

Texture Repairing by Unified Low Rank Optimization

Xiao Liang¹, Xiang Ren², Zhengdong Zhang³, and Yi Ma⁴, *Fellow, IEEE, Member, ACM*

¹*Institute for Advanced Study, Tsinghua University, Beijing 100086, China*

²*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.*

³*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge MA 02139, U.S.A.*

⁴*School of Information Science and Technology, ShanghaiTech University, Shanghai 200031, China*

E-mail: liangx04@mails.tsinghua.edu.cn; xren7@illinois.edu; zhangzd@mit.edu; mayi@shanghaitech.edu.cn

Received October 15, 2015; revised March 3, 2016.

Abstract In this paper, we show how to harness both low-rank and sparse structures in regular or near-regular textures for image completion. Our method is based on a unified formulation for both random and contiguous corruption. In addition to the low rank property of texture, the algorithm also uses the sparse assumption of the natural image: because the natural image is piecewise smooth, it is sparse in certain transformed domain (such as Fourier or wavelet transform). We combine low-rank and sparsity properties of the texture image together in the proposed algorithm. Our algorithm based on convex optimization can automatically and correctly repair the global structure of a corrupted texture, even without precise information about the regions to be completed. This algorithm integrates texture rectification and repairing into one optimization problem. Through extensive simulations, we show our method can complete and repair textures corrupted by errors with both random and contiguous supports better than existing low-rank matrix recovery methods. Our method demonstrates significant advantage over local patch based texture synthesis techniques in dealing with large corruption, non-uniform texture, and large perspective deformation.

Keywords low-rank texture, convex optimization, sparse error correction, image repairing

1 Introduction

Image completion has been an important but challenging problem widely studied in computer vision, computer graphics, and image processing in recent years. Given an image, with intensity values of certain regions missing (due to occlusion or corruption), the goal is to automatically recover or regenerate the missing pixel values in a way that the resulting image looks visually acceptable. This is inherently an extremely ill-conditioned problem as the statement of the problem does not ensure there will be a well-defined unique solution — in theory, one can fill in arbitrary values for the missing entries.

To make this problem more well-defined, people typically seek a solution such that the completed image is in some sense statistically or structurally consistent. More specifically, it requires that the pixel val-

ues of the missing regions follow the same statistical or geometric structures as the rest of the image. In the literature, based on different assumptions on the (local or global) statistics or structures of the input image, many methods which can directly or indirectly deal with image completion problems have been developed for different types of images or textures: from the synthesis of stationary stochastic random textures^[1-3], to the inpainting of piece-wise smooth natural images^[4-6], and to the synthesis of highly symmetric and highly-structured regular textures^[7].

In this paper, we focus on the class of images or textures whose structures have very low intrinsic dimensionality or complexity. More specifically we consider textures that, when viewed as samples or signals in a high-dimensional space, span a very low-dimensional subspace or have a very sparse representation (w.r.t.

certain basis). We call such textures as “Sparse Low-Rank Textures”. A direct motivation for considering such a texture model is that many regular, symmetric patterns commonly seen in man-made environments (e.g., building facades and indoor decorations) are naturally sparse and low-rank. However, this simple texture model actually encompasses a much richer class of textures. As we will see, our method works equally well when the structure to be completed is *not* strictly periodic or stochastically stationary. In fact, our method works even when the regular patterns are distorted by certain deformations such as a nonuniform scaling or a perspective transformation, which is typically the case for completing real images. See Fig.1 for a real example.

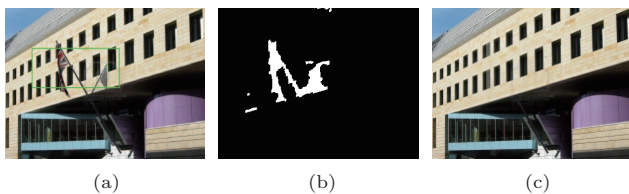


Fig.1. Sample result produced by our method. The green window in (a) denotes the input window to our system. (a) Input image. (b) Estimated support. (c) Repaired image.

Another differentiating factor for image completion methods is their assumptions about the support of the missing pixels/regions, or equivalently, what types of missing regions they intend to handle. Are the missing pixels distributed uniformly or clustered as blocks in the image? How large can the missing regions be? Or do we want to extend the texture indefinitely? The last case is often referred to as texture synthesis, which we do not consider in this paper. Many image completion and inpainting methods need to know the exact support of the missing pixels and are very sensitive to whether the support is given precisely. It is desirable that a method should be able to automatically identify and repair some corrupted or occluded regions whose statistics or structures are not consistent with the rest (say, tree branches or a street sign blocking a building facade). When the exact support of the missing pixels is unknown, the inpainting problem is usually modeled as random impulse noise removal in literature. In this case, a common approach is to estimate the support set of the noisy pixels using an outlier detection method, and then apply the reconstruction method.

Contributions. In this paper, we leverage recent breakthroughs in convex optimization for recovering sparse and low-rank structures and develop effective

and efficient methods that can automatically repair sparse low-rank textures despite unknown or partially known corruptions and despite deformations (see Fig.1 for example). By simultaneously enforcing the low-rank and sparse priors on a recovered image, our method is able to handle a much higher level of corruptions than conventional low-rank recovery techniques. We will show with extensive simulations and experiments that for the class of sparse low-rank textures, our method achieves superior performance over existing image inpainting or image completion methods, even when the support of the corruption is provided for these methods. In particular, we compare thoroughly how well these methods can handle random errors, random block errors, large contiguous errors, or deformations. In the end, we show with numerous challenging examples how our method is readily applicable to many realistic image completion and repairing tasks.

Relations to Previous Work. There are many types of methods developed for image inpainting, completion or texture synthesis. The first class of methods uses generative parametric models (say, stochastic PDEs) to describe the structures of the image signals. The completion process essentially relies on the generalizability of such models and extrapolates the missing part of the image from the given one. For example, Bertalmio *et al.*^[8] filled holes in an image by propagating image Laplacians in the isophote direction continuously from the exterior. Their PDE-based method has its roots in the Navier-Stokes equation for fluid dynamics^[9]. Oliveira *et al.*^[10] proposed an image-based algorithm based on an isotropic diffusion model extended with the notion of user-defined diffusion barriers. Levin *et al.*^[11] also performed image inpainting in the gradient domain using an image-specified prior.

There have also been studies (see [12-14] and references therein) which perform inpainting based on sparse structures of image patches: patches inside the missing pixel region are synthesized as a sparse linear combination of elements from a patch dictionary. Furthermore, Bugeau *et al.*^[15] combined geometric partial differential equation (PDEs) and patch-based texture synthesis in a variational model and performed image inpainting by minimizing the proposed energy function.

The above two classes of inpainting techniques work very well for natural images corrupted in small regions or thin gaps. However, for large missing regions that this paper will consider for image completion or repairing, they tend to generate blurring artifacts or often are not even applicable. Thus in our paper, we will not

make direct comparison with them.

In order to complete a large missing region or even extend a texture indefinitely, one must rely on more restrictive assumptions that the statistics or structures of the textures are stationary (for random textures) or homogenous (for regular patterns). The third class of methods relies on such an assumption and they essentially borrow and stitch together similar pixels or patches from given areas to complete the image. Such methods are widely used in computer graphics^[1-3,5,16-17]. While these methods are designed to work for random textures, Liu *et al.*^[7] showed how the patch-based techniques can be extended to work for regular or near-regular symmetric patterns by explicitly estimating the underlying symmetric structures of the texture. In this work, we also work on regular or near-regular textures. However, we will take a holistic approach that does not use any local statistics or patches. In addition, we directly impose regularity on the entire texture in terms of low-dimensionality and sparseness and hence completely bypass the difficult symmetry-estimation step.

Based on these studies, especially patch-based techniques, there are several studies done specifically for image completion. Criminisi *et al.*^[6] employed an exemplar-based texture synthesis technique modulated by a unified scheme for determining the fill order of the target region. Sun *et al.*^[5] utilized user interaction to specify the structure information in the image and help to do image completion. Komodakis and Tziritis^[4] treated image completion, texture synthesis and image inpainting in a unified manner by posing all of the above image-editing tasks in the form of a discrete global optimization problem. Many effective image completion systems are based on such patch-based techniques, including PatchMatch (PM) used by Photoshop^[18], image completion with structure propagation (SP) developed by Microsoft^[5], and Shift Map (SM)^[9].

In recent years, the studies on low-rank recovery and sparse representation have attracted considerable attention. Most of those methods are based on nuclear-norm minimization^[20-23]. These numerous new algorithms are widely used in image recovery and representation problems^[24-26]. For example, IRPCA (Inductive Robust Principal Component Analysis)^[22] is recently proposed to generalize RPCA (Robust Principal Component Analysis)^[20] by calculating a low-rank projection. Unlike the transductive method RPCA, IRPCA targets on the error correction and aims at representing new data inductively. Another low rank recovery

method LRR (Low-Rank Representation)^[27] has also been proposed to handle mixed data. By setting the given data matrix itself as the dictionary, the performance of LRR is largely influenced by the noisy level of input data. The optimal learning of dictionary is still very difficult in reality.

The inpainting problem basically involves detecting outliers. Early approaches for estimating the missing values use median filtering to discard outliers, such as Adaptive Median Filter (AMF)^[28]. These detection methods are widely used by impulse noise removal algorithms such as [29], which uses an absolute difference or l_1 -data fidelity term with an edge preserving regularizer to restore corrupted images. However, some methods do not use a separate support detection stage, but estimate the support or impulse noise field during the iterative process. In [30], a low rank matrix recovery method is presented where the set of patches of pixels in input image is modeled as a union of low-dimensional subspaces, and an alternating least squares method is used for reconstruction. Another single phase iterative method called blind inpainting by Adaptive Outlier Pursuit (AOP)^[31] simultaneously estimates the image and support set of the observed pixels, with wavelet regularization on the image and l_0 norm regularization on the mask.

A preliminary version of this paper has appeared as a conference version in [32]. In this fully developed journal version, we have made the following improvements over the conference version. First we provide a complete and much more detailed derivation of the Linearized Alternating Direction Method with Adaptive Penalty (LADMAP) algorithm used for the optimization. Second we introduce a more unified algorithm based on MRF to handle both random and spatially contiguous corruptions. Third we add two comparison experiments with low rank matrix recovery methods and image denoising methods respectively. Fourth we extend the method to handle curved surfaces. Fifth we provide more detailed studies of the limitations and possible extensions of the proposed method.

2 Low-Rank Textures Inpainting

In this section, we describe the mathematical model for the class of sparse low-rank textures considered in this paper.

2.1 Definition of Low-Rank Textures

We consider a 2D texture as a function $\mathbf{I}(x, y)$, defined on $\mathbb{R}^2$. We say that $\mathbf{I}$ is a low-rank texture if the family of one-dimensional functions $\{\mathbf{I}(x, y_0) \mid y_0 \in \mathbb{R}\}$ span a finite low-dimensional linear subspace, i.e.,

$$r \doteq \dim(\text{span}\{\mathbf{I}(x, y_0) \mid y_0 \in \mathbb{R}\}) \leq k$$

for some small positive integer k . If r is finite, then we refer to $\mathbf{I}$ as a rank- r texture. To a large extent, the notion of low-rank texture unifies many of the conventional local features. By this definition, it is easy to see that images of regular symmetric patterns always lead to low-rank textures.

In practice, we are never given the 2D texture as a continuous function in $\mathbb{R}^2$. Typically, we only have its values sampled on a finite discrete grid in $\mathbb{Z}^2$ of size $m \times n$. In this case, the 2D texture $\mathbf{I}(x, y)$ is represented by an $m \times n$ real matrix. For a low-rank texture, we always assume that the size of the sampling grid is significantly larger than the intrinsic rank of the texture, i.e.,

$$r \ll \min\{m, n\}.$$

It is easy to show that as long as the sampling rate is not one of the aliasing frequencies of the function $\mathbf{I}$, the resulting matrix has the same rank as the continuous function. Thus, the 2D texture $\mathbf{I}(x, y)$ when discretized as a matrix, also denoted by $\mathbf{I}$ for convenience, has very low rank relative to its dimensions.^①

2.2 Recovery of Low-Rank Textures

Suppose $\mathbf{D} \in \mathbb{R}^{m \times n}$ is the given image and Ω is the set of observed entries (pixels). The goal is to recover an image $\mathbf{I}$ of the lowest possible rank that agrees with $\mathbf{D}$ on all the observed pixels. This completion problem is equivalent to the following problem:

$$\min_{\mathbf{I}} \text{rank}(\mathbf{I}) \quad \text{s.t.} \quad \mathbf{I}_{ij} = \mathbf{D}_{ij}, \forall (i, j) \in \Omega. \quad (1)$$

Although this is in general an NP-hard problem, Candès and Recht^[33] proved that under very mild conditions, even when a significant portion of entries are missing, the optimal low-rank solution can be found by solving the convex surrogate to the above problem:

$$\min_{\mathbf{I}} \|\mathbf{I}\|_* \quad \text{s.t.} \quad P_{\Omega}(\mathbf{I}) = P_{\Omega}(\mathbf{D}),$$

where $\|\cdot\|_*$ is the nuclear norm of a matrix (i.e., the sum of singular values) and P_{Ω} is a linear operator that restricts the equality only on the entries belonging to Ω .

3 Sparse Low-Rank Texture

3.1 Exploring Spatial Structure in Low-Rank Texture

Although being low-rank is a necessary condition for most regular, structured images, it is certainly not sufficient. Fig.2 shows three images that have exactly the same rank. Obviously the first two are more smooth, realistic textures than the third one. As discussed in Subsection 2.1, a rank- r texture is a 2D function $\mathbf{I}(x, y)$, defined on $\mathbb{R}^2$. Then $\mathbf{I}(x, y)$ can be factored as

$$\mathbf{I}(x, y) = \sum_{i=1}^r g_i(x)h_i(y).$$

If $\mathbf{I}$ represents a more realistic regular or near-regular pattern, it is typically piecewise smooth. Hence, functions g_i and h_i are not arbitrary and they have additional structures. Piecewise smooth functions are typically sparse in certain transformed domain (say, by Fourier or wavelet transform).

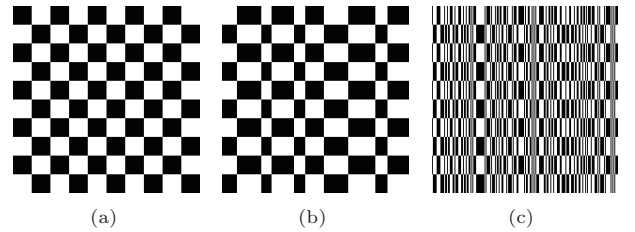


Fig.2. Example of rank's insufficiency. These three images have exactly the same rank, but they go from (a) purely regular to (b) nearly regular and to (c) nearly random textures.

Therefore, if we factorize matrix $\mathbf{I}$ as

$$\mathbf{I} = \mathbf{U}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ can be represented as

$$\mathbf{U} = \mathbf{B}_1\mathbf{X}_1 \quad \text{and} \quad \mathbf{V} = \mathbf{B}_2\mathbf{X}_2$$

for some bases $(\mathbf{B}_1, \mathbf{B}_2)$. If the bases are properly chosen, both $\mathbf{X}_1$ and $\mathbf{X}_2$ will be sparse. Or equivalently, matrix $\mathbf{W} = \mathbf{X}_1\mathbf{X}_2^T$ will be a sparse matrix, which has the same (low) rank as $\mathbf{I}$ since $\mathbf{I} = \mathbf{B}_1\mathbf{W}\mathbf{B}_2^T$.

^①According to the format requirements of JCST, and different from existing low rank matrix optimization literature, this paper denotes vectors and matrices by bold italic capital characters, and operators by italic capital characters.

3.2 Sparse Low-Rank Texture Inpainting

Hence, if we want the recovered image to be both low-rank and sparse (in certain transformed domain), we could modify the above low-rank matrix completion problem (1) as follows to impose additional spatial structures:

$$\begin{aligned} \min_{\mathbf{I}, \mathbf{W}} \text{rank}(\mathbf{I}) + \lambda \|\mathbf{W}\|_0 \\ \text{s.t. } P_{\Omega}(\mathbf{I}) = P_{\Omega}(\mathbf{D}), \mathbf{I} = \mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T, \end{aligned} \quad (2)$$

where $\|\mathbf{W}\|_0$ denotes the number of non-zero entries in $\mathbf{W}$. That is, we aim to find the texture $\mathbf{I}$ of the lowest possible rank and the transform matrix $\mathbf{W}$ with the fewest possible non-zero entries that agree with the partial observation $P_{\Omega}(\mathbf{D})$ and the transformation $\mathbf{I} = \mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T$. Here, λ is a weighting parameter which trades off the rank and the sparsity of the recovered image.

In the above problem (2), both the rank function and the ℓ_0 -norm are extremely difficult to optimize (in general NP-hard). Recent breakthroughs in sparse representation and low-rank matrix recovery have shown that under fairly broad conditions, they can be replaced by their convex surrogates ([20, 33-34]): the matrix nuclear norm $\|\mathbf{I}\|_*$ for $\text{rank}(\mathbf{I})$ and the ℓ_1 -norm $\|\mathbf{W}\|_1$ for $\|\mathbf{W}\|_0$, respectively. Thus, we end up with the following optimization problem:

$$\begin{aligned} \min_{\mathbf{I}, \mathbf{W}} \|\mathbf{I}\|_* + \lambda \|\mathbf{W}\|_1 \\ \text{s.t. } P_{\Omega}(\mathbf{I}) = P_{\Omega}(\mathbf{D}), \mathbf{I} = \mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T. \end{aligned} \quad (3)$$

If we further assume that the bases used are orthonormal, we have

$$\|\mathbf{I}\|_* = \|\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T\|_* = \|\mathbf{W}\|_*.$$

The convex program (3) is equivalent to:

$$\min_{\mathbf{W}} \|\mathbf{W}\|_* + \lambda \|\mathbf{W}\|_1 \quad \text{s.t. } P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T) = P_{\Omega}(\mathbf{D}).$$

4 Sparse Low-Rank Texture Repairing

Almost all previous methods for image inpainting or completion (e.g., [8, 12-13]) need information about the support Ω^c of the corrupted regions. The support is usually either manually marked out by the user or detected by other independent methods. This often severely limits the applicability of all the image completion or inpainting methods.

In many practical scenarios, the information about the support of the corrupted regions might not be

known or only partially known. Hence the pixels in the given region Ω can also contain some corruptions that violate the low-rank and sparse structures. We could model such corruptions as a sparse error term $\mathbf{E}$, and solve the following optimization:

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{E}} \|\mathbf{W}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 \\ \text{s.t. } P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) = P_{\Omega}(\mathbf{D}), \end{aligned} \quad (4)$$

where α is a weighting parameter between sparsity and low-rankness. Notice that if we know nothing about the corruption areas, we only need to set Ω to be the entire image. Just like Robust PCA^[20], the above convex program will decompose the image into a low-rank component and a sparse one. Of course, the nonzero entries in estimated $\mathbf{E}$ can help us to further refine the error support Ω^c . For instance, we could estimate the support by hard-thresholding the error:

$$\text{supp}(\mathbf{E}) = \{(i, j) \mid \forall |\mathbf{E}_{ij}| > \varepsilon\} \quad (5)$$

for some threshold $\varepsilon > 0$. Or we could estimate the support of $\mathbf{E}$ using a more sophisticated model to encourage additional structures such as spatial continuity. We will discuss one such option in Section 6.

We could further iterate between the image completion and the support estimation till convergence. The repaired image is obtained by $\mathbf{I}^* = \mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T$. The algorithm has been summarized in Algorithm 1. In practice, we also notice a good side effect of adding the additional $\mathbf{E}$ term: we can not only obtain the support of the corrupted regions but also reduce the noise on the repaired texture image $\mathbf{I}^*$.

5 Solving Inner Loop via Linearized Alternating Direction Method

The most computationally expensive part of Algorithm 1 is solving the convex program (4) in the inner loop. This can be solved using conventional algorithms such as interior-point methods. While interior-point methods have excellent convergence properties, they do not scale very well with problem size and hence, are unsuitable for real applications involving large images. There is recently a flurry of work in developing fast, scalable algorithms for nuclear norm minimization ([35-38]). To solve the convex optimization problem in (4), we use the LADMAP method ([39-41]). This method has been proven to be one of the fastest algorithms for solving various low-rank matrix completion and recovery problems. For the sake of completeness,

Algorithm 1. Sparse Low-Rank Texture Repairing

Input: input image $\mathbf{D} \in \mathbb{R}^{m \times n}$, initial error support Ω^0 , transformation bases $\mathbf{B}_1, \mathbf{B}_2$ and weights $\lambda > 0, \alpha > 0$
While not converged **Do**

Step 1 (inner loop): solve the convex optimization:

$$(\mathbf{W}^{i+1}, \mathbf{E}^{i+1}) \leftarrow \operatorname{argmin}_{\mathbf{W}, \mathbf{E}} \|\mathbf{W}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 \quad \text{s.t.} \quad P_{\Omega^i}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) = P_{\Omega^i}(\mathbf{D});$$

Step 2: update the error support:

$$\Omega^{c,i+1} \leftarrow \Omega^{c,i} \cup \operatorname{supp}(\mathbf{E}^{i+1}), \quad i \leftarrow i + 1;$$

End While

Output: optimal solution $\mathbf{W}^*, \mathbf{E}^*$ and repaired image $\mathbf{I}^* = \mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T$

in this section, we explain how LADMAP method can be adapted to solve our problem first.

5.1 Sketch of LADMAP

We first review the Linearized Alternating Direction Method (LADM) algorithm in a more general setting, rather than our specific problem. Let us consider convex optimization problems of the form:

$$\begin{aligned} \min_{\mathbf{x}_i} \quad & \sum_i f_i(\mathbf{x}_i) \\ \text{s.t.} \quad & \sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \mathbf{x}_i \in \mathbf{X}_i, i = 1, 2, \dots, N, \end{aligned} \quad (6)$$

where $\mathbf{X}_i$ are convex sets, $f_i(\mathbf{x}_i)$ are convex (not necessarily smooth) functions defined on $\mathbf{X}_i$, $\mathbf{A}_i$ are linear projection operators, and $\mathbf{b}$ is a vector of appropriate dimension.

For brief of notation, we abridge the collection of $\mathbf{x}_i$ into $\mathbf{X}$. For the above problem (6), we can define the augmented Lagrange function as follows:

$$\begin{aligned} L(\mathbf{X}, \mathbf{Y}) = & \sum_i f_i(\mathbf{x}_i) + \left\langle \sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i - \mathbf{b}, \mathbf{Y} \right\rangle + \\ & \frac{\mu}{2} \left\| \sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i - \mathbf{b} \right\|_F^2, \end{aligned}$$

where $\mathbf{Y}$ is a Lagrange multiplier vector of appropriate dimension, $\|\cdot\|_F$ denotes the Frobenius norm, $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, and $\mu > 0$ denotes the penalty imposed upon infeasible points.

The basic idea of Lagrangian methods is to convert the above constrained optimization problem into an unconstrained problem that has the same optimal solution. The classical ALM iteration proposed in [39]

is given by

$$\begin{aligned} \mathbf{x}_i^{k+1} = & \operatorname{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \left\langle \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b}, \mathbf{Y}_k \right\rangle + \\ & \frac{\mu_k}{2} \left\| \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} \right\|_F^2 \\ = & \operatorname{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\mu}{2} \left\| \mathbf{A}_i \mathbf{x}_i + \right. \\ & \left. \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2, \end{aligned} \quad (7)$$

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu_k \left(\sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i^{k+1} - \mathbf{b} \right),$$

$$\mu_{k+1} = \rho \mu_k,$$

where $\{\mu_k\}$ is a monotonically increasing positive sequence ($\rho > 1$), $\mathbf{x}_i^k$ and $\mathbf{Y}_k$ indicate the estimates of $\mathbf{x}_i$ and $\mathbf{Y}$ in the k -th iteration respectively. The above iteration is computationally useful only if $\mathcal{L}_\mu(\mathbf{X}, \mathbf{Y})$ is easy to minimize with respect to $\mathbf{X}$. Lin *et al.*^[41] invented the following algorithm to tackle the above optimization. The method enjoys theoretical guarantee in convergence and can easily be parallelized in implementation. For the problem in our case, we need to approximate the subproblem of $\mathbf{X}$ by linearizing the quadratic term of its objective function. With this linearization, the resulting approximation to (7) is then simple enough to have closed-form solution. More specifically, we have

$$\begin{aligned} & \left\| \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 \\ = & \left\| \sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 + \left\langle \frac{\partial}{\partial \mathbf{x}_i} \left\| \mathbf{A}_i \mathbf{x}_i + \right. \right. \\ & \left. \left. \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 \right|_{\mathbf{x}_i = \mathbf{x}_i^k}, \mathbf{x}_i - \mathbf{x}_i^k \rangle + \\ & \frac{1}{2} (\mathbf{x}_i - \mathbf{x}_i^k)^T \mathbf{H} (\mathbf{x}_i - \mathbf{x}_i^k). \end{aligned}$$

Here $\mathbf{H} = \frac{\partial^2}{\partial \mathbf{x}_i^2} \left\| \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 \Big|_{\mathbf{x}_i = \mathbf{x}_i^k}$ is the Hessian matrix. By easy calculation, we get:

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{x}_i} \left\| \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 \Big|_{\mathbf{x}_i = \mathbf{x}_i^k} \\ &= 2\mathbf{A}_i^T \left(\sum_i \mathbf{A}_i \mathbf{x}_i^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right), \mathbf{H} = 2\mathbf{A}_i^T \mathbf{A}_i. \end{aligned}$$

Also bear in mind that $\mathbf{x}^T \mathbf{A} \mathbf{x} \leq \|\mathbf{A}\|^2 \|\mathbf{x}\|^2$. Therefore we have:

$$\begin{aligned} & \left\| \mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 \\ & \leq \left\| \sum_{i=1}^N \mathbf{A}_i \mathbf{x}_i^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right\|_F^2 + \langle 2\mathbf{A}_i^T \left(\sum_i \mathbf{A}_i \mathbf{x}_i^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu} \right), \mathbf{x}_i - \mathbf{x}_i^k \rangle + \|\mathbf{A}_i\|^2 \|\mathbf{x}_i - \mathbf{x}_i^k\|^2. \end{aligned}$$

Here to ensure global convergence, Lin *et al.*^[41] proposed a modification to the coefficients of the term $\|\mathbf{x} - \mathbf{x}_i\|^2$, by replacing $\|\mathbf{A}_i\|^2$ with η such that $\eta_i \geq n\|\mathbf{A}_i\|^2$. Thus putting them together, we finally get:

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \operatorname{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\mu_k}{2} \|\mathbf{A}_i \mathbf{x}_i + \sum_{j \neq i} \mathbf{A}_j \mathbf{x}_j^k - \mathbf{b} + \frac{\mathbf{Y}_k}{\mu_k}\|_F^2 \\ &= \operatorname{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\mu_k \eta_i}{2} \|\mathbf{x}_i - \mathbf{x}_i^k + \frac{\mathbf{A}_i^T (\mu_k (\sum_i \mathbf{A}_i \mathbf{x}_i^k - \mathbf{b}) + \mathbf{Y}_k)}{\eta_i \mu_k}\|_F^2. \end{aligned}$$

5.2 Solving Inner Loop via LADMAP

To adopt LADMAP method to our problem, we need to make our objective function separable. Thus we introduce an auxiliary variable $\mathbf{A}$ to replace $\mathbf{W}$ in the low-rank term, which turns the optimization problem into the following:

$$\begin{aligned} & \min_{\mathbf{A}, \mathbf{E}, \mathbf{W}} \|\mathbf{A}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 \\ & \text{s.t. } \mathbf{A} = \mathbf{W}, P_\Omega(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) = P_\Omega(\mathbf{D}). \end{aligned} \quad (8)$$

For problem (8), the augmented Lagrangian function is defined as:

$$\begin{aligned} & \mathcal{L}_\mu(\mathbf{A}, \mathbf{W}, \mathbf{E}, \mathbf{Y}_1, \mathbf{Y}_2) \\ &= \|\mathbf{A}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 + \langle \mathbf{Y}_1, \mathbf{A} - \mathbf{W} \rangle + \langle \mathbf{Y}_2, P_\Omega(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) - P_\Omega(\mathbf{D}) \rangle + \frac{\mu}{2} \|\mathbf{A} - \mathbf{W}\|_F^2 + \frac{\mu}{2} \|P_\Omega(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) - P_\Omega(\mathbf{D})\|_F^2, \end{aligned} \quad (9)$$

where $\mathbf{Y}_1, \mathbf{Y}_2$ are the Lagrange multipliers, and $\mu > 0$ is a penalty parameter.

From previous discussion in Subsection 5.1, the classic linearized ALM iteration scheme for our problem is given by

$$\begin{aligned} & (\mathbf{A}_{k+1}, \mathbf{W}_{k+1}, \mathbf{E}_{k+1}) \\ &= \operatorname{argmin}_{\mathbf{A}, \mathbf{W}, \mathbf{E}} L_{\mu_k}(\mathbf{A}, \mathbf{W}, \mathbf{E}, \mathbf{Y}_{1,k}, \mathbf{Y}_{2,k}), \\ & \mathbf{Y}_{1,k+1} = \mathbf{Y}_{1,k} + \mu_k P_\Omega(\mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T + \mathbf{E}_{k+1} - \mathbf{D}), \\ & \mathbf{Y}_{2,k+1} = \mathbf{Y}_{2,k} + \mu_k (\mathbf{A}_{k+1} - \mathbf{W}_{k+1}), \\ & \mu_{k+1} = \rho \mu_k, \end{aligned}$$

where $\rho > 1$ is a constant.

We now focus on efficiently solving the first step of the above iterative scheme. In general, it is computationally expensive to minimize over all the variables $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{E}$ simultaneously. Therefore, we adopt a common strategy to solve it approximately by adopting an alternating minimizing strategy, i.e., minimizing with respect to $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{E}$ one at a time:

$$\mathbf{A}_{k+1} = \operatorname{argmin}_{\mathbf{A}} \mathcal{L}_{\mu_k}(\mathbf{A}, \mathbf{W}_k, \mathbf{E}_k, \mathbf{Y}_{1,k}, \mathbf{Y}_{2,k}), \quad (10)$$

$$\mathbf{W}_{k+1} = \operatorname{argmin}_{\mathbf{W}} \mathcal{L}_{\mu_k}(\mathbf{A}_{k+1}, \mathbf{W}, \mathbf{E}_k, \mathbf{Y}_{1,k}, \mathbf{Y}_{2,k}), \quad (11)$$

$$\mathbf{E}_{k+1} = \operatorname{argmin}_{\mathbf{E}} \mathcal{L}_{\mu_k}(\mathbf{A}_{k+1}, \mathbf{W}_{k+1}, \mathbf{E}, \mathbf{Y}_{1,k}, \mathbf{Y}_{2,k}). \quad (12)$$

Subproblem (10) is easy to optimize and we can get to the closed-form solution without doing any approximation:

$$\begin{aligned} \mathbf{A}_{k+1} &= \operatorname{argmin}_{\mathbf{A}} \|\mathbf{A}\|_* + \frac{\mu_k}{2} \|\mathbf{A} - \mathbf{W}_k + \frac{1}{\mu_k} \mathbf{Y}_{1,k}\|_F^2 \\ &= \mathbf{S}_{\frac{1}{\mu_k}} \left(\mathbf{W}_k - \frac{1}{\mu_k} \mathbf{Y}_{1,k} \right). \end{aligned} \quad (13)$$

Here we use $S_\varepsilon(\cdot)$ as the singular value shrinkage operator:

$$S_\varepsilon(\mathbf{X}) = \mathbf{U} T_\varepsilon(\mathbf{\Sigma}) \mathbf{V}^T,$$

in which $\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ is the SVD of $\mathbf{X}$ and $T_\varepsilon(\cdot)$ represents the soft-thresholding operator which is defined on scalars as follows:

$$T_\varepsilon(x) = \operatorname{sgn}(x) \times (|x| - \varepsilon).$$

The soft-thresholding operator is extended to vectors and matrices by applying it elementwise.

Subproblems (11) and (12) need linear approximation since there are linear operators on variables $\mathbf{E}$ and $\mathbf{W}$. As we discussed in Subsection 5.1, we can approximate the objective function by linearizing the quadratic penalty term in the augmented Lagrangian function (9)

and adding a proximal term to update $\mathbf{W}$ and $\mathbf{E}$. For $\mathbf{W}$, we have

$$\begin{aligned} & \mathbf{W}_{k+1} \\ &= \operatorname{argmin}_{\mathbf{W}} \lambda \|\mathbf{W}\|_1 + \langle \mathbf{Y}_{1,k}, \mathbf{A}_{k+1} - \mathbf{W} \rangle + \langle \mathbf{Y}_{2,k}, \\ & \quad P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) \rangle + \frac{\mu_k}{2} \|\mathbf{A}_k - \mathbf{W}\|_F^2 + \\ & \quad \frac{\mu_k}{2} \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D})\|_F^2 \\ &= \operatorname{argmin}_{\mathbf{W}} \lambda \|\mathbf{W}\|_1 + \frac{\mu_k}{2} \|\mathbf{A}_{k+1} - \mathbf{W} + \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \\ & \quad \frac{\mu_k}{2} \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2. \end{aligned} \quad (14)$$

Then we should linearize $\|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2$ with respect to $\mathbf{W}$. Here we want to emphasize the representation for Jacobian of $P_{\Omega}(\mathbf{X})$ with respect to $\mathbf{X}$. Since $P_{\Omega}(\cdot)$ is a linear operator, we have $P_{\Omega}(\mathbf{X} + \Delta \mathbf{X}) = P_{\Omega}(\mathbf{X}) + P_{\Omega}(\Delta \mathbf{X})$. Therefore here by a little abuse of notation, we write $P_{\Omega}(\mathbf{X})$ as $P_{\Omega} \cdot \mathbf{X}$. Then we have $\frac{\partial P_{\Omega}(\mathbf{X})}{\partial \mathbf{X}} = P_{\Omega}$. Another thing we want to put emphasis on is that since $\langle P_{\Omega}(\mathbf{X}), P_{\Omega^c}(\mathbf{X}) \rangle = 0$, we have $\langle P_{\Omega}(\mathbf{A}), P_{\Omega}(\mathbf{B}) \rangle = \langle P_{\Omega}(\mathbf{A}), \mathbf{B} \rangle$. From the

property of $P_{\Omega}(\cdot)$, the following equations hold: $P_{\Omega}^T = P_{\Omega}$, $P_{\Omega} P_{\Omega} = P_{\Omega}$. Thus

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{W}} \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \Big|_{\mathbf{W}=\mathbf{W}_k} \\ &= 2\mathbf{B}_1^T P_{\Omega}^* P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2 \\ &= 2\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2. \end{aligned}$$

And the hessian matrix is:

$$\begin{aligned} \mathbf{H} &= \frac{\partial^2}{\partial \mathbf{W}^2} \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \Big|_{\mathbf{W}=\mathbf{W}_k} \\ &= \frac{\partial}{\partial \mathbf{W}} 2\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2 \Big|_{\mathbf{W}=\mathbf{W}_k} \\ &= 2(\mathbf{B}_2^T \otimes \mathbf{B}_1^T)(P_{\Omega} \cdot (\mathbf{B}_2 \otimes \mathbf{B}_1)), \end{aligned}$$

where $\otimes$ denotes Kronecker product. Choosing $\mathbf{B}_1$ and $\mathbf{B}_2$ as standard orthogonal basis, we have $\|\mathbf{H}/2\|^2 = \|(\mathbf{B}_2^T \otimes \mathbf{B}_1^T)(P_{\Omega} \cdot (\mathbf{B}_2 \otimes \mathbf{B}_1))\|^2 = \|P_{\Omega}\|^2 \leq 1$. Therefore, we have

$$\begin{aligned} & \|\mathbf{A}_{k+1} - \mathbf{W} + \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \\ &= \|\mathbf{W} - \mathbf{W}_k + \mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \\ &\leq \|\mathbf{W} - \mathbf{W}_k\|_F^2 + \langle 2(\mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}), \mathbf{W} - \mathbf{W}_k \rangle + \|\mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \|P_{\Omega}\|^2 \|\mathbf{W} - \mathbf{W}_k\|_F^2 + \\ & \quad \langle 2\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2, \mathbf{W} - \mathbf{W}_k \rangle + \|P_{\Omega}(\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \\ &= (1 + \|P_{\Omega}\|^2) \|\mathbf{W} - \mathbf{W}_k\|_F^2 + \langle 2(\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2 + \mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}), \mathbf{W} - \mathbf{W}_k \rangle + \\ & \quad \|\mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \|P_{\Omega}(\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2 \\ &= (1 + \|P_{\Omega}\|^2) \|\mathbf{W} - \mathbf{W}_k\|_F^2 + \frac{\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \mathbf{Y}_{2,k}/\mu_k) \mathbf{B}_2 + \mathbf{W}_k - \mathbf{A}_{k+1} - \mathbf{Y}_{1,k}/\mu_k}{1 + \|P_{\Omega}\|^2} \Big|_F^2 - \\ & \quad \frac{\|\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \mathbf{Y}_{2,k}/\mu_k) \mathbf{B}_2 + \mathbf{W}_k - \mathbf{A}_{k+1} - \mathbf{Y}_{1,k}/\mu_k\|_F^2}{1 + \|P_{\Omega}\|^2} + \\ & \quad \|\mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k}\|_F^2 + \|P_{\Omega}(\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}\|_F^2. \end{aligned}$$

According to (14), for the update of $\mathbf{W}$, we need to solve the following sub-optimization:

$$\begin{aligned} \mathbf{W}_{k+1} &= \operatorname{argmin}_{\mathbf{W}} \lambda \|\mathbf{W}\|_1 + \frac{\mu_k \eta_1}{2} \|\mathbf{W} - \mathbf{W}_k + \\ & \quad \frac{\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \mathbf{Y}_{2,k}/\mu_k) \mathbf{B}_2 + \mathbf{W}_k - \mathbf{A}_{k+1} - \mathbf{Y}_{1,k}/\mu_k}{\eta_1} \Big|_F^2 \\ &= T_{\frac{\lambda}{\mu_k \eta_1}} \left(\mathbf{W}_k - \frac{\mathbf{B}_1^T P_{\Omega} (\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{1}{\mu_k} \mathbf{Y}_{2,k}) \mathbf{B}_2 + \mathbf{W}_k - \mathbf{A}_{k+1} - \frac{1}{\mu_k} \mathbf{Y}_{1,k}}{\eta_1} \right). \end{aligned} \quad (15)$$

Similarly for $\mathbf{E}$, we have:

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{E}} \left\| P_{\Omega}(\mathbf{E}) + P_{\Omega}(\mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{I}) + \frac{\mathbf{Y}_{2,k}}{\mu_k} \right\|_F^2 \Big|_{\mathbf{E}=\mathbf{E}_k} = P_{\Omega}^T 2(P_{\Omega}(\mathbf{E}) + P_{\Omega}(\mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{I}) + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \\
& = 2P_{\Omega}(\mathbf{E} + \mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{I} + \frac{\mathbf{Y}_{2,k}}{\mu_k}), \\
& \frac{\partial^2}{\partial \mathbf{E}^2} \left\| P_{\Omega}(\mathbf{E}) + P_{\Omega}(\mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{I}) + \frac{\mathbf{Y}_{2,k}}{\mu_k} \right\|_F^2 \Big|_{\mathbf{E}=\mathbf{E}_k} = 2P_{\Omega}. \\
& \mathbf{E}_{k+1} = \operatorname{argmin}_{\mathbf{E}} \alpha \|\mathbf{E}\|_1 + \frac{\mu_k \eta_2}{2} \|\mathbf{E} - \mathbf{E}_k + \frac{P_{\Omega}(\mathbf{E}_k + \mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{D} + \mathbf{Y}_{2,k}/\mu_k)}{\eta_2}\|_F^2 \\
& = T_{\frac{\alpha}{\mu_k \eta_2}} \left(\mathbf{E}_k - \frac{P_{\Omega}(\mathbf{E}_k + \mathbf{B}_1 \mathbf{W}_{k+1} \mathbf{B}_2^T - \mathbf{D}) + \frac{1}{\mu_k} \mathbf{Y}_{2,k}}{\eta_2} \right). \tag{16}
\end{aligned}$$

Because $\|P_{\Omega}\|^2 \leq 1$, $1 + \|P_{\Omega}\|^2 \leq 2$, we choose $\eta_1 = \eta_2 = 3$ in this paper.

5.3 Stopping Criteria

According to the KKT conditions, there exists optimal solution $(\mathbf{A}^*, \mathbf{W}^*, \mathbf{E}^*, \mathbf{Y}_1^*, \mathbf{Y}_2^*)$ such that

$$\begin{aligned}
& \mathbf{A}^* = \mathbf{W}^*, P_{\Omega}(\mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T + \mathbf{E}^*) = P_{\Omega}(\mathbf{D}), \tag{17} \\
& 0 \in \partial_{\mathbf{A}} L_{\mu} \Big|_{\mathbf{A}=\mathbf{A}^*}, 0 \in \partial_{\mathbf{W}} L_{\mu} \Big|_{\mathbf{W}=\mathbf{W}^*}, 0 \in \partial_{\mathbf{E}} L_{\mu} \Big|_{\mathbf{E}=\mathbf{E}^*}. \tag{18}
\end{aligned}$$

Here $\partial_X(L_{\mu})$ denotes the partial derivative of L_{μ} with respect to X . The first stopping criterion (17) is the feasibility:

$$\begin{aligned}
& \|\mathbf{A}^* - \mathbf{W}^*\|^2 \leq \epsilon_1, \\
& \|P_{\Omega}(\mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T + \mathbf{E}^*) - P_{\Omega}(\mathbf{D})\|^2 \leq \epsilon_2.
\end{aligned}$$

The stopping criteria according to the second KKT condition (18) are not so intuitive. Take $0 \in \partial_{\mathbf{W}} L_{\mu} \Big|_{\mathbf{W}=\mathbf{W}^*}$ for example. We should notice that the nuclear norm and the l_1 norm are both non-differentiable, thereby here we use $\partial_X(\|\mathbf{X}\|_*)$ and $\partial_X(\|\mathbf{X}\|_1)$ to denote the subgradient of nuclear norm and l_1 norm respectively. From (15), we have

$$\begin{aligned}
& 0 \in \partial_{\mathbf{W}} \left(\lambda \|\mathbf{W}\|_1 + \frac{\mu_k \eta_1}{2} \|\mathbf{W} - \mathbf{W}_k + \frac{1}{\eta_1} (\mathbf{B}_1^T P_{\Omega}(\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2 + \right. \\
& \left. \mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k})\|_F^2 \right) \Big|_{\mathbf{W}=\mathbf{W}_{k+1}} \\
& = \partial_{\mathbf{W}} (\lambda \|\mathbf{W}\|_1) \Big|_{\mathbf{W}=\mathbf{W}_{k+1}} + \mu_k \eta_1 (\mathbf{W}_{k+1} - \mathbf{W}_k) + \\
& \mu_k \left(\mathbf{B}_1^T P_{\Omega}(\mathbf{B}_1 \mathbf{W}_k \mathbf{B}_2^T + \mathbf{E}_k - \mathbf{D} + \frac{\mathbf{Y}_{2,k}}{\mu_k}) \mathbf{B}_2 + \right. \\
& \left. \mathbf{W}_k - \mathbf{A}_{k+1} - \frac{\mathbf{Y}_{1,k}}{\mu_k} \right). \tag{19}
\end{aligned}$$

$$0 \in \partial_{\mathbf{W}} L_{\mu}(\mathbf{A}^*, \mathbf{W}, \mathbf{E}^*, \mathbf{Y}_1^*, \mathbf{Y}_2^*) \Big|_{\mathbf{W}=\mathbf{W}^*} :$$

$$\begin{aligned}
& 0 \in \partial_{\mathbf{W}} \left(\lambda \|\mathbf{W}\|_1 + \frac{\mu}{2} \|\mathbf{A}^* - \mathbf{W} + \frac{\mathbf{Y}_1^*}{\mu}\|_F^2 + \right. \\
& \left. \frac{\mu}{2} \|P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}^* - \mathbf{D}) + \frac{\mathbf{Y}_2^*}{\mu}\|_F^2 \right) \Big|_{\mathbf{W}=\mathbf{W}^*} \\
& = \partial_{\mathbf{W}} (\lambda \|\mathbf{W}\|_1) \Big|_{\mathbf{W}=\mathbf{W}^*} - \mu (\mathbf{A}^* - \mathbf{W}^* + \frac{\mathbf{Y}_1^*}{\mu}) + \\
& \mu (\mathbf{B}_1^T P_{\Omega}(\mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T + \mathbf{E}^* - \mathbf{D} + \frac{\mathbf{Y}_2^*}{\mu}) \mathbf{B}_2) \\
& = \partial_{\mathbf{W}} (\lambda \|\mathbf{W}\|_1) \Big|_{\mathbf{W}=\mathbf{W}^*} + \mu \left(\mathbf{B}_1^T P_{\Omega}(\mathbf{B}_1 \mathbf{W}^* \mathbf{B}_2^T + \right. \\
& \left. \mathbf{E}^* - \mathbf{D} + \frac{\mathbf{Y}_2^*}{\mu}) \mathbf{B}_2 + \mathbf{W}^* - \mathbf{A}^* - \frac{\mathbf{Y}_1^*}{\mu} \right). \tag{20}
\end{aligned}$$

Comparing (19) and (20), $\mu \eta_1 \|\mathbf{W}_{k+1} - \mathbf{W}_k\|^2$ should be small enough. This leads to the second stopping criterion:

$$\mu \eta_1 \|\mathbf{W}_{k+1} - \mathbf{W}_k\|^2 \leq \epsilon_{\mathbf{W}}.$$

Similarly, we can get another two stopping criteria: $\mu \|\mathbf{A}_{k+1} - \mathbf{A}_k\|^2 \leq \epsilon_{\mathbf{A}}$, $\mu \eta_2 \|\mathbf{E}_{k+1} - \mathbf{E}_k\|^2 \leq \epsilon_{\mathbf{E}}$.

Finally, we have finished solving problem (8). We summarize the LADMAP scheme for solving (8) in Algorithm 2. We choose sequence $\{\mu_k\}$ to satisfy $\mu_{k+1} = \rho \mu_k$ for some $\rho > 1$.

To show the advantage of adding the robust error term $\mathbf{E}$ for support detection, we do a comparison with another low-rank texture completion method based on matrix completion^[42]. In Fig.3, we first corrupt an input low-rank image with a ground truth support Ω , and then we generate a new support $\tilde{\Omega}$ by randomly switching 5% of support Ω (from 1 to 0 or from 0 to 1). We use $\tilde{\Omega}$ as the input support for our method and the LRTC method^[42] and compare the completion results. In the comparison, we set parameters $\lambda = 0.001$, $\alpha = 0.85$ for our algorithm and use DCT as the basis for $\mathbf{B}_1$ and $\mathbf{B}_2$. For the LRTC method, all parameters are set as the same values as those in the public

Algorithm 2. Solving Inner Loop via LADMAP

Input: input image $D \in \mathbb{R}^{m \times n}$, error support Ω , transformation bases B_1, B_2 , and weights $\lambda > 0, \alpha > 0$
Initialization: $k = 0, A_0 = D, E_0 = 0, W_0 = 0, Y_{1,0} = 0, Y_{2,0} = 0, \mu_0 > 0, \rho > 1$;
While not converged **Do**
 Update A_{k+1} using (13);
 Update W_{k+1} using (15);
 Update E_{k+1} using (16);
 $Y_{1,k+1} = Y_{1,k} + \mu_k P_{\Omega}(B_1 W_{k+1} B_2^T + E_{k+1} - D)$;
 $Y_{2,k+1} = Y_{2,k} + \mu_k (A_{k+1} - W_{k+1})$;
 $\mu_{k+1} = \rho \mu_k$;
End While
Output: solution (A, W, E) to (4)

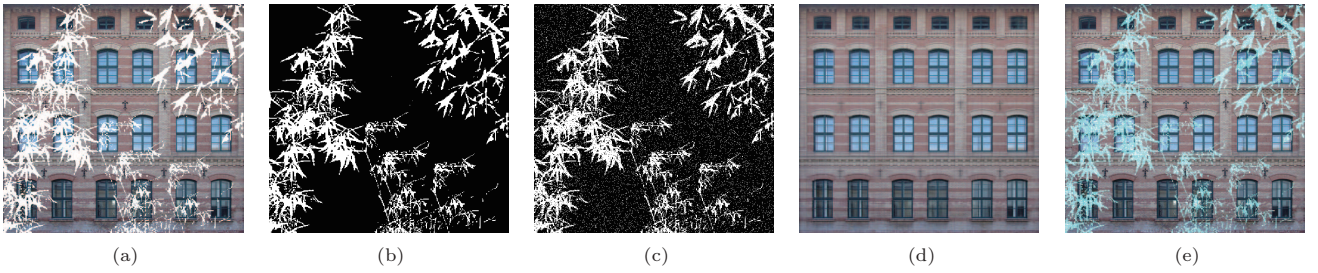


Fig.3. Robustness to imperfect input support. This figure shows that with 5% support map corrupted, our algorithm recovers the image while the method based on low-rank matrix completion LRTC^[42] fails. (a) Input image. (b) True support. (c) Input support. (d) Our result. (e) LRTC result.

code of LRTC. The results in Fig.3 clearly show that our method can handle erroneous input support while other matrix completion methods would fail.

6 Handling Error E with Contiguous Support

In a real image, very often the regions we need to repair have contiguous support — occluded or corrupted pixels are likely to be adjacent to each other in the image. The simple hard-thresholding used in (5) treats each pixel independently and does not take this additional information for estimating the error support. To incorporate such prior information, we could use a Markov random field (MRF) to model the spatial continuity of the error support $\Omega^c = \text{supp}(Err) = \text{supp}(D - B_1 W^* B_2^T)$, as shown in [43]. We do not make any other assumptions about locations, sizes, shapes, or the number of occluded regions.

Following a similar strategy in [43] to estimate a continuity-prone support, we only have to replace the support update in (5) in Algorithm 1 by the following:

$$\text{supp}(Err) = \underset{\Omega \in \{-1,1\}^{m \times n}}{\text{argmax}} \sum_{(l,n) \in M} \gamma s[l]s[n] +$$

$$\sum_{l \in V} \log(p(e[l]|s[l])), \quad (21)$$

with

$$\log(p(e[l]|s[l] = -1)) = \begin{cases} -\log \beta, & \text{if } |e[l]| \leq \beta, \\ \log \beta, & \text{if } |e[l]| > \beta, \end{cases}$$

$$\log(p(e[l]|s[l] = 1)) = \begin{cases} 0, & \text{if } |e[l]| > \beta, \\ \log \beta, & \text{if } |e[l]| \leq \beta. \end{cases}$$

$G = (V, M)$ is a graph which represents the image domain, $s = \text{vec}(\Omega) \in \mathbb{R}^{mn}$, $e = \text{vec}(E) \in \mathbb{R}^{mn}$, $V = \{1, \dots, mn\}$ denotes the set of $m \times n$ pixels and M denotes the edges connecting neighboring pixels. Here, parameter β indicates the level of error we would accept before considering an entry of the image as being occluded. Parameter γ in the Markov random field model controls the strength of mutual interaction between adjacent pixels. It should correspond to the level of spatial continuity for the error supports. Fig.4 shows how MRF works. We use $\beta = 0.015, \gamma = 5$ and $\lambda = 0.001, \alpha = 0.85$ in the example (empirically settings since they bring the best average performance).

According to (21), the optimization function would degenerate to hard threshold if we set $\gamma = 0$. There-

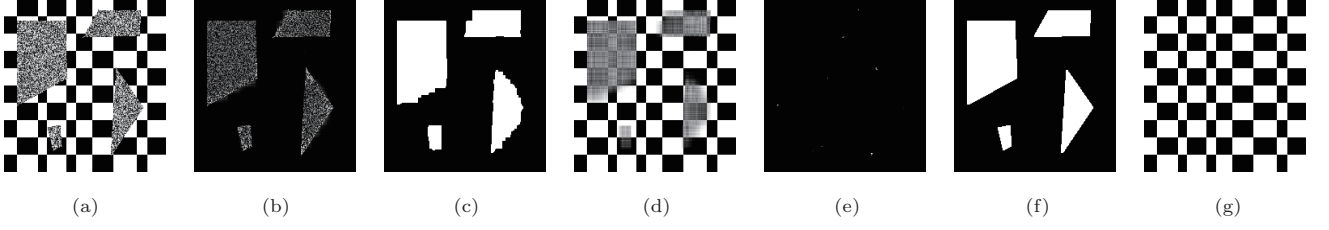


Fig. 4. Estimation of contiguous error support using MRF. This figure shows an example of estimating a contiguous support by our method after merging MRF into our model. It takes only two iterations for our method to converge precisely to the correct support. (a) Input $\mathbf{D}$. (b) $\mathbf{E}^1$. (c) $\mathbf{\Omega}^1$. (d) $\mathbf{I}^1$. (e) $\mathbf{E}^2$. (f) $\mathbf{\Omega}^2$. (g) $\mathbf{I}^2$.

fore, in order to handle both random and contiguous corruption, we run MRF twice with different parameters respectively: $\gamma = 0, \beta = 0.3$ and $\gamma = 5, \beta = 0.015$. The first result $\mathbf{\Omega}_1^c$ is equal to hard threshold and the second one $\mathbf{\Omega}_2^c$ is the result of continuous support. If $\|\mathbf{\Omega}_2^c\|_0 > 2\|\mathbf{\Omega}_1^c\|_0$, we consider that MRF successfully recovers the continuous support and use $\mathbf{\Omega}_2^c$ as the estimated support. If $\|\mathbf{\Omega}_1^c\|_0/2 < \|\mathbf{\Omega}_2^c\|_0 \leq 2\|\mathbf{\Omega}_2^c\|_0$, this support usually includes both contiguous and random corruption. Then the estimated support should be $\mathbf{\Omega}^c = \mathbf{\Omega}_1^c \cup \mathbf{\Omega}_2^c$. Otherwise, we use $\mathbf{\Omega}^c = \mathbf{\Omega}_1^c$. Fig. 5 shows an example. The result indicates this proposed method works better than simply MRF. And we will also find this sample strategy works very well via the massive results shown in Section 9. If not specified, all the other experiments in this paper use MRF to build support with the same parameters and strategies as here.

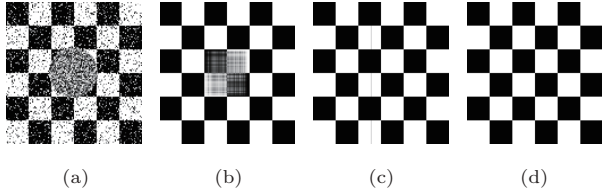


Fig. 5. Comparing proposed method with simply MRF. This figure shows an example of estimating a support with both contiguous and random corruption. (a) Input $\mathbf{D}$. (b) $\gamma = 0, \beta = 0.3$. (c) $\gamma = 5, \beta = 0.015$. (d) Proposed method.

7 Handling Deformed Texture $\mathbf{D}$

In practice, textures in a real image that we need to perform the completion or repairing task rarely have precise regular patterns. Very often we see a distorted version of the regular patterns. For instance, a building facade seen through a perspective camera introduces a planar homography between the facade plane and the image plane.

In this case, the recovered low-rank texture $\mathbf{I} = \mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T$ agrees with the observed texture $\mathbf{D}$ up to certain transformation. To recover $\mathbf{I}$, we then need to solve the following problem instead:

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{E}, \tau} & \|\mathbf{W}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 \\ \text{s.t.} & P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) = P_{\Omega}(\mathbf{D} \circ \tau), \end{aligned}$$

where $\tau : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ belongs to a certain group of transforms, e.g., affine transform, perspective transforms^[44], and even general cylindrical deformation^[45]. Some results are shown in Subsection 9.6.1. The above problem is not a convex program as the constraint is now nonlinear. Similar to the work of TILT^[44], we could linearize $\mathbf{D} \circ \tau$ at previous τ^i as

$$\mathbf{D} \circ (\tau^i + \Delta\tau) \approx \mathbf{D} \circ \tau^i + \mathbf{J} \Delta\tau,$$

where $\mathbf{J}$ is the Jacobian: the derivative of the image with respect to the transformation parameters. Then, to handle deformation, our method can be easily modified as follows:

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{E}, \Delta\tau} & \|\mathbf{W}\|_* + \lambda \|\mathbf{W}\|_1 + \alpha \|\mathbf{E}\|_1 \\ \text{s.t.} & P_{\Omega}(\mathbf{B}_1 \mathbf{W} \mathbf{B}_2^T + \mathbf{E}) = P_{\Omega}(\mathbf{D} \circ \tau + \mathbf{J} \Delta\tau). \end{aligned} \quad (22)$$

For ease of representation, we will omit the subscript i on $\mathbf{\Omega}$. We summarize the algorithm to handle deformation τ in the sparse low-rank texture repairing work as Algorithm 3.

Solving (22) is similar to solving (4) with LADMAP method. The algorithm runs in the same spirit as TILT, except that it further imposes the sparsity of the recovered texture and repairs the image once the locations of errors are detected. Nevertheless the improvement of our method in repairing the rectified texture is very significant. Fig. 6 shows an example that compares the results of our method with the TILT code released by the authors of [44].

Algorithm 3. Deformed Sparse Low-Rank Texture Repairing

Input: input image $D \in \mathbb{R}^{m \times n}$, initial transformation $\tau \in \mathbb{G}$ (affine or projective), initial error support Ω^0 , Jacobian matrix J , transformation bases B_1, B_2 and weights $\lambda > 0, \alpha > 0$

While not converged **Do**

Step 1 (inner loop): solve the convex optimization:

$$\min_{W, E, \Delta\tau} \|W\|_* + \lambda \|W\|_1 + \alpha \|E\|_1 \quad \text{s.t.} \quad P_{\Omega^i}(B_1 W B_2^T + E) = P_{\Omega^i}(D \circ \tau^i + J \Delta\tau);$$

Step 2: update the transformation: $\tau^{i+1} = \tau^i + \Delta\tau^{i+1}$;

Step 3: update the error support: $\Omega^{i+1} \leftarrow \text{supp}(D \circ \tau^{i+1} - B_1 W^{i+1} B_2^T), \quad i \leftarrow i + 1$;

End While

Output: optimal solution W^*, E^*, τ^* and repaired image $I^* = B_1 W^* B_2^T$

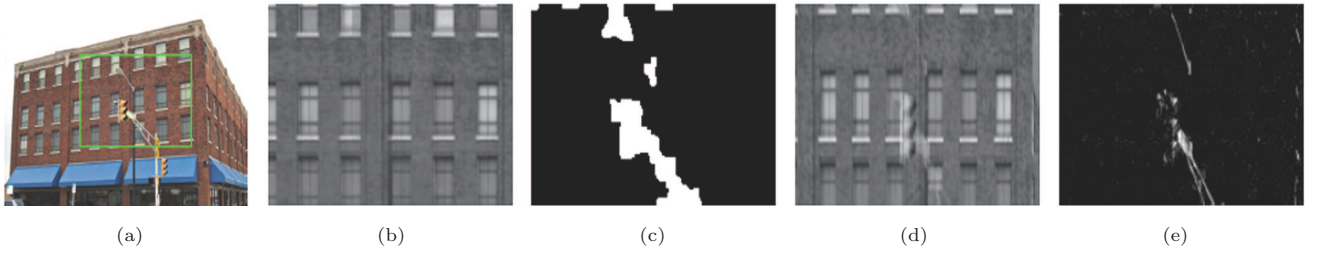


Fig.6. Comparison with TILT^[44]. Notice that TILT does not (intend to) fully repair the occluded parts while our method does. (a) Input image. (b) Our result. (c) Our support. (d) TILT result. (e) E of TILT.

8 Repair Enhancement with Texture Synthesis

As we see in the above results, our algorithm could recover the low-rank and sparse structures in regular or near-regular textures very well. However, because the output of our algorithm is the optimization result of a convex problem, our output result is sometimes not so sharp as the original input image. To handle this problem, one may add a post-processing step based on a simple patch-matching method: simply borrowing raw pixels from the input image based on local similarity. A naive process can be as follows. For every pixel in the repaired region, we take a, say 15×15 , patch centered at the pixel, and find its closest matched patch in the original input image. We then replace the pixel value with that from the original image. After this post-processing step, all pixels of the final image now come from the original image. Both sharpness and noise-characteristics would be the same as the input image.

Fig.7 shows such an example. This suggests that our method can be combined with existing local patch based texture synthesis methods. The result system not only could remedy the limitations of both approaches, but also could simultaneously handle both regular and random textures. We leave such work for future investigation.

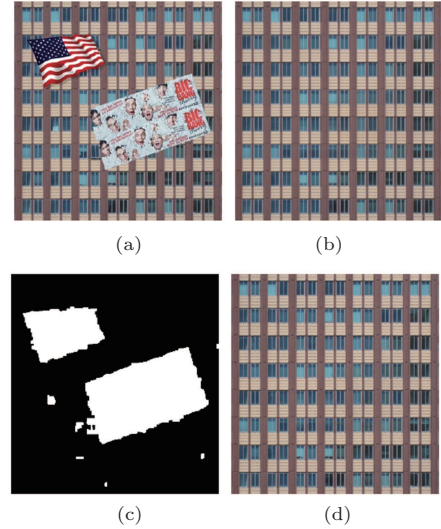


Fig.7. Patch-matching based post-processing. (a) Input image. (b) Recovered image by our method. (c) Estimated support. (d) Result after post-processing. (Subtle difference between the two results of (b) and (d) is better noticed in the electronic version.)

9 Simulations and Experiments

9.1 Comparison with Low-Rank Matrix Recovery with Known Support

Here we conduct simulations to study the working range of our method. We corrupt different percentages

of a clean texture (without deformation) with gross errors and see how much corruption we can handle for different types of corruption (random, one large block, and random small blocks).

To demonstrate the importance of the sparse prior on recovering natural low-rank textures and the effectiveness of the proposed solution, we compare our method with the state-of-the-art low-rank matrix completion or recovery method: RPCA^[20] (the objective function is $\min_{\mathbf{A}, \mathbf{E}} \|\mathbf{A}\|_* + \alpha \|\mathbf{E}\|_1$ s.t. $P_{\Omega}(\mathbf{D}) = P_{\Omega}(\mathbf{A} + \mathbf{E})$, we set $\alpha = 0$ in this experiment and will further compare the algorithms with sparse prior $\mathbf{E}$ in next simulation). This is also the method which some of the latest texture completion methods such as LRTC^[42] are based upon.

In this simulation, we restrict to the simpler case that support Ω is precisely known (hence RPCA is equivalent to matrix completion). The ratio $p/(m \times n)$ between the number of corrupted/missing entries (pixels) and the number of entries in the matrix is denoted by “SR” (sampling ratio). We use the normalized resi-

dual $\|\mathbf{I}^* - \mathbf{I}^{\text{gt}}\|_F / \|\mathbf{I}^{\text{gt}}\|_F$ to measure the recovery error, where $\mathbf{I}^*$ is the recovered image, and $\mathbf{I}^{\text{gt}}$ is the ground truth.

The parameter setting for our algorithm and RPCA is $\lambda = 0.001, \alpha = 0$. We use DCT as the basis in all simulations and experiments unless otherwise stated. We have tested three kinds of corruptions: uniform random corruptions, one disk corruption, and random block corruptions on three representative low-rank textures: a synthesized checkerboard image and two real texture images. The checkerboard is of precise rank 2 and the other two are full-rank but approximately low-rank.

We test the two algorithms for input images with SR growing from 0% to 99.9%. Fig.8 shows the quantitative comparison between the two methods with different levels and kinds of corruptions. Fig.9 further shows some qualitative results that compare the two methods.

These results clearly show that the sparse prior indeed helps low-rank texture completion in all cases.

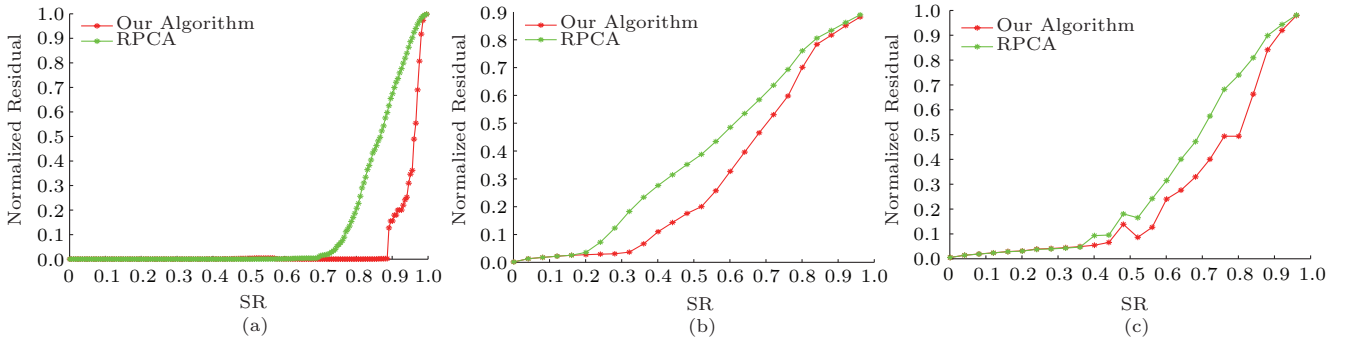


Fig.8. Quantitative comparisons with RPCA. This figure shows normalized residual error versus the sampling rate of our method and RPCA^[20] on different kinds of corruptions. (a) Random. (b) One disk. (c) Random blocks.

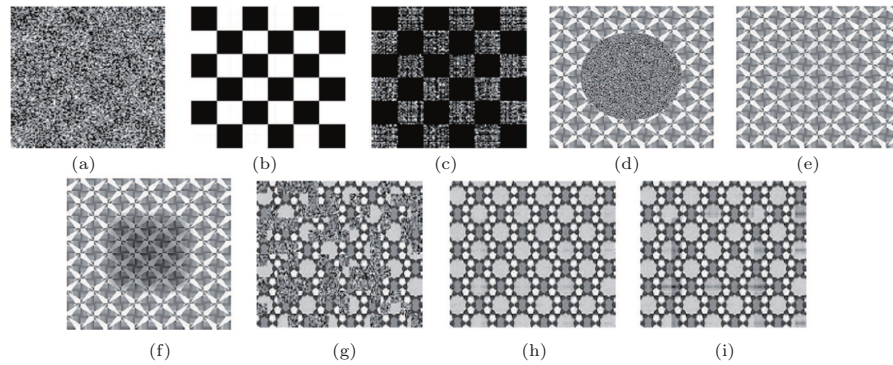


Fig.9. Qualitative comparisons with RPCA on simulated images. (a) Random input, a checkerboard texture with 91% random corruption. (b) Random proposed. (c) Random RPCA. (d) Disk input, a real texture with 30% corruption on a disk like region. (e) Disk proposed. (f) Disk RPCA. (g) Block input, a real texture with 40% corruption by random blocks. (h) Block proposed. (i) Block RPCA.

9.2 Comparison with Low-Rank Matrix Recovery with Support Unknown

This experiment gives comparison results with RPCA and LRR (Low-Rank Representation). The main difference with the previous experiment in Subsection 9.1 is that the support of corruption is unknown this time. We use the same settings as in the last experiment to generate test images but set the input support $\Omega = \mathbf{1}$ for all the three algorithms. The parameter setting of our algorithm and RPCA is $\lambda = 0.001$, $\alpha = 0.85$. And for LRR algorithm^[27], following their paper, we use the data matrix $\mathbf{D}$ as dictionary. The objective function is $\min_{\mathbf{A}, \mathbf{E}} \|\mathbf{A}\|_* + \alpha \|\mathbf{E}\|_1$ s.t. $\mathbf{D} = \mathbf{D}\mathbf{A} + \mathbf{E}$. We use their default parameters with $\alpha = 1$.^②

We test the three algorithms with SR of test images growing from 0% to 99.9%. Fig.10 shows quantitative comparison results of the three methods with different levels and kinds of corruptions. The result shows that the proposed algorithm can recover low-rank texture better than RPCA and LRR most of the time.

Table 1 gives the average computation time and the average iteration number of the three methods corresponding to Fig.10. The MATLAB implementation of our method requires about 1 second to repair a 120×120 image and 10.56 seconds for a 300×300 image on a machine with a 2.30 GHz processor and 8 GB of RAM. We can see that although our algorithm has the best repairing result, it is slower than RPCA and LRR. The most time-consuming task for all the three algorithms is the SVD decomposition in each iteration; thus the iteration number determines the time cost.

Table 1. Average Computation Time and Iteration Number

Method	Random 120 × 120		One Disk 300 × 300		Blocks 160 × 120	
	Time	Iter	Time	Iter	Time	Iter
	(s)	Num	(s)	Num	(s)	Num
RPCA	0.334	52.0	2.801	60.25	0.437	54.68
LRR	0.975	123.2	7.059	126.30	1.761	122.60
Proposed	1.045	114.6	10.560	137.30	1.522	124.30

Note: Iter Num: iteration number.

9.3 Handling Error $\mathbf{E}$ with MRF

In this subsection, we show that the MRF strategy (21) not only can handle contiguous support better than hard threshold, but also works on sparse support after setting the mutual interaction control $\gamma = 0$.

Then we give a comparison result of the proposed support estimation method in Section 6 with our old ECCV paper^[32].

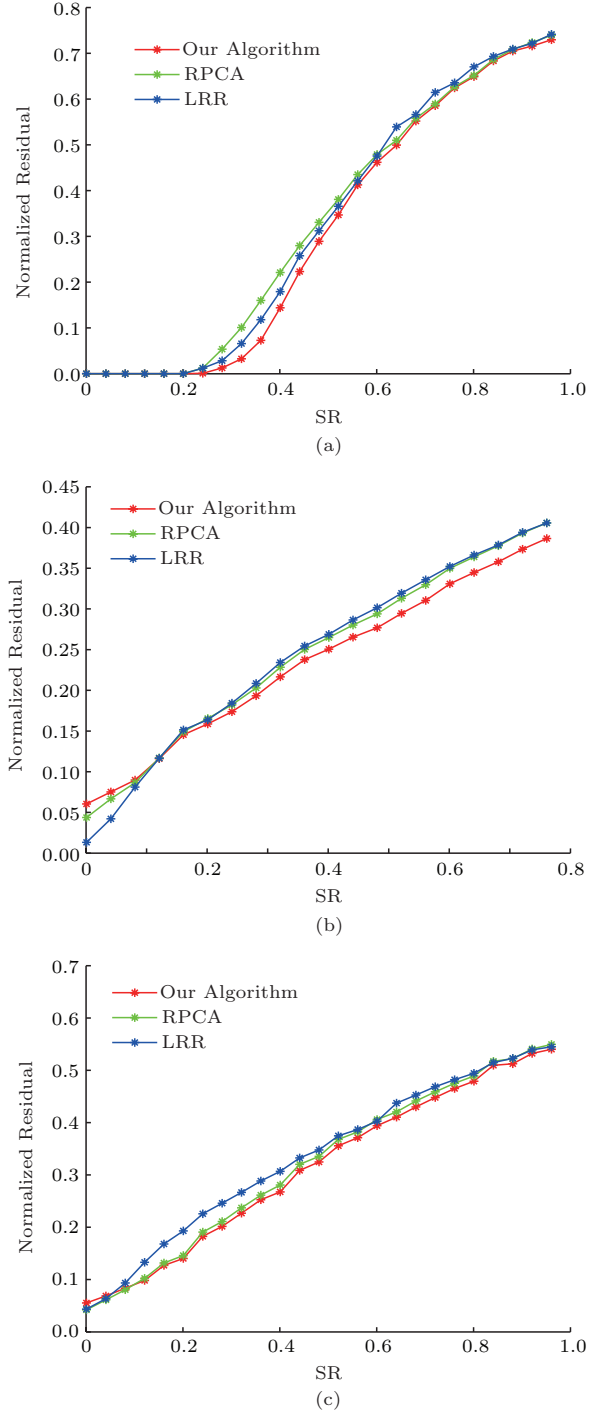


Fig.10. Comparison results with RPCA and LRR. This figure shows normalized residual error versus the sampling rate of our algorithm, RPCA^[26] and LRR^[27] on different kinds of corruptions. (a) Random. (b) One disk. (c) Random blocks.

^②Thank Professor Liu *et al.* for sharing their code^[27] with us.

This experiment compares MRF with hard threshold on two types of support. The parameter setting for MRF is $\gamma = 0, \beta = 0.3$ for random support and $\gamma = 5, \beta = 0.015$ for continuous support. For hard-thresholding, $\Omega^i = |\mathbf{E}| > \beta$. The iteration will be terminated when $\|\Omega^{i+1} - \Omega^i\|_1 < mn/100$. The input error support $\Omega^0 \in \mathbb{R}^{m \times n}$ is set to be all zero. We can see from Fig.11 that MRF is better than hard-thresholding with contiguous support and has exactly the same result with hard-thresholding when $\gamma = 0$.

We have proposed an unified support estimation method in Section 6. Compared with the simple MRF used in [32], this new method handles images corrupted with both random and continuous support better. Fig.12 shows an example. The parameters used for [32] are $\gamma = 5, \beta = 0.015$. It could be seen that the repairing result of [32] has some scratches and small unrepaired region left. And the new method fixes these problems by more accurate support estimate.

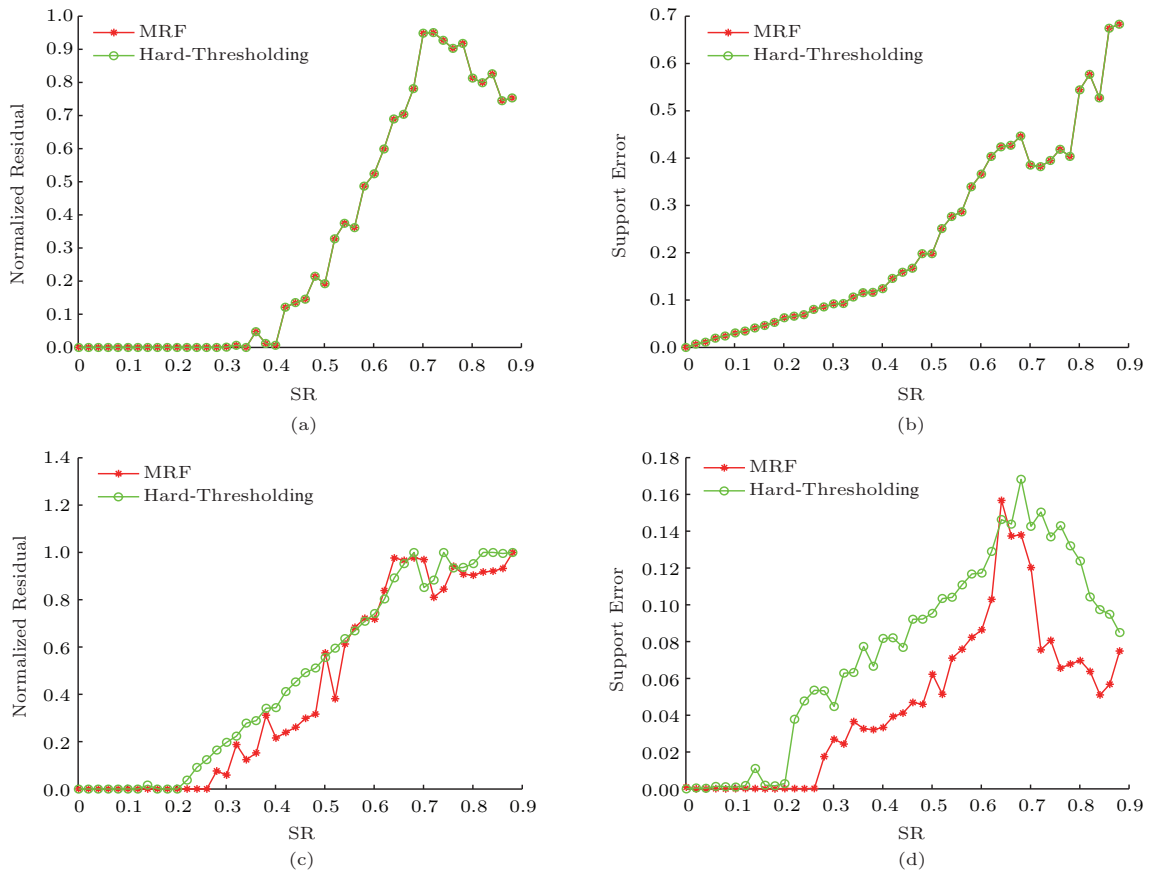


Fig.11. Qualitative comparisons with hard-thresholding on a checkerboard image. (a) Image error of random corruption. (b) Support error of random corruption. (c) Image error of disk corruption. (d) Support error of disk corruption. Images in (a) and (c) show the normalized residual error of recovered image versus the sampling rate of MRF and hard-thresholding. Images in (b) and (d) show the normalized error of estimated support versus the sampling rate of MRF and hard-thresholding.

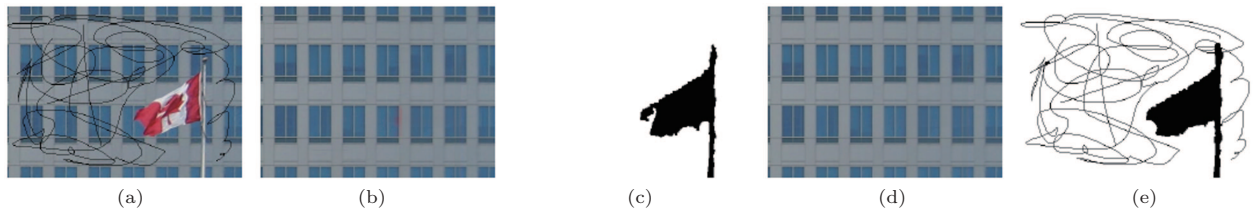


Fig.12. Comparison with [32]. (a) Input damaged image. (b) Repairing result by [32]. (c) Reconstructed support by [32]. (d) Repairing result by proposed method. (e) Reconstructed support by proposed method.

9.4 Comparison with Impulse Noise Removal Method

Here we introduce another application related to image inpainting — impulsive noise removal. Removing impulse noise from images is different from removing Gaussian noise as the values of damaged pixels contain no information of the truth at all. Thus, removing impulse noise is essentially an image inpainting problem.

Besides the visual comparison of the results, the peak signal to noise ratio (PSNR) measurement is used to quantitatively evaluate the quality of the restoration results. Given an image $\mathbf{I} \in [0, 255]^{m \times n}$, the PSNR of its estimate $\mathbf{I}^*$ is defined as

$$PSNR(\mathbf{I}^*, \mathbf{I}) = 10 \log_{10} \frac{255^2}{\frac{1}{m \times n} \sum_{i,j} (\mathbf{I}_{i,j}^* - \mathbf{I}_{i,j})^2}.$$

And we also use the fraction of incorrectly estimated support pixels to evaluate the support estimation. It is computed using the binary exclusive (XOR) operation with the ground truth support, which produces a logical value equal to 1 at the support pixels estimated incorrectly, and 0 otherwise. Hence, the sum over all the pixels of the XOR operation is a measure of the errors in the estimate of support Ω .

Because impulse noise is randomly distributed over the image, we choose parameters for MRF for random support in our method. All parameters remain the same as previous simulations and experiments, which means $\lambda = 0.001, \alpha = 0.85$. Fig.13 compares the estimate obtained using the proposed method with those obtained using fast two-phase deblurring with TV^[29] and K-ALS^[30] for 45% of the pixels missing.

To give some quantitative results, we further test the three algorithms using input images generated by adding random-valued impulse noise with different levels ($SR = 15\%, 30\%, 45\%, 60\%, 75\%$) onto the textures shown in Fig.13. The PSNR values of the restoration results by these three methods are summarized in Table 2. K-ALS^[30] is found not to provide a good denoising result outside the ranges of fractions of missing pixels, as reported by its authors. We can see from this table that our method does work very well on textures which meet our low rank assumption.

9.5 Comparison with Image Completion Systems

In this experiment, we have conducted a series of comparative tests between our method and a few recent image completion methods such as the Shift-Map

(SM) method^[19] and some highly engineered commercial systems: PatchMatch (PM) used by Photoshop ([18, 46]), and Image Completion with Structure Propagation (SP) developed by Microsoft Research Asia^[5]. These methods all share the spirit of sample-based texture synthesis: they stitch sampled local patches together to ensure certain global consistency. As these methods rely mostly on local statistics and structures, they tend to work on natural images or random textures too while our method does not, as shown in Fig.14 and Fig.15. However, this experiment aims to demonstrate significant advantages of our method on completing or repairing regular or near-regular low-rank patterns. The reason is partially because these methods normally do not or cannot exploit global structural information about the textures.

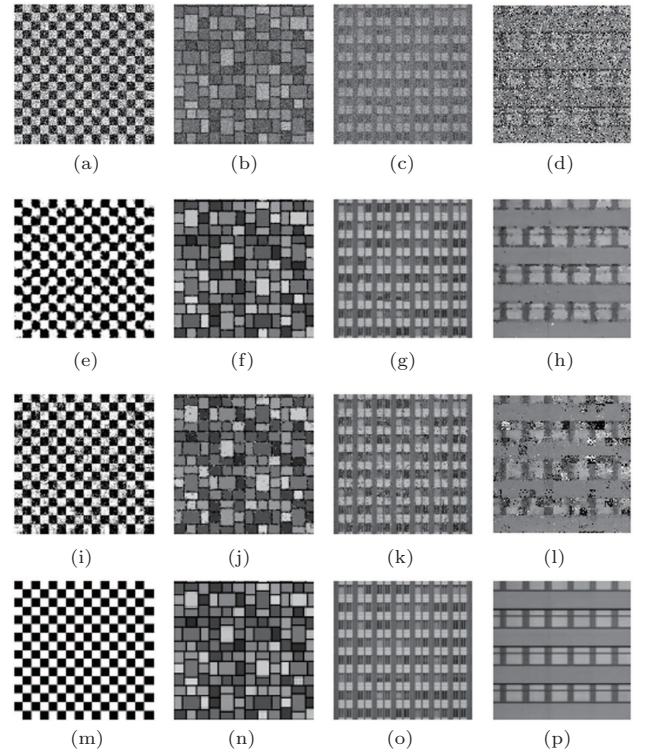


Fig.13. Random valued impulse noise removal. The input images are contaminated by random impulse noise with $SR = 45\%$. (a) Input 1. (b) Input 2. (c) Input 3. (d) Input 4. (e) TV 1. (f) TV 2. (g) TV 3. (h) TV 4. (i) KALS 1. (j) KALS 2. (k) KALS 3. (l) KALS 4. (m) Proposed 1. (n) Proposed 2. (o) Proposed 3. (p) Proposed 4.

Unlike our method, these image completion systems typically require users to mark out rather precisely the to-be-completed region or regions (marked out as red regions in this paper), and even to provide additional information about the structures to be recovered (such as that required by the SP method). Notice that if the

Table 2. Comparison for Impulse Noise Removal

SR	Method	Checkerboard		Pattern		Building1		Building2	
		PSNR	Ω	PSNR	Ω	PSNR	Ω	PSNR	Ω
		(dB)	Err(%)	(dB)	Err(%)	(dB)	Err(%)	(dB)	Err(%)
15	TV ^[29]	21.29	2.73	30.71	1.88	28.26	5.76	27.67	5.37
	KALS ^[30]	19.46	8.20	18.65	6.11	25.44	7.12	19.18	11.18
	Proposed	107.50	0.84	44.58	1.66	34.11	2.62	40.41	1.89
30	TV ^[29]	16.53	7.82	26.24	4.31	26.38	8.07	25.91	8.01
	KALS ^[30]	12.70	15.97	16.01	12.11	19.52	13.68	17.66	17.93
	Proposed	75.51	1.64	38.85	3.49	33.04	4.33	38.45	3.58
45	TV ^[29]	14.03	15.57	22.61	8.28	24.56	11.17	23.81	10.76
	KALS ^[30]	1.72	30.42	1.10	20.87	6.03	22.87	13.55	21.02
	Proposed	66.06	2.51	30.27	6.71	31.56	6.43	36.41	5.20
60	TV ^[29]	11.10	31.70	18.05	15.90	22.03	16.08	21.57	16.03
	KALS ^[30]	-3.70	42.37	3.31	28.67	1.94	30.28	5.70	20.70
	Proposed	17.08	7.83	20.69	14.46	29.43	8.84	32.90	8.15
75	TV ^[29]	8.26	47.91	14.26	26.54	18.30	24.70	19.07	23.25
	KALS ^[30]	4.54	28.89	8.34	23.75	7.47	23.83	7.11	19.01
	Proposed	7.01	23.54	14.34	28.76	25.65	12.15	23.66	15.34

Note: *SR* indicates the fraction of corrupted pixels.

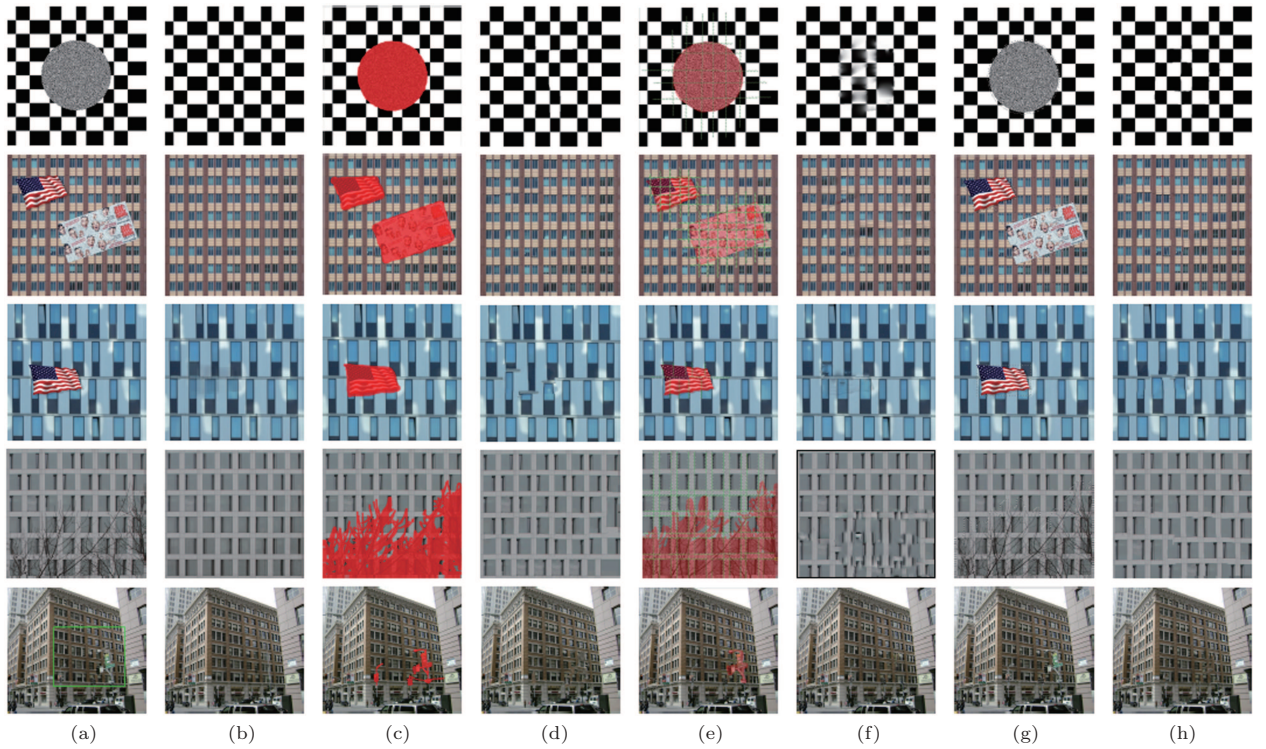


Fig.14. Comparison results with Shift-Map^[19], SP^[5] and Photoshop^[18]. (a) Input and (b) result of our method. (c) Input and (d) result of Shift-Map. (e) Input and (f) result of SP. (g) Input and (h) result of Photoshop. For rectified textures, our method can take the whole image as input, while in the deformed case, a green window indicates our input.

support of the corrupted region is small, our method does not even need information about the support at all. For our method, we only have to specify a large window that contains some corrupted regions (see the fifth row in Fig.14 for example). Nevertheless, if the

corruption is somewhat too large (say, larger than 25% of the input region), our method can also take a rough support for the corrupted region Ω^0 as part of the input (marked with red curves in the input images shown in Fig.15). The repairing result can be significantly



Fig.15. Realistic repairing results. The green windows in input images denote the input windows to our system, and red contours denote the initial error supports to our system. (a) Input 1. (b) Input 2. (c) Input 3. (d) Input 4. (e) Input 5. (f) Output 1. (g) Output 2. (h) Output 3. (i) Output 4. (j) Output 5. (k) Support 1. (l) Support 2. (m) Support 3. (n) Support 4. (o) Support 5.

improved. As our method is inherently robust, the provided support does not need to be so precise.

In addition, these methods are not designed at all to handle textures that are deformed from a purely homogeneous texture or regular pattern. For instance, they cannot handle images with perspective deformation. In our method, we can recover this kind of images by Algorithm 3. For color images, three matrices associated with the three RGB channels are repaired independently. In this experiment, all parameters remain the same as previous simulations and experiments.

In Fig.14, we present comparisons with Shift-Map method^[19], PatchMatch (PM) ([18, 46]) and structure propagation (SP)^[5] on five different images: a simulated non-uniform low-rank texture, a uniform building facade, a somewhat less uniform building facade, a real building with tree branch occlusion and a real building facade image with perspective deformation, which correspond to the five rows in Fig.14, respectively. As we see, these three methods all have trouble in correctly or accurately recovering the structural patterns of the original images.

9.6 Real Examples of Image Repairing

In this subsection, we test the efficacy of our algorithm on natural images belonging to various categories. Besides some examples where our algorithm works very well, we also present some cases that are particularly challenging where our algorithm succeeds only to some extent, and some examples where it fails completely. We believe that from these examples, we may gain a better understanding of both the strengths and the limitations of our algorithm. To show how our method can handle realistic image repairing tasks, Fig.15 gives several interesting examples produced by our method on real urban scene images. The process to generate these results is exactly the same as that in previous comparison experiments, which means $\lambda = 0.001$, $\alpha = 0.85$.

9.6.1 Curved Surface Cases

The current algorithm is only a basic version and its capability is still limited, especially when we try to apply it to cases where the assumptions are not fully met. Through the remainder of this subsection, we will discuss some of the limitations, as well as potential ex-

tensions that make it work better in some of the more challenging cases. Fig.16 shows an extension to images with low rank texture on a generalized cylindrical surface. This extension generalizes our techniques from planar surfaces to a much larger class of important 3D surfaces. To handle such generalized cylindrical surfaces, we only need a more complicated parametric model for τ than plane cases, which parameterize the unknown camera pose and surface shape. For details, please refer to [45].

9.6.2 Challenges and Limitations

As our algorithm is designed to be robust to sparse errors, sometimes it may tend to make correction to (sparse) boundaries if the error tolerance is too low. Fig.17 shows such cases.

For the image with pink dots that we used in the experiment of Fig.15, we have changed the parameter β for random corruption from 0.3 to 0.02. The result changes a lot compared with Fig.15(a). Fig.17(d) shows

a similar example. Our algorithm over-corrects around edges in these two results.

Sometimes, an input image contains multiple regions, each of which might be distorted differently. In this case, we need to deal with them one after another in order to obtain satisfactory results. Fig.18 shows such an example. We run the proposed algorithm twice on two different regions respectively. The last image is the final result.

9.6.3 Expected Failures

Although our algorithm works very robustly under very broad conditions and for a wide range of near-regular textures, it may fail to hallucinate the missing part correctly if the conditions are too challenging or the assumptions are violated. Fig.19 shows some of the cases. Fig.19(a) shows the limitations of the percentage of the support region in the input low-rank texture. When the occluded region is too large, our algorithm tends to converge at an optimal solution with

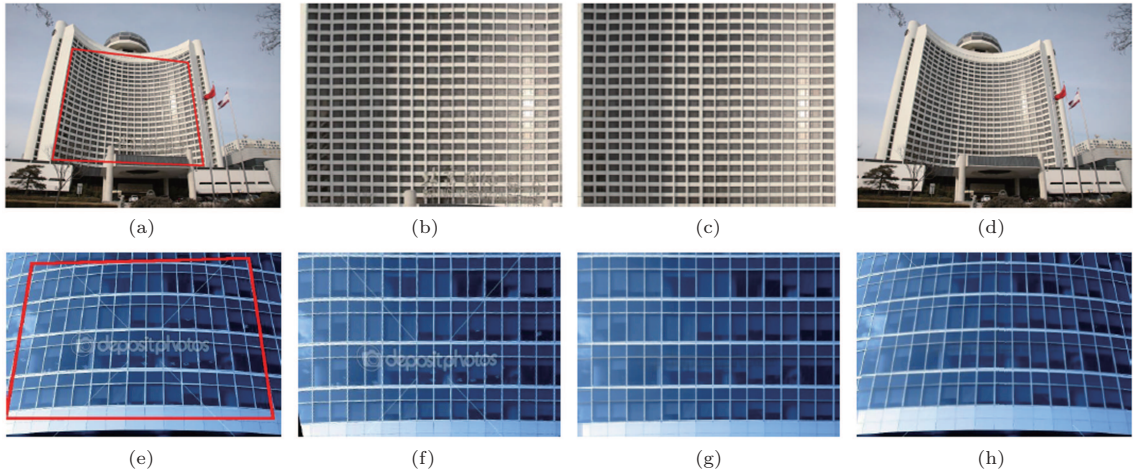


Fig.16. Curved low-rank texture. (a) Input 1. (b) Rectified 1. (c) Repaired 1. (d) Output 1. (e) Input 2. (f) Rectified 2. (g) Repaired 2. (h) Output 2. (a) and (e) are the input images, and windows with red border are the original input. (b) and (f) are the rectified low-rank textures (after deformation undone by our algorithm). (c) and (f) are the repaired low-rank textures. (d) and (g) are the repaired textures mapping back to the input images.

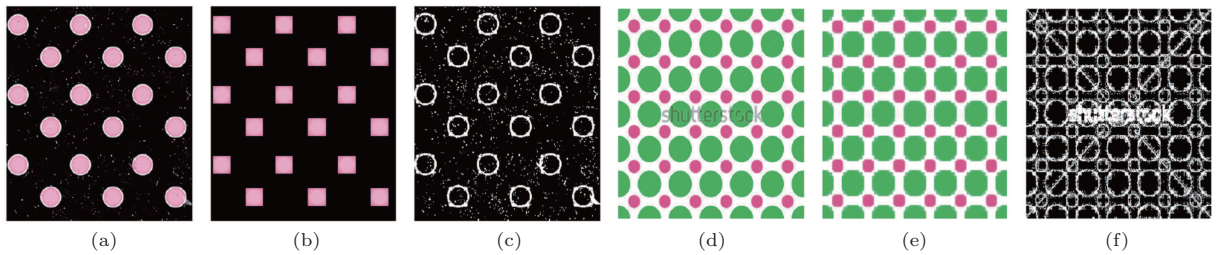


Fig.17. Over-correction cases. Our algorithm over-corrects around edges in the input image. (a) Input 1. (b) Output 1. (c) Support 1. (d) Input 2. (e) Output 2. (f) Support 2.

dark inpainting result. The repairing result is limited by the nuclear convex surrogate. This drawback would be overcome if we can handle low-rank minimization better. The second example is a facade with not only occlusion but also severe reflections. The repairing result is reasonable but not perfect. Only low rank is not enough to handle these challenging cases. We need more information such as more input images to make a good recovery.

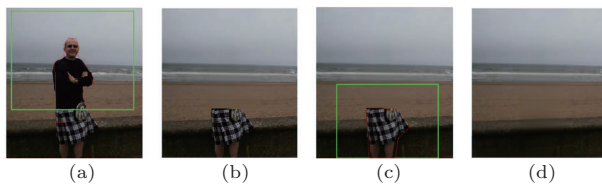


Fig.18. Two regions with different distortion. (a) Input image. (b) Result after our method applied to the top-half. (c) Applying our method to the bottom half. (d) Final result.

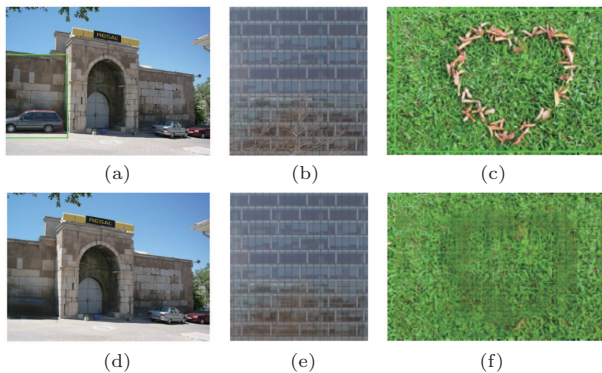


Fig.19. Some failure cases. (a) Too much occlusion. (b) Large occlusion and reflection. (c) Random (high-rank) textures.

As mentioned earlier in Section 2, our algorithm is not designed to work on random textures occurring in nature, such as Fig.19(c). Although there has been work in the literature showing that it is possible to repair a reasonable approximation of the lawn based on the statistical properties of the random texture, our algorithm is certainly not designed to handle such cases. It is effective for regular symmetric textures, but not for random textures which normally have high rank as matrices.

10 Conclusions

In this paper, we showed how to harness both low-rank and sparse structures in regular or near-regular textures for image completion. Our method leverages the new convex optimization for low-rank and sparse

signal recovery and can automatically correctly repair the global structure of a corrupted texture, even without precise information about the regions to be completed. Through extensive simulations, we showed our method can complete and repair textures corrupted by errors with both random and contiguous supports better than existing low-rank matrix recovery methods. Through experimental comparisons with existing image completion systems (such as Photoshop), our method demonstrates significant advantage over local patch based texture synthesis techniques in dealing with large corruption, non-uniform texture, and large perspective deformation.

Acknowledgement We thank Dr. Xin Tong for the valuable suggestions and comments for this paper.

References

- [1] Efros A A, Leung T K. Texture synthesis by non-parametric sampling. In *Proc. the 7th IEEE International Conference on Computer Vision (ICCV)*, September 1999, pp.1033-1038.
- [2] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer. In *Proc. the 28th SIGGRAPH*, Aug. 2001, pp.341-346.
- [3] Liang L, Liu C, Xu Y, Guo B, Shum H. Real-time texture synthesis by patchbased sampling. *ACM Transactions on Graphics (TOG)*, 2001, 20(3): 127-150.
- [4] Komodakis N, Tziritas G. Image completion using global optimization. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2006, pp.442-452.
- [5] Sun J, Yuan L, Jia J, Shum H Y. Image completion with structure propagation. *ACM Transactions on Graphics (TOG)*, 2005, 24(3): 861-868.
- [6] Criminisi A, Pérez P, Toyama K. Object removal by exemplar-based inpainting. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003, pp.721-728.
- [7] Liu Y, Lin W C, Hays J. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics (TOG)*, 2004, 23(3): 368-376.
- [8] Bertalmio M, Sapiro G, Caselles V, Ballester C. Image inpainting. In *Proc. the 27th SIGGRAPH*, July 2000, pp.417-424.
- [9] Bertalmio M, Bertozzi A L, Sapiro G. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2001, pp.355-362.
- [10] Oliveira M M, Bowen B, McKenna R, Chang Y. Fast digital image inpainting. In *Proc. the International Conference on Visualization, Imaging and Image Processing (VIIP)*, September 2001, pp.261-266.
- [11] Levin A, Zomet A, Weiss Y. Learning how to inpaint from global image statistics. In *Proc. the 9th IEEE International Conference on Computer Vision (ICCV)*, October 2003, p.305.

- [12] Mairal J, Elad M, Sapiro G. Sparse representation for color image restoration. *IEEE Trans. Image Processing (TIP)*, 2007, 17(1): 53-69.
- [13] Fadili M J, Starck J, Murtagh F. Inpainting and zooming using sparse representations. *The Computer Journal*, 2009, 52(1): 64-79.
- [14] Elad M, Starck J, Querre P, Donoho D. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied Computational Harmonic Analysis*, 2005, 19(3): 340-358.
- [15] Bugeau A, Bertalmio M, Caselles V, Sapiro G. A comprehensive framework for image inpainting. *IEEE Trans. Image Processing (TIP)*, 2010, 19(10): 2634-2645.
- [16] Kwatra V, Schödl A, Essa I, Turk G, Bobick A. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG)*, 2003, 22(3): 277-286.
- [17] Hays J, Efros A A. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 2007, 26(3): Article No. 4.
- [18] Barnes C, Shechtman E, Finkelstein A, Goldman D B. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 2009, 28(3): Article No. 24.
- [19] Pritch Y, Kav-Venaki E, Peleg S. Shift-map image editing. In *Proc. the 12th International Conference on Computer Vision (ICCV)*, October 2009, pp.151-158.
- [20] Candès E, Li X, Ma Y, Wright J. Robust principal component analysis? *Journal of the ACM*, 2011, 58(3): Article No. 11.
- [21] Zhou T Y, Tao D C. GoDec: Randomized lowrank and sparse matrix decomposition in noisy case. In *Proc. the 28th International Conference on Machine Learning (ICML)*, Jun. 28-Jul. 2, 2011, pp.33-40.
- [22] Bao B, Liu G, Xu C, Yan S. Inductive robust principal component analysis. *IEEE Trans. Image Processing (TIP)*, 2012, 21(8): 3794-3800.
- [23] Zhang Z, Yan S, Zhao M. Similarity preserving low-rank representation for enhanced data representation and effective subspace learning. *Neural Networks*, 2014, 53: 81-94.
- [24] Peng Y, Ganesh A, Wright J, Xu W, Ma Y. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2012, 34(11): 2233-2246.
- [25] Zhang Z, Yan S, Zhao M, Li F. Bilinear low-rank coding framework and extension for robust image recovery and feature representation. *Knowledge-Based Systems*, 2015, 86: 143-157.
- [26] Zhang Z, Liu C, Zhao M. A sparse projection and low-rank recovery framework for handwriting representation and salient stroke feature extraction. *ACM Trans. Intelligent Systems and Technology*, 2015, 6(1): 9:1-9:26.
- [27] Liu G, Lin Z, Yan S, Sun J, Yu Y, Ma Y. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2013, 35(1): 171-184.
- [28] Gonzalez R C, Woods R E, Eddins S L. Digital Image Processing Using MATLAB (2nd edition). Gatesmark Publishing, 2009.
- [29] Cai J, Chan R H, Nikolova M. Fast two-phase image deblurring under impulse noise. *Journal of Mathematical Imaging and Vision*, 2010, 36: 46-53.
- [30] Wang Y, Szlam A, Lerman G. Robust locally linear analysis with applications to image denoising and blind inpainting. *SIAM Journal on Imaging Sciences*, 2013, 6(1): 526-562.
- [31] Yan M. Restoration of images corrupted by impulse noise and mixed Gaussian impulse noise using blind inpainting. *SIAM Journal on Imaging Sciences*, 2013, 6(3): 1227-1245.
- [32] Liang X, Ren X, Zhang Z, Ma Y. Repairing sparse low-rank texture. In *Proc. ECCV*, October 2012, pp.482-495.
- [33] Candès E J, Recht B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 2009, 9(6): 717-772.
- [34] Chandrasekaran V, Sanghavi S, Parrilo P A, Willsky A S. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 2011, 21(2): 572-596.
- [35] Cai J, Candès E, Shen Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 2010, 20(4): 1956-1982.
- [36] Toh K, Yun S. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 2010, 6(3): 615-640.
- [37] Ganesh A, Lin Z, Wright J, Wu L, Chen M, Ma Y. Fast algorithms for recovering a corrupted low-rank matrix. In *Proc. the 3rd International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, December 2009, pp.203-216.
- [38] Lin Z, Chen M, Wu L, Ma Y. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. UIUC Technical Report, UILU-ENG-09-2215. arxiv: 1009.5055, 2009. <http://arxiv.org/abs/1109.0367>, Apr. 2016.
- [39] Bertsekas D P. Nonlinear Programming (2nd edition). Athena Scientific, 2004.
- [40] Yang J, Yuan X. Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of Computation*, 2013, 82(281): 301-329.
- [41] Lin Z, Liu R, Su Z. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in Neural Information Processing Systems*, Shawe-Taylor J, Zemel R S, Bartlett P L (eds.), 2011, pp.612-620.
- [42] Liu J, Musialski P, Wonka P, Ye J. Tensor completion for estimating missing values in visual data. In *Proc. the 12th IEEE International Conference on Computer Vision (ICCV)*, Sept. 27-Oct. 4, 2009, pp.2114-2121.
- [43] Zhou Z, Wagner A, Mobahi H, Wright J, Ma Y. Face recognition with contiguous occlusion using Markov random fields. In *Proc. the 12th IEEE International Conference on Computer Vision (ICCV)*, Sept. 29-Oct. 2, 2009, pp.1050-1057.
- [44] Zhang Z, Liang X, Ganesh A, Ma Y. TILT: Transform-invariant low-rank textures. *International Journal of Computer Vision (IJCV)*, 2012, 99(1): 1-24.
- [45] Zhang Z, Liang X, Ma Y. Unwrapping low-rank textures on generalized cylindrical surfaces. In *Proc. the International Conference on Computer Vision (ICCV)*, November 2011, pp.1347-1354.

- [46] Barnes C, Shechtman E, Goldman D B, Finkelstein A. The generalized Patch-Match correspondence algorithm. In *Proc. the 11th European Conference on Computer Vision (ECCV)*, September 2010, pp.29-43.



Xiao Liang is currently a Ph.D. candidate in computer science and technology at Institute for Advanced Study of Tsinghua University, Beijing. She received her B.E. degree in electronic engineering from Tsinghua University, Beijing, in 2008. During her study, she interned at Microsoft Research Asia, Beijing, for over four years. Her research interests include texture processing, 3D computer vision and sparsity, and low-rank matrix recovery.



Xiang Ren is a fourth-year Ph.D. candidate in computer science, University of Illinois at Urbana-Champaign. He received his B.E. degree from CS and Chu Kochen Honors Program, Zhejiang University, Hangzhou, in 2012. During his undergraduate, he interned at Microsoft Research Asia, Beijing, for over one year. He enjoys discovering principles and developing computational models for knowledge acquisition from text data and structure analysis on linked data. He likes to design data-driven methods to discover semantic structures (e.g., entities, scientific concepts, clusters, relationships) from massive text corpora.



Zhengdong Zhang is currently a third-year Ph.D. candidate at the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge. He is working with Prof. Vivienne Sze on energy efficient vision systems. Before he joined MIT, he worked at Microsoft Research Asia, Beijing, with Prof. Yi Ma on 3D reconstruction and rank minimization. He finished his undergraduate study at Tsinghua Xuetang Special Pilot CS Class (Yao Class) supervised by Turing Award Winner Prof. Andrew Chi-Chih Yao in Institute of Interdisciplinary Information Sciences, Tsinghua University, Beijing, in 2011. His research interest spans the area of sparsity, low-rank matrix recovery, symmetry/regularity of textures, 3D computer vision, computational photography, object detection and video compression. His current research focuses on the design of energy-efficient vision systems.



Yi Ma is currently the executive dean of School of Information Science and Technology in ShanghaiTech University, Shanghai. He received his Bachelor's degree in automation and applied mathematics from Tsinghua University, Beijing, in 1995. He received his Master's degree in electrical engineering and computer sciences (EECS) in 1997, his second Master's degree in mathematics in 2000, and his Ph.D. degree in EECS in 2000, all from the University of California at Berkeley. From 2000 to 2011, he served as a tenured associated professor of the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, where he now holds an Adjunct Professorship. Since January 2009, he has served as research manager for the Visual Computing Group at Microsoft Research Asia, Beijing. From January 2012, he also holds a Guest Professorship at the University of Science and Technology of China, Hefei. He currently serves as the associate editor for the IEEE Transactions on Information Theory, the International Journal of Computer Vision (IJCV), the IMA Journal on Information and Inference, and the SIAM Journal on Imaging Sciences. Dr. Ma is a fellow of IEEE, and a member of ACM and SIAM.