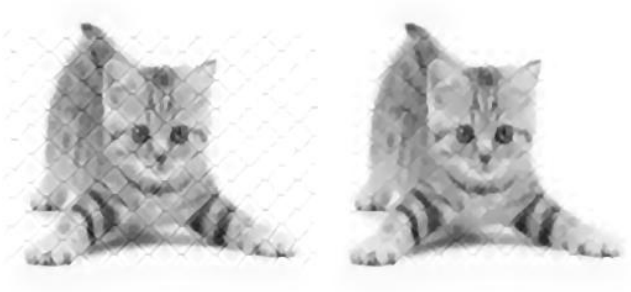
where $\mathbf{g}_0$ and $\mathbf{g}_1$ are filters that compute the image gradients along the rows and columns, respectively. Once, $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$ are estimated, the two components can be obtained by $\hat{\mathbf{y}}_c = \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \hat{\mathbf{x}}_{c,k}$ and $\hat{\mathbf{y}}_t = \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \hat{\mathbf{x}}_{t,k}$.

A. Optimization

The optimization problem can be solved iteratively over $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$.

1) Update step for $\mathbf{x}_{c,k}$: In this step, we assume that $\mathbf{x}_{t,k}$ is fixed. As a result, the following problem needs to be solved

$$\begin{aligned}\hat{\mathbf{x}}_{c,k} = \arg \min_{\mathbf{x}_{c,k}} & \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right\|_2^2 \\ & + \lambda_c \sum_{k=1}^{K_c} \|\mathbf{x}_{c,k}\|_1 + \lambda \mathrm{TV} \left(\sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right)\end{aligned} \tag{6}$$



(a) W/O TV-regularization (b) With TV-regularization

Fig. 2: Sample results of the proposed CSCD methods with and without the TV-regularization.

Since, $\mathbf{x}_{t,k}, \mathbf{d}_{t,k}$ and $\mathbf{d}_{c,k}$ are fixed, (6) is essentially a sparse coding problem with the TV-regularization, which can be solved using the DFT-based ADMM algorithm [35]. Optimization details are given in the Appendix A.

Note that the TV-regularization is used to preserve important details such as edges in the cartoon component and to remove some undesirable artifacts. Figure 2 gives an example of the cartoon image obtained using the proposed CSCD method with and without the TV-regularization. As can be seen from Figure 2 (a), when CSCD is used without the TV-regularization, we observe the presence of some unwanted artifacts. From Figure 2 (b) we see that these artifacts are removed when the TV-regularization is included into the optimization and we obtain much better cartoon component.

2) Update step for $\mathbf{x}_{t,k}$: For a fixed $\mathbf{x}_{c,k}$, we have to solve the following problem to obtain $\mathbf{x}_{t,k}$

$$\hat{\mathbf{x}}_{t,k} = \arg\min_{\mathbf{x}_{t,k}} \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} \right\|_2^2 + \lambda_t \sum_{k=1}^{K_t} \|\mathbf{x}_{t,k}\|_1 . \quad (7)$$

Again, this problem can be solved using the ADMM framework proposed in [34].

The overall CSCD algorithm is summarized in the Algorithm 1. Here, λ_c, λ_t, and λ are regularization parameters, L is the total iteration number, $\mathbf{y}$ is the input image to be separated and $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}_t$ are the estimated cartoon component and the texture component, respectively.

Algorithm 1: The CSCD Algorithm for *Cartoon+Texture* Image Decomposition.

Input: $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$, $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$, $\mathbf{y}$, λ_c, λ_t, λ, L
  for $i = 1 : L$
    Obtain $\hat{\mathbf{x}}_{c,k}$ by solving (6).
    Obtain $\hat{\mathbf{x}}_{t,k}$ by solving (7).
  end for
  $\hat{\mathbf{y}}_c = \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \hat{\mathbf{x}}_{c,k}$
  $\hat{\mathbf{y}}_t = \sum_{k=1}^{K_{l,t}} \mathbf{d}_{t,k} * \hat{\mathbf{x}}_{t,k}$
Output: $\hat{\mathbf{y}}_c$, $\hat{\mathbf{y}}_t$

IV. Convolutional Low-rank Coding-Based Image Decomposition (CLCD)

Since textures inherently have low-rank characteristics, one can learn a better representation for them by promoting a low-rank property on the coefficients rather than a sparsity property as is normally done in CSC. The proposed framework for CLC is very similar to CSC, which involves solving the following optimization problem

$$\arg\min_{\mathbf{d}_k, \mathbf{x}_{m,k}} \frac{1}{2} \sum_{m=1}^{M} \left\| \mathbf{y}_m - \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{x}_{m,k} \right\|_2^2 + \lambda \sum_{m=1}^{M} \sum_{k=1}^{K} \|\mathbf{x}_{m,k}\|_* \quad \text{subject to } \|\mathbf{d}_k\|_2^2 \leq 1 \quad \forall k \in \{1, \cdots, K\}, \quad (8)$$

where $\mathbf{x}_{m,k}$ are the low-rank coefficients that approximate the data $\mathbf{y}_m$ when convolved with the corresponding filters $\mathbf{d}_k$ of fixed support and $\|\cdot\|_*$ is the nuclear norm, which is the sum of the singular values. Optimization problem (8) can be solved iteratively by updating $\mathbf{d}_k$ while fixing $\mathbf{x}_{m,k}$ and then updating $\mathbf{x}_{m,k}$ while fixing $\mathbf{d}_k$ as it is a bi-convex problem. Optimization details are given in the Appendix B.

Assume that we have learned a set of convolutional sparsity-based filters $\{\mathbf{d}_{c,k}\}$ using CSC to sparsely represent the cartoon part and another set of convolutional low-rank based filters $\{\mathbf{d}_{t,k}\}$ using CLC to efficiently represent the texture component. That is, we have learned $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$ and $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$ such that $\mathbf{y}_c = \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k}$ and $\mathbf{y}_t = \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k}$, where $\mathbf{x}_{c,k}$ are the sparse coefficients and $\mathbf{x}_{t,k}$ are the low-rank coefficients that approximate $\mathbf{y}_c$ and $\mathbf{y}_t$ when convolved with the filters $\mathbf{d}_{c,k}$ and $\mathbf{d}_{t,k}$, respectively. Then, we propose to estimate the cartoon and texture components via $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$, respectively by solving the following CLCD optimization problem

$$\begin{aligned} \hat{\mathbf{x}}_{c,k}, \hat{\mathbf{x}}_{t,k} = \arg\min_{\mathbf{x}_{c,k}, \mathbf{x}_{t,k}} & \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} \right\|_2^2 \\ & + \lambda_c \sum_{k=1}^{K_c} \|\mathbf{x}_{c,k}\|_1 + \lambda_t \sum_{k=1}^{K_t} \|\mathbf{x}_{t,k}\|_* \\ & + \lambda \mathrm{TV} \left(\sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right), \end{aligned} \quad (9)$$

where λ_t, λ_c, and λ are positive regularization parameters. Note that in CLCD, we promote sparsity property on the coefficients corresponding to the cartoon part and low-rank property on the coefficients corresponding to the texture part. This is in contrast to CSCD where coefficients corresponding to both cartoon and texture components are required to be sparse.

A. Optimization

The resulting optimization problem can be solved iteratively over $\mathbf{x}_{c,k}$ and $\mathbf{x}_{t,k}$.

Algorithm 2: The CLCD Algorithm for *Cartoon+Texture* Image Decomposition.

Input: $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$, $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$, $\mathbf{y}$, λ_c, λ_t, λ, L
  for $i = 1 : L$
    Obtain $\hat{\mathbf{x}}_{c,k}$ by solving (6).
    Obtain $\hat{\mathbf{x}}_{t,k}$ by solving (11).
  end for
  $\hat{\mathbf{y}}_c = \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \hat{\mathbf{x}}_{c,k}$
  $\hat{\mathbf{y}}_t = \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \hat{\mathbf{x}}_{t,k}$
Output: $\hat{\mathbf{y}}_c$, $\hat{\mathbf{y}}_t$

1) Update step for $\mathbf{x}_{c,k}$*:* When $\mathbf{x}_{t,k}$ is fixed, we need to solve the following problem to obtain the sparse coefficients $\mathbf{x}_{c,k}$

$$\hat{\mathbf{x}}_{c,k} = \arg\min_{\mathbf{x}_{c,k}} \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right\|_2^2 + \lambda_c \sum_{k=1}^{K_c} \|\mathbf{x}_{c,k}\|_1 + \lambda \mathrm{TV} \left(\sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} \right). \tag{10}$$

This problem is exactly the same as (6) and can be solved using the DFT-based ADMM method described in the Appendix A.

2) Update step for $\mathbf{x}_{t,k}$*:* For a fixed $\mathbf{x}_{c,k}$, we have to solve the following problem to obtain $\mathbf{x}_{t,k}$

$$\hat{\mathbf{x}}_{t,k} = \arg\min_{\mathbf{x}_{t,k}} \frac{1}{2} \left\| \mathbf{y} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} \right\|_2^2 + \lambda_t \sum_{k=1}^{K_t} \|\mathbf{x}_{t,k}\|_* . \tag{11}$$

This problem is very similar to the sub-problem that we solve in CLC for finding the low-rank coefficients when $\mathbf{d}_k$ are fixed. Let $\mathbf{y}_r = \mathbf{y} - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k}$. Then (11) can be rewritten as

$$\hat{\mathbf{x}}_{t,k} = \arg\min_{\mathbf{x}_{t,k}} \frac{1}{2} \left\| \mathbf{y}_r - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k} \right\|_2^2 + \lambda_t \sum_{k=1}^{K_t} \|\mathbf{x}_{t,k}\|_* ,$$

which can be solved using the optimization procedure described in Appendix B-2.

The overall CLCD algorithm for *Cartoon+Texture* separation is summarized in Algorithm 2.

V. Experimental Results

In this section, we present the results of our proposed image separation algorithms and compare them with the sparsity-based MCA method [11], adaptive dictionary learning-based MCA (A-MCA) method [14] and a recent state-of-the-art low-rank-based Block Nuclear Norm (BNN) method [24]. We also compare our methods with two edge-preserving techniques: image guided filtering (GF) based method [38] and recently introduced rolling guidance filter (RGF) [28]. Furthermore, we compare the performance of different methods with a CLCD method where instead of learning low-rank filters for the texture component, we learn sparsity-based filters using CSC. We call this method S-CLCD. This will clearly show the significance of learning texture filters by CLC. In these experiments, we use Peak Signal to Noise Ratio (PSNR) to measure the performance of the routines tested. For the MCA method, curvelet and local DCT dictionaries are used to represent the cartoon and the texture components, respectively. For the A-MCA method, we use a curvelet dictionary for sparsely representing the cartoon component and learn a set of local patch-based texture dictionaries using the images shown in Figure 5 (a) to represent the texture component. Following the common practice in CSC [20], the input image is first high-pass and low-pass filtered. The CSCD and CLCD methods are applied only on the high-pass filtered image since convolutional sparse representations do not provide a good representation of the low-frequency components of an image. The low-pass filtered image is considered as part of the cartoon component. During training, 100 cartoon images and 50 texture images are selected to learn the cartoon and texture filters, respectively. Experiments were conducted using Python on an Ubuntu 14.04 system with Intel Xeon(R) CPU E5-2623 v3 3.00GHz processor.

All the parameters corresponding to the proposed methods are empirically determined, as tabulated in Table I, where i indicates the index of iteration. In all the experiments, the total number of outer (L) and inner iterations are set equal to 10 and 50, respectively. The parameters for the RGF method [28] were also fine-tuned for different experiments. They are chosen as follows: we set spatial variance $\sigma_s = 1.5$ and range variances $\sigma_r = 0.18$ for the first three experiments, $\sigma_s = 1.5$ and $\sigma_r = 0.15$ for the fourth experiment (Barbara Image) and $\sigma_s = 2.3$ and $\sigma_r = 0.2$ for the last experiment. In our experiments, we found that the anisotropic TV-norm performs slightly better than the isotropic TV-norm. Hence, we use the anisotropic TV-norm based optimization for image separation in this paper.

A. Cartoon + Texture Image Separation

Some training images shown in Figure 5 (a) and Figure 5 (b) are used to learn the convolution sparse filters $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$ and $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$, respectively using the CSC method proposed in [20]. The corresponding learned filters are shown in Figure 5 (c) and Figure 5 (d) for the texture and the cartoon components, respectively. The convolution low-rank filters $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$ learned by CLC using the textures in Figure 5 (a), are shown in Figure 5 (e). The size of each filter is set equal to 11×11 for all three methods.

Figures 3, 4 and 6 show the original images and the decomposed images corresponding to different methods. All the test images were excluded from the images used to learn the convolutional filters. In all three figures, the first column shows the original test image, original cartoon image and original texture image. Second, third and fourth columns show the results corresponding to our CSCD, S-CLCD and CLCD methods, respectively. Fifth, sixth, seventh, eighth and ninth columns show the results corresponding to the BNN [24],

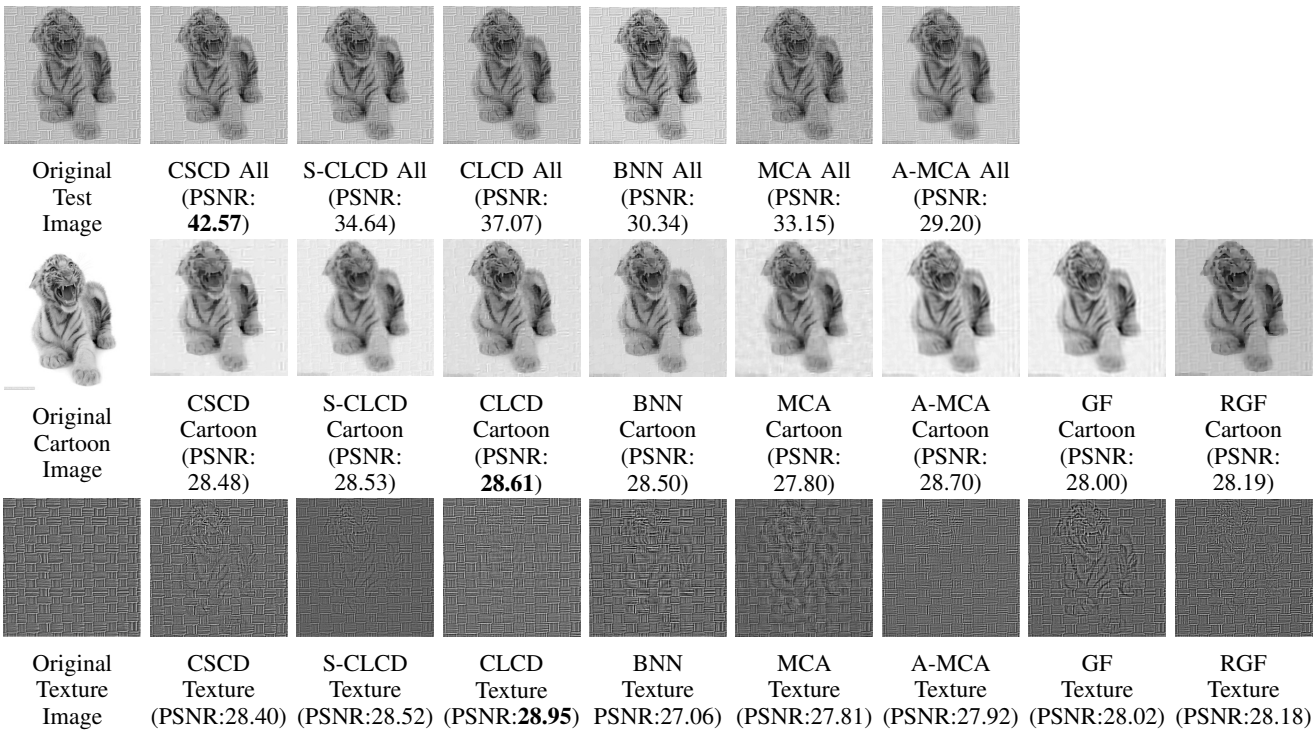


Fig. 3: Image decomposition results on the *Tiger+Texture* image. We compare the performance of our three methods with that of BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28].

TABLE I: Parameters used in our experiemnts(i represents the iteration).

	CSCD	S-CLCD	CLCD
First Four Experiments	$\lambda_c = \max(0.55 - 0.07 * i, 0.003)$; $\lambda_t = \max(0.5 - 0.07 * i, 0.003)$; $\lambda = 0.05$;	$\lambda_c = \max(0.47 - 0.063 * i, 0.002)$; $\lambda_t = \max(5.08 - 0.72 * i, 0.38)$; $\lambda = 0.05$;	$\lambda_c = \max(0.53 - 0.07 * i, 0.003)$; $\lambda_t = \max(4.80 - 0.65 * i, 0.35)$; $\lambda = 0.05$;
Fingerprints	$\lambda_c = \max(0.55 - 0.05 * i, 0.15)$; $\lambda_t = \max(0.5 - 0.05 * i, 0.05)$; $\lambda = 0.03$;	$\lambda_c = \max(0.45 - 0.06 * i, 0.12)$; $\lambda_t = \max(5.00 - 0.65 * i, 0.65)$; $\lambda = 0.03$;	$\lambda_c = \max(0.45 - 0.06 * i, 0.12)$; $\lambda_t = \max(5.00 - 0.75 * i, 0.60)$; $\lambda = 0.03$;

MCA [11], A-MCA [14], GF [38] and RGF [28] methods, respectively.[1]

As can be seen from these figures, our methods are able to separate the morphological components from the given images better than the other methods. In particular, experiments with the *Tiger+Texture* image shown in Figure 3 show that CSCD, S-CLCD and CLCD methods achieve better PSNR results of 28.48 dB, 28.53 dB and 28.61 dB on the separated cartoon component compared to the PSNRs of 28.50 dB, 27.80 dB, 28.70 dB, 28.00 dB and 28.19 dB corresponding to BNN, MCA, A-MCA, GF, RGF methods, respectively. Similarly, CSCD, S-CLCD and CLCD obtain the PSNR of 28.40 dB, 28.52 dB and 28.95 dB for the texture component compared to the PSNRs of 27.06 dB, 27.81 dB, 27.92 dB. 28.02 dB and 28.18 dB corresponding to BNN, MCA, A-MCA, GF and RGF methods, respectively. Overall, our method achieves the PSNR of 42.57 dB, 34.64 dB and 37.07 dB when the two estimated components are combined compared to the PSNRs of 30.34 dB, 33.15 dB and 29.20 dB for the BNN, MCA and A-MCA methods, respectively. Similar PSNR performances are also observed in Figure 4 and 6 with the experiments on the *Cat+Cage* image and *Boy+Mixed Texture* image, respectively. These results clearly indicate that an improvement is achieved when CSC and CLC methods are used to separate morphological components of an image as can be seen by comparing the visual as well as PSNR results of our method with that of MCA, A-MCA BNN, GF and RGF in Figures 3, 4 and 6.

To better demonstrate the effectiveness of the proposed methods, more quantitative results are evaluated on another twenty synthetic images. The average results evaluated on the recovered cartoon and texture components are tabulated in Table II. According to the quantitative performance, it can be observed that our proposed CLCD method achieves the best performance on average compared to the other methods. In particular, CLCD achieves much better results on the recovered texture components. This demonstrates the effectiveness of using convolutional low-rank coding to represent the texture

[1]As there is no specific prior in the GF [38] and RGF [28] methods in charactering texture component, we regard the subtraction between the input image and the recovered cartoon image as the recovered texture part. Therefore, we are unable to evaluate how well these two methods perform overall on the addition of the two components. (the PSNR measured on the addition of the recovered cartoon and the recovered texture part compared with the original test image).

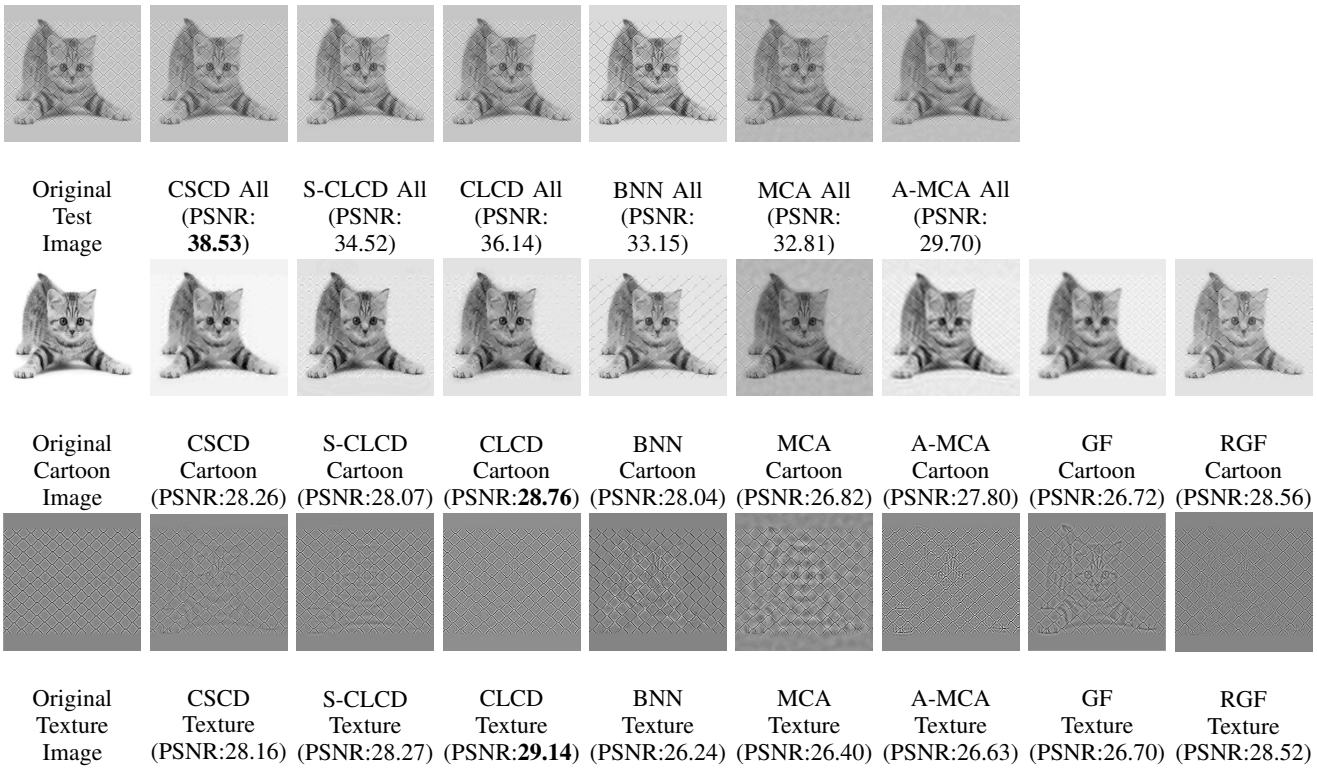


Fig. 4: Image decomposition results on the *Cat+Cage* image. We compare the performance of our three methods with that of BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28].

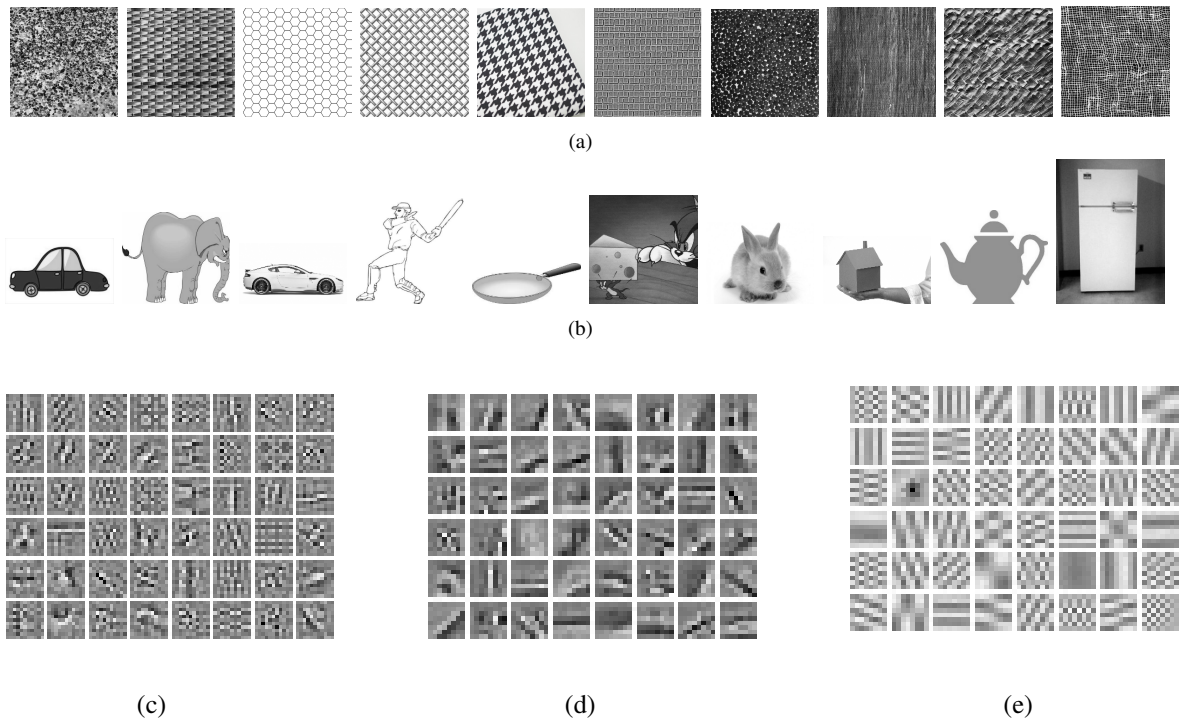


Fig. 5: (a) Training texture images used for learning a set of texture filters $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$. (b) Training cartoon images used for learning a set of cartoon filters $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$. (c) Learned sparsity-based texture filters $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$. (d) Learned cartoon filters $\{\mathbf{d}_{c,k}\}_{k=1}^{K_c}$. (e) Learned low-rank based texture filters $\{\mathbf{d}_{t,k}\}_{k=1}^{K_t}$.

components in our application.

It is interesting to note that CLCD recovered the texture component better than the other methods, demonstrating its effectiveness in characterizing the texture component using a

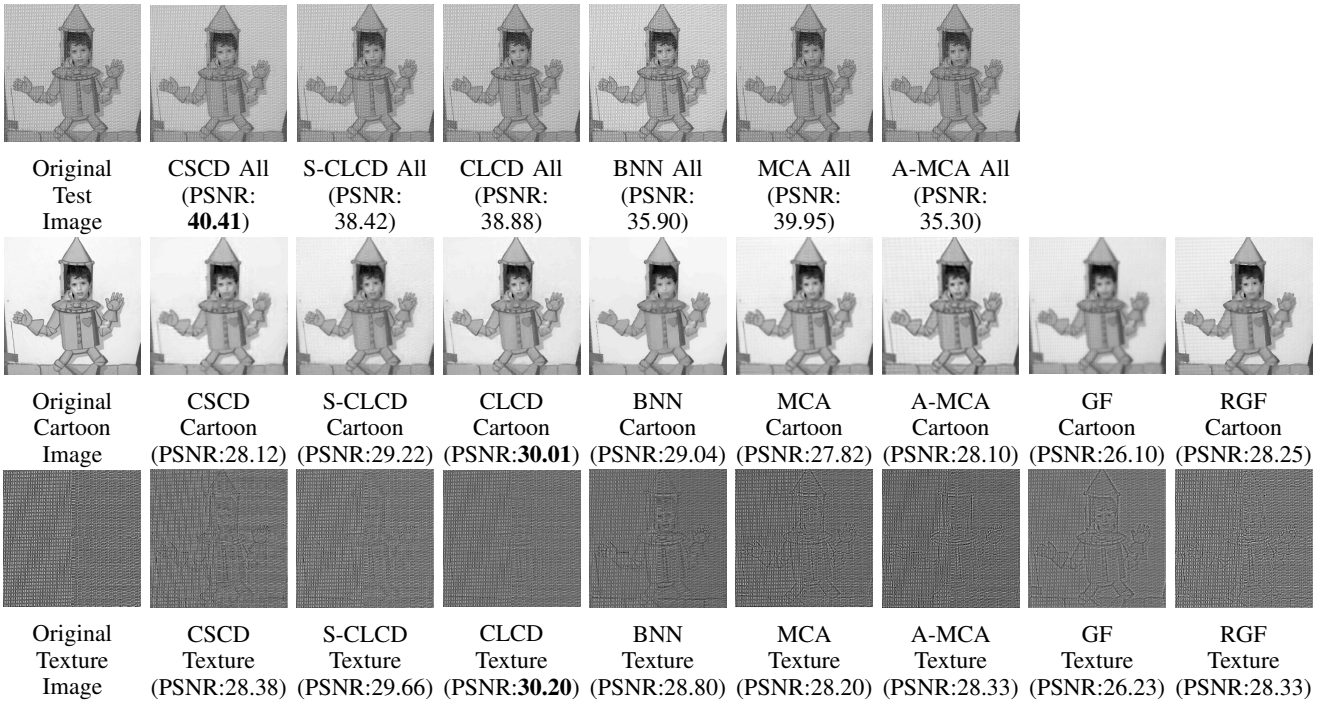


Fig. 6: Image decomposition results on the *Boy + Mixed texture* image. We compare the performance of our three methods with that of BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28].

TABLE II: Average quantitative results evaluated on twenty synthetic images.

PSNR	MCA [11]	A-MCA [14]	BNN [24]	GF [38]	RGF [28]	CSCD	S-CSCD	CLCD
Cartoon (dB)	26.88	27.62	27.49	26.83	27.45	27.55	28.02	**28.15**
Texture (dB)	26.56	27.58	26.89	26.92	27.52	27.28	27.89	**28.80**

low-rank prior. Furthermore, the recovered texture component in Figure 6 indicates that the proposed CLCD method can deal with heterogeneous textures better compared to BNN. In general better results are obtained by our methods than previous patch-based methods, indicting the advantage of using convolutional coding for image separation.

Using a set of twenty synthetic images (including the three examples we use in this paper), we show the average evolution of the CSCD and CLCD objective functions in Figure 7. Note that in Figure 7, the relative error decreases significantly after a few iterations and saturates around the ninth iteration, showing that the proposed method is efficient and requires a few number of iterations to converge.

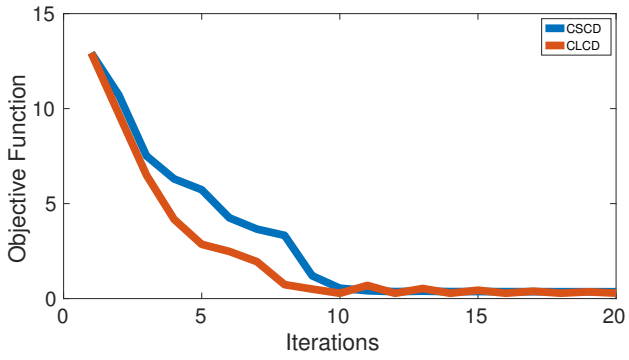


Fig. 7: The objective function value as a function of iteration number for the experiments conducted on twenty synthetic images.

B. Real Image Separation

We also evaluate different methods on the Barbara image. The decomposition results as shown in Fig 8, where the recovered cartoon part, recovered texture part and close-up of the recovered cartoon part are shown in the first, second and third row, respectively. As can be seen from this figure, the proposed methods achieve similar performance compared to the state-of-art BNN method in the recovered texture part. Furthermore, from the third row, we can see that our methods can capture less texture component in the recovered cartoon part, which clearly demonstrates the effectiveness of the proposed methods in separating real-world images.

C. Fingerprint Extraction

In the last set of experiments, we present an application of the proposed CSCD and CLCD methods in extracting the underlying fingerprint from a latent fingerprint. Latent fingerprints are among the most valuable and common types of physical evidence. Latent fingerprints obtained from crime scenes can be used as crucial evidence in forensic identification. However, matching latent fingerprints with the enrolled fingerprints is a difficult problem as latent fingerprints are

often of poor quality. In this experiment, we show that one can use the proposed CSCD, S-CLCD and CLCD methods to extract the underlying fingerprint from a latent fingerprint, which can be then matched with the enrolled fingerprints. In this experiment, we only present the separation results as the matching of latent fingerprints is beyond the scope of this paper. We use the same learned cartoon filters as used in the previous experiments. However, we learn the texture sparsity based and low-rank-based filters corresponding to fingerprints, from a set of clean fingerprints. For the A-MCA method, we use a curvelet dictionary for sparsely representing the cartoon component and learn a local patch-based dictionary using the same set of clean clean fingerprints. The parameters used in this experiment are tabulated in Table I.

The first row of Figure 9 shows the input latent fingerprint and the second row presents the learned sparsity-based texture (fingerprint) filters and a set of learned low-rank-based texture (fingerprint) filters. The learned fingerprint filters show some characteristics unique to fingerprints. As can be seen from the binarized close-ups of delta and whorl shown in the first column in Figure 9, fingerprints are difficult to analyze due to the presence of black marks. In the third row, we show the extracted fingerprints by the CSCD, S-CLCD and CLCD methods as well as BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28] methods. We can observe from this figure that our methods are able to extract the underlying structure of the fingerprint better than MCA, A-MCA, GF and RGF. This can be seen by comparing the binarized extracted delta and whorl patterns in the last two rows of this figure. These two features are close-ups of certain regions in recovered fingerprints component. It is interesting to see that if the cartoon part is piecewise smooth, then our method and BNN can recover the shape of the fingerprint better than MCA since it uses local DCT to represent the texture component. Furthermore, as can be seen from the results of A-MCA, learning a local dictionary to represent the fingerprint textures does not produce good results. This experiment clearly shows the significance of our proposed convolutional coding-based methods compared to MCA and A-MCA that use fixed and patch-based adaptive dictionaries, respectively.

D. Analysis of Algorithms with Noisy Test Data

We evaluate the performance of the proposed image separation methods in the presence of additive white Gaussian noise. Experiments are conducted by corrupting the *Tiger+Texture*, *Cat+Cage* and *Boy + Mixed Texture* images by Gaussian noise with varying standard deviations. In these experiments, we empirically determined the parameters λ_c and λ_t for different noise levels. We also fine-tune the parameter for other methods in the presence of gaussian noise with different variance. Results are tabulated in Tables III, IV and V.

As can be seen from these tables, our methods achieve the best PSNR results in most of the cases, which demonstrate their robustness in the presence of noise compared with other methods. Meanwhile, we can observe that though edge-preserving methods [38], [28] achieve comparable performance in recovering the cartoon component, they are unable to separate texture from noise.

E. Boundary Artifacts

Since our methods are solved in the DFT domain, they implicitly impose periodic boundary conditions, which may result in artifacts around the boundary when the input images are not circularly symmetric on the boundary. One possible solution is to involve the boundary spatial mask to mask out the boundaries of the padded estimation as discussed in [36]. However, the spatial mask does not have a compact representation in the DFT domain. In our future work, we will explore different ways to handle boundary artifacts within our optimization framework.

F. Computational Complexity

In this section, we analyze the computational complexity of the proposed image separation algorithms. Suppose we are given an $N \times N$ test image, $\mathbf{y}_t$ and we have learned a set of $\{\mathbf{d}_k\}_{k=1}^K$ convolutional filters using CSC or CLC. Then, the computational complexity of the CSCD algorithm is as follows [31]: The complexity of solving (17) is $O(KN^2)$. The complexity in transferring (16) in the DFT domain is $O(KN^2 \log N^2)$ and the complexity of solving (16) in the DFT domain is $O(KN^2)$. As a result, the overall complexity of the CSCD algorithm is $O(KN^2 \log N^2)$.

Similar analysis is also applied to the CLCD algorithm: The complexity of solving (29) is $O(KN^3)$, which is the most computationally heavy part in CLCD. The complexity in transferring (28) in the DFT domain is $O(KN^2 \log N^2)$ and the complexity in solving (28) in the DFT domain is $O(KN^2)$. Therefore, the overall complexity of the CLCD algorithm is $O(KN^3)$.

VI. Conclusion

In this paper, we presented convolutional coding-based image separation methods to decompose a given image into a texture and a cartoon component. Our methods entail learning cartoon and texture filters directly from training examples. Using these learned filters, we proposed low-rank and sparsity-based optimization frameworks for image separation. Various experiments showed the significance of our CSC and CLC-based image separation methods over the sparse representation-based methods that use global dictionaries and patch-based methods that use local dictionaries for image separation.

In this work, we have leveraged a nested optimization approach for image decomposition. In the future, we will explore more efficient and integrated approaches to solve this problem. Furthermore, we will apply the proposed image separation methods on various computer vision problems such as intrinsic image estimation and albedo estimation that require separating a specific component from a given image.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. They are also grateful to Brendt Wohlberg for his insightful discussion and providing us the code for [35]. This work was supported by an ARO grant W911NF-16-1-0126.

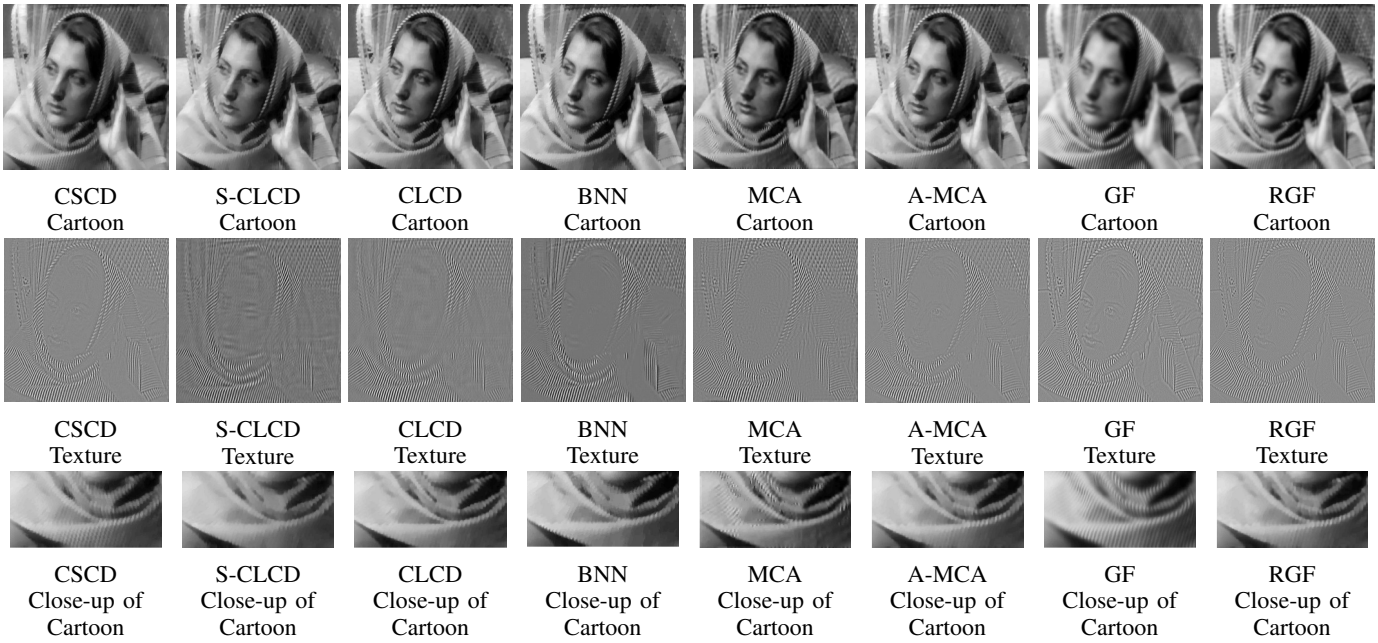


Fig. 8: Real image separation results. We compare the performance of our three methods with that of BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28].

TABLE III: Analysis of algorithms with noisy *Tiger+Texture* image.

σ^2/ PSNR (dB)	CSCD: All/Cartoon/Texture	S-CLCD: All/Cartoon/Texture	CLCD:All/Cartoon/Texture	BNN: All/Cartoon/Texture	MCA: All/Cartoon/Texture	A-MCA: All/Cartoon/Texture	GF: All/Cartoon/Texture	RGF: All/Cartoon/Texture
0.0025 (26.02)	27.98/28.05/25.77	**28.14**/27.02/**25.88**	27.99/**28.51**/25.72	25.39/27.95/24.78	26.52/28.25/24.63	28.03/28.40/24.76	*/27.83/24.46	*/28.01/24.12
0.0056 (22.50)	25.62/27.84/25.04	25.31/27.89/24.89	**25.84/28.08/25.06**	22.74/26.69/22.61	24.32/27.70/23.77	25.41/27.62/24.58	*/27.54/22.25	*/27.62/22.78
0.01 (20.04)	24.14/27.31/23.80	24.06/28.02/23.82	**24.58/27.96/24.23**	20.62/25.68/21.15	22.82/27.34/22.5 4	23.82/26.64/23.05	*/27.11/20.48	*/27.01/19.76
0.017 (17.73)	22.39/26.57/22.33	22.78/**27.22**/22.30	**22.88**/27.04/**22.82**	19.19/23.95/19.37	20.62/26.43/21.37	22.51/26.75/22.15	*/26.29/18.68	*/26.65/18.08

TABLE IV: Analysis of algorithms with noisy *Cat+Cage* image.

σ^2/PSNR (dB)	CSCD: All/Cartoon/Texture	S-CLCD: All/Cartoon/Texture	CLCD: All/Cartoon/Texture	BNN: All/Cartoon/Texture	MCA: All/Cartoon/Texture	A-MCA: All/Cartoon/Texture	GF: All/Cartoon/Texture	RGF: All/Cartoon/Texture
0.0025 (26.02)	27.65/28.27/25.89	28.19/**28.47**/**26.19**	**27.75**/28.15/25.85	25.98/27.8/23.5	26.74/27.92/24.14	26.44/25.05/22.72	*/26.49/23.95	*/27.49/23.83
0.0056 (22.51)	24.39/**28.09**/23.81	24.68/27.99/23.34	**24.97**/27.89/**24.20**	23/24.2/21.6	23.72/27.65/22.77	23.91/26.74/22.94	*/26.17/21.95	*/27.46/21.50
0.01 (20.01)	22.79/**28.01**/22.52	22.75/27.95/21.97	**23.14**/27.44/**22.84**	20.88/24.36/20.70	22.14/26.72/21.55	22.40/26.72/21.94	*/25.83/22.23	*/27.24/19.41
0.017 (17.73)	21.16/27.16/21.19	21.16/**27.33**/21.00	**21.47**/26.78/**21.41**	19.56/24.46/19.88	20.71/26.10/20.40	21.01/26.15/21.04	*/24.79/18.60	*/26.95/17.41

TABLE V: Analysis of algorithms with noisy *Boy+ Mixed Texture* image.

σ^2/PSNR (dB)	CSCD: All/Cartoon/Texture	S-CLCD: All/Cartoon/Texture	CLCD: All/Cartoon/Texture	BNN: All/Cartoon/Texture	MCA: All/Cartoon/Texture	A-MCA: All/Cartoon/Texture	GF: All/Cartoon/Texture	RGF: All/Cartoon/Texture
0.0025 (26.04)	27.30/27.29/25.92	**27.97/27.65/26.02**	27.89/27.56/25.93	25.92/27.32/24.5	26.7/26.62/23.98	26.7/26.58/24.07	*/26.50/23.99	*/26.71/23392
0.0056 (22.51)	24.92/26.32/23.09	25.32/**27.08**/25.52	**25.51**/26.58/**25.67**	22.71/27.18/22.27	23.72/25.82/22.94	24.22/25.73/23.54	*/26.27/22.06	*/26.51/21.92
0.01 (20.01)	23.48/26.06/21.92	24.10/**26.95**/24.27	**24.24**/26.19/**24.59**	20.39/26.69/20.20	22.37/25.78/21.81	23.42/25.89/22.54	*/25.79/20.33	*/26.38/19.25
0.017 (17.73)	21.78/24.93/20.00	22.19/26.24/22.37	**22.25**/25.93/**22.59**	19.56/24.46/19.88	21.15/24.91/21.12	21.68/25.02/21.71	*/25.01/18.71	*/**26.16**/17.41

APPENDIX

A. The ADMM procedure for solving (6)

Let $\mathbf{y}_r = \mathbf{y} - \sum_{k=1}^{K_t} \mathbf{d}_{t,k} * \mathbf{x}_{t,k}$, then (6) can be rewritten as

$$\hat{\mathbf{x}}_{c,k} = \arg\min_{\mathbf{x}_{c,k}} \quad \frac{1}{2}\left\|\mathbf{y}_r - \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k}\right\|_2^2 + \lambda_c \sum_{k=1}^{K_c} \|\mathbf{x}_{c,k}\|_1 + \lambda\left(\left\|(\mathbf{g_0} * \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k})\right\|_1 + \left\|(\mathbf{g_1} * \sum_{k=1}^{K_c} \mathbf{d}_{c,k} * \mathbf{x}_{c,k})\right\|_1\right). \tag{12}$$

The last term in (12) corresponding to TV can be expressed as

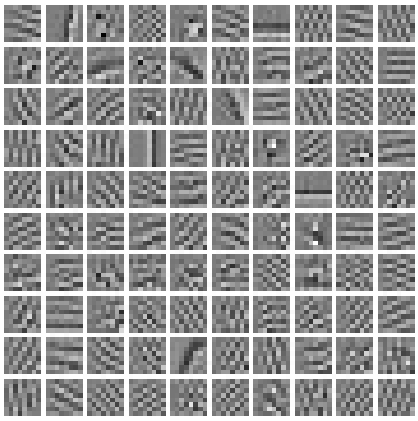
$$\left\|\sum_{k=1}^{K_c}(\mathbf{g_0} * \mathbf{d}_{c,k}) * \mathbf{x}_{c,k}\right\|_1 + \left\|\sum_{k=1}^{K_c}(\mathbf{g_1} * \mathbf{d}_{c,k}) * \mathbf{x}_{c,k}\right\|_1. \tag{13}$$

Define the linear operators $\mathbf{G}_{0,k}, \mathbf{G}_{1,k}$ and $\mathbf{D}_{c,k}$ as $\mathbf{G}_{0,k}\mathbf{x}_{c,k} = (\mathbf{g_0} * \mathbf{d}_{c,k}) * \mathbf{x}_{c,k}$, $\mathbf{G}_{1,k} = (\mathbf{g_1} * \mathbf{d}_{c,k}) * \mathbf{x}_{c,k}$, and $\mathbf{D}_{c,k}\mathbf{x}_{c,k} = \mathbf{d}_{c,k} * \mathbf{x}_{c,k}$. Then, the TV term can be expressed as

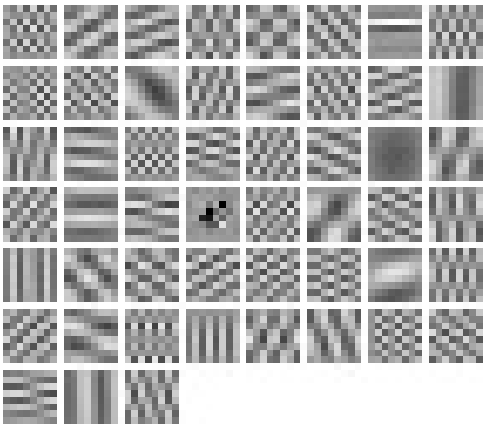
$$\left\|\sum_{k=1}^{K_c}\mathbf{G}_{0,k}\mathbf{x}_{c,k}\right\|_1 + \left\|\sum_{k=1}^{K_c}\mathbf{G}_{1,k}\mathbf{x}_{c,k}\right\|_1. \tag{14}$$

Defining $\mathbf{D} = [\mathbf{D}_{c,1}, \mathbf{D}_{c,2}, \cdots, \mathbf{D}_{c,K_c}]$, $\mathbf{x}^\top = [\mathbf{x}_{c,1}, \mathbf{x}_{c,2}, \cdots, \mathbf{x}_{c,K_c}]$, $\mathbf{H}_2 = [\mathbf{I}, \cdots, \mathbf{I}]$, and

Original Image



Fingerprints CSC Filters



Fingerprints CLC Filters

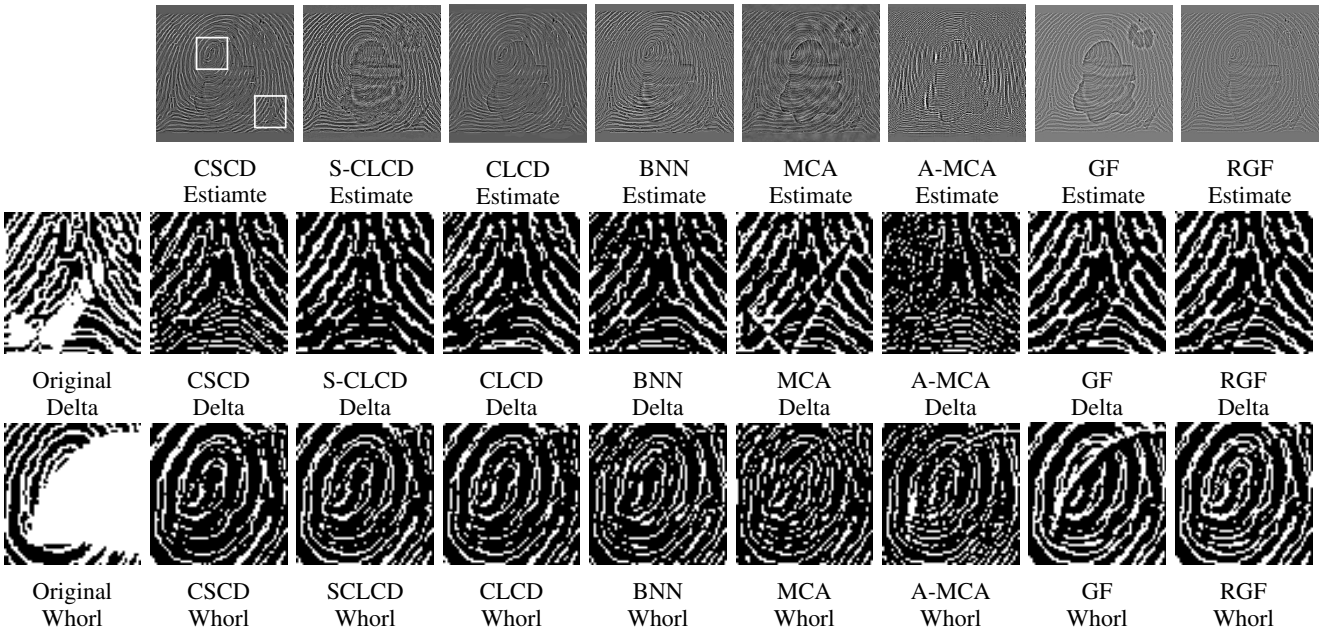


Fig. 9: Fingerprint separation results. We compare the performance of our three methods with that of BNN [24], MCA [11], A-MCA [14], GF [38] and RGF [28].

$\mathbf{H}_\ell = [\mathbf{G}_{\ell,1}, \mathbf{G}_{\ell,2}, \cdots, \mathbf{G}_{\ell,K_c}]$, $\ell = 0, 1$, (12) can be written in the ADMM form as

$$\begin{aligned} &\arg\min_{\mathbf{x},\mathbf{y}_0,\mathbf{y}_1,\mathbf{y}_2} \frac{1}{2}\|\mathbf{y}_r - \mathbf{D}\mathbf{x}\|_2^2 + \lambda_c\|\mathbf{y}_2\|_1 + \lambda\|\mathbf{y}_0\|_1 + \lambda\|\mathbf{y}_1\|_1 \\ &\text{subject to} \begin{pmatrix} \mathbf{H}_0\mathbf{x} \\ \mathbf{H}_1\mathbf{x} \\ \mathbf{H}_2\mathbf{x} \end{pmatrix} - \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = 0. \end{aligned} \tag{15}$$

Then, the iterative update rules are as follows:

$$\begin{aligned} \mathbf{x}^{(j+1)} = \arg\min_{\mathbf{x}} \quad & \frac{1}{2}\|\mathbf{y}_r - \mathbf{D}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{H}_0\mathbf{x} - \mathbf{y}_0^{(j)} + \mathbf{u}_0^{(j)}\|_2^2 \\ & + \frac{\rho}{2}\|\mathbf{H}_1\mathbf{x} - \mathbf{y}_1^{(j)} + \mathbf{u}_1^{(j)}\|_2^2 + \frac{\rho}{2}\|\mathbf{H}_2\mathbf{x} - \mathbf{y}_2^{(j)} + \mathbf{u}_2^{(j)}\|_2^2, \end{aligned} \tag{16}$$

$$\mathbf{y}_\ell{}^{(j+1)} = \arg\min_{\mathbf{y}_\ell} \quad \lambda\|\mathbf{y}_\ell\|_1 + \frac{\rho}{2}\|\mathbf{H}_\ell\mathbf{x}^{(j+1)} - \mathbf{y}_\ell + \mathbf{u}_\ell^{(j)}\|_2^2, \tag{17}$$

for $\ell = 0, 1$.

$$\mathbf{y}_2{}^{(j+1)} = \arg\min_{\mathbf{y}_2} \quad \lambda_c\|\mathbf{y}_2\|_1 + \frac{\rho}{2}\|\mathbf{H}_2\mathbf{x}^{(j+1)} - \mathbf{y}_2 + \mathbf{u}_2^{(j)}\|_2^2, \tag{18}$$

$$\mathbf{u}_\ell{}^{(j+1)} = \mathbf{u}_\ell^{(j)} + \mathbf{H}_\ell\mathbf{x}^{(j+1)} - \mathbf{y}_\ell^{(j+1)}, \ \ \ell = 0, 1, 2. \tag{19}$$

Note that (16) can be solved in the DFT domain via iterative application of the Sherman-Morrison formula [20] and the sub-problems (17) and (18) can be optimized via soft-thresholding [39].

B. The ADMM procedure for solving the CLC problem (8)

1) Fix $\mathbf{x}_{m,k}$ and update $\mathbf{d}_k$.: We solve the following optimization problem for updating each filter:

$$\begin{aligned} \arg\min_{\mathbf{d}_k} \quad & \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2 \\ \text{subject to} \quad & \|\mathbf{d}_k\|_2 \le 1, \quad \forall k. \end{aligned} \tag{20}$$

We can regard the $\|\mathbf{d}_k\|_2 \le 1$ as post-processing after each iteration. Then, (20) can be rewritten as

$$\arg\min_{\mathbf{d}_k} \quad \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2. \tag{21}$$

To solve (21) in the DFT domain, we zero pad $\mathbf{d}_k$ so that it has the same spatial support as $\mathbf{x}_{m,k}$. We form another optimization problem (22) that can directly include the zero-padding and normalization procedure for $\mathbf{d}_k$ in the objective [20] as

$$\begin{aligned} \arg\min_{\mathbf{d}_k,\mathbf{g}_k} \quad & \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2 + \sum_{k=1}^{K} l_{C_{zp}}(\mathbf{g}_k) \\ \text{subject to} \quad & \mathbf{d}_k - \mathbf{g}_k = 0 \quad \forall k, \end{aligned} \tag{22}$$

where $l_{C_{zp}}$ is the indicator function of the constraint set C_{zp} [2]. The iterative update methods for solving (22) are as follows

$$\begin{aligned} \mathbf{d}_k{}^{(j+1)} = \arg\min_{\mathbf{d}_k} \quad & \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2 \\ & + \frac{\sigma}{2}\sum_{k=1}^{K}\|\mathbf{d}_k - \mathbf{g}_k^{(j)} + \mathbf{q}_k^{(j)}\|_2^2, \end{aligned} \tag{24}$$

$$\begin{aligned} \mathbf{g}_k{}^{(j+1)} = \arg\min_{\mathbf{g}_k} \quad & \sum_{k=1}^{K} l_{C_{zp}}(\mathbf{g}_k) \\ & + \frac{\sigma}{2}\sum_{k=1}^{K}\|\mathbf{d}_k^{(j+1)} - \mathbf{g}_k + \mathbf{q}_k^{(j)}\|_2^2, \end{aligned} \tag{25}$$

$$\mathbf{q}_k{}^{(j+1)} = \mathbf{q}_k^{(j)} + \mathbf{d}_k^{(j+1)} - \mathbf{g}_k^{(j+1)}. \tag{26}$$

[2] $l_C()$ is defined as

$$l_C(p) = \begin{cases} 0, & \text{if} \quad p \in C \\ \infty, & \text{if} \quad p \notin C. \end{cases} \tag{23}$$

The optimization problem (24) can be solved using the DFT-based method proposed in [20], and (25) can be solved using the proximal operator, which can be regarded as generalized projections [40].

2) Fix $\mathbf{d}_k$ and update $\mathbf{x}_{m,k}$.: We rewrite (8) as

$$\begin{aligned} \arg\min_{\mathbf{x}_{m,k},\mathbf{z}_{m,k}} \quad & \frac{1}{2}\sum_{m=1}^{M}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2 \\ & + \lambda_l\sum_{m=1}^{M}\sum_{k=1}^{K}\|\mathbf{z}_{m,k}\|_* \\ \text{subject to} \quad & \mathbf{x}_{m,k} - \mathbf{z}_{m,k} = 0, \ \ \forall k. \end{aligned} \tag{27}$$

Then, the iterative update rules for solving (27) are as follows

$$\begin{aligned} \mathbf{x}_{m,k}{}^{(j+1)} = \arg\min_{\mathbf{x}_{m,k}} \quad & \frac{1}{2}\|\mathbf{y}_m - \sum_{k=1}^{K}\mathbf{d}_k * \mathbf{x}_{m,k}\|_2^2 \\ & + \frac{\rho}{2}\sum_{k=1}^{K}\|\mathbf{x}_{m,k} - \mathbf{z}_{m,k}^{(j)} + \mathbf{u}_{m,k}^{(j)}\|_2^2, \end{aligned} \tag{28}$$

$$\begin{aligned} \mathbf{z}_{m,k}{}^{(j+1)} = \arg\min_{\mathbf{z}_{m,k}} \quad & \lambda_l\sum_{k=1}^{K}\|\mathbf{z}_{m,k}\|_* + \\ & \frac{\rho}{2}\sum_{k=1}^{K}\|\mathbf{x}_{m,k}^{(j+1)} - \mathbf{z}_{m,k} + \mathbf{u}_{m,k}^{(j)}\|_2^2, \end{aligned} \tag{29}$$

$$\mathbf{u}_{m,k}{}^{(j+1)} = \mathbf{u}_{m,k}^{(j)} + \mathbf{x}_{m,k}^{(j+1)} - \mathbf{z}_{m,k}^{(j+1)}. \tag{30}$$

Problems (28) can be solved using the optimization method proposed in [20] and (29) can be solved using Singular Value Thresholding (SVT) [39].

REFERENCES

[1] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations: The Fifteenth Dean Jacqueline B. Lewis Memorial Lectures*. Boston, MA, USA: American Mathematical Society, 2001.

[2] L. A. Vese and S. J. Osher, "Modeling textures with total variation minimization and oscillating patterns in image processing," *Journal of Scientific Computing*, vol. 19, no. 1, pp. 553–572, 2003.

[3] Y. Liu, T. Belkina, J. H. Hays, and R. Lublinerman, "Image de-fencing."

[4] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown, "Rain streak removal using layer priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2736–2744.

[5] Y. Luo, Y. Xu, and H. Ji, "Removing rain from a single image via discriminative sparse coding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3397–3405.

[6] H. Zhang and V. M. Patel, "Convolutional sparse and low-rank coding-based rain streak removal," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 1259–1267.

[7] J.-F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle, *Scale Space Methods in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ch. Image Decomposition Application to SAR Images, pp. 297–312.

[8] V. M. Patel, G. R. Easley, R. Chellappa, and N. M. Nasrabadi, "Separated component-based restoration of speckled sar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 2, pp. 1019–1029, 2014.

[9] P. Wang, H. Zhang, and V. M. Patel, "Sar image despeckling using a convolutional neural network," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1763–1767, 2017.

[10] X. Gibert, V. M. Patel, D. Labate, and R. Chellappa, "Discrete shearlet transform on GPU with applications in anomaly detection and denoising," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, p. 64, 2014.

[11] J.-L. Starck, M. Elad, and D. L. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *Image Processing, IEEE Transactions on*, vol. 14, no. 10, pp. 1570–1582, 2005.

[12] J. Bobin, J. L. Starck, J. M. Fadili, Y. Moudden, and D. L. Donoho, "Morphological component analysis: An adaptive thresholding strategy," *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2675–2681, Nov 2007.

[13] S. Taheri, V. M. Patel, and R. Chellappa, "Component-based recognition of facesand facial expressions," *IEEE Transactions on Affective Computing*, vol. 4, no. 4, pp. 360–371, Oct 2013.

[14] G. Peyré, J. Fadili, and J.-L. Starck, "Learning the morphological diversity," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 646–669, 2010.

[15] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[16] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. Springer, 2010.

[17] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2528–2535.

[18] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 391–398.

[19] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 5135–5143.

[20] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 301–315, Jan 2016.

[21] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *International Conference on Computer Vision*, Nov 2011, pp. 2018–2025.

[22] C. Osendorfer, H. Soyer, and P. van der Smagt, *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part III*. Cham: Springer International Publishing, 2014, ch. Image Super-Resolution with Fast Approximate Convolutional Sparse Coding, pp. 250–257.

[23] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *IEEE International Conference on Computer Vision*, Dec 2015, pp. 1823–1831.

[24] S. Ono, T. Miyata, and I. Yamada, "Cartoon-texture image decomposition using blockwise low-rank texture characterization," *Image Processing, IEEE Transactions on*, vol. 23, no. 3, pp. 1128–1142, 2014.

[25] H. Schaeffer and S. Osher, "A low patch-rank interpretation of texture," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 226–262, 2013.

[26] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted guided image filtering." *IEEE Transactions on Image processing*, vol. 24, no. 1, pp. 120–129, 2015.

[27] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 839–846.

[28] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling guidance filter," in *European Conference on Computer Vision*. Springer, 2014, pp. 815–830.

[29] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *arXiv preprint arXiv:1701.05957*, 2017.

[30] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2944–2956, 2017.

[31] H. Zhang and V. M. Patel, "Convolutional sparse coding-based image decomposition," in *British Machine Vision Conference*, 2016.

[32] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[34] B. Wohlberg, "Efficient convolutional sparse coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2014, pp. 7173–7177.

[35] ——, "Convolutional sparse representations with gradient penalties," *arXiv preprint arXiv:1705.04407*, 2017.

[36] ——, "Boundary handling for convolutional sparse representations," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1833–1837.

[37] I. Y. Chun and J. A. Fessler, "Convolutional dictionary learning: Acceleration and convergence," *IEEE Transactions on Image Processing*, 2017.

[38] K. He, J. Sun, and X. Tang, "Guided image filtering," in *European conference on computer vision*. Springer, 2010, pp. 1–14.

[39] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[40] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, jan 2014.

He Zhang [S'14] received the M.S. degree in Electrical and Computer Engineering from Rutgers University, NJ, USA in 2016. He is currently a PhD candidate in the Department of Electrical and Computer Engineering at the Rutgers University, NJ, USA. His research interests include image restoration, generative adversarial network, deep learning, sparse and low-rank representation and open-set recognition.

Vishal M. Patel [SM'16] is an A. Walter Tyson Assistant Professor in the Department of Electrical and Computer Engineering at Rutgers University. Prior to joining Rutgers University, he was a member of the research faculty at the University of Maryland Institute for Advanced Computer Studies (UMIACS). He completed his Ph.D. in Electrical Engineering from the University of Maryland, College Park, MD, in 2010. His current research interests include signal processing, computer vision, and pattern recognition with applications in biometrics and imaging. He has received a number of awards including the 2016 ONR Young Investigator Award, the 2016 Jimmy Lin Award for Invention, A. Walter Tyson Assistant Professorship Award, Best Paper Award at IEEE AVSS 2017, Best Paper Award at IEEE BTAS 2015, and Best Poster Awards at BTAS 2015 and 2016. He is an Associate Editor of the IEEE Signal Processing Magazine, IEEE Biometrics Compendium, and serves on the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. He is a member of Eta Kappa Nu, Pi Mu Epsilon, and Phi Beta Kappa.