Repairing Sparse Low-Rank Texture

Xiao Liang[1,2], Xiang Ren[2], Zhengdong Zhang[2], and Yi Ma[2,3]

[1] Tsinghua University
[2] Visual Computing Group, Microsoft Research Asia
[3] Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

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 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 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. Through experimental comparisons with existing image completion systems (such as Photoshop) our method demonstrate significant advantage over local patch based texture synthesis techniques in dealing with large corruption, non-uniform texture, and large perspective deformation.

Key words: Low-Rank and Sparse Matrix Recovery, Texture Completion, 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 values 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].

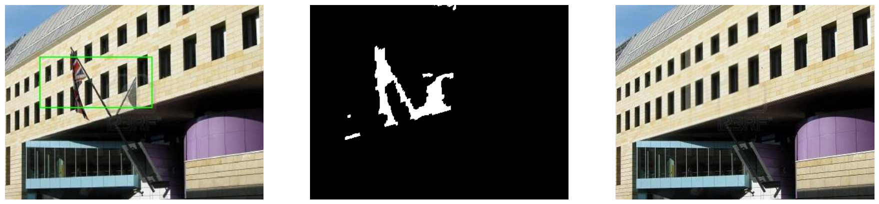


Fig. 1. A sample result produced by our method. Left column: input image, where the green window denotes the input window to our system. Middle column: estimated support of corruption. Right column: repaired image.

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. Nevertheless, 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.

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. Actually, almost all image completion and inpainting methods need to know the exact support of the missing pixels, and many are very sensitive to whether the support is given precisely (see Figure 3 for an example). This requirement has severely limited the applicability of image completion methods in practical scenarios. Very often we do not know the exact pixels in an image that need to be repaired. 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 support of the corrupted regions is only partially known or entirely unknown, we refer to the task as *image repairing*, to distinguish from the conventional image completion tasks with known support.

Contributions of This Paper. 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 Figure 1 for an example). We will show that by simultaneously enforcing the low-rank and sparse priors on a recovered image, we are able to handle 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 use generative parametric models (say stochastic PDEs) to describe the structures of the image signals. The completion processes essentially relies on the generalizability of such models and extrapolate the missing part of the image from the given one. For example, Bertalmio *et al.* [8] fills holes in an image by propagating image Laplacians in the isophote direction continuously from the exterior. Their PDE-based method has it 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.

Recently, there have also been works (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] combine geometric partial differential equation (PDEs) and patch-based texture synthesis in a variational model and performs 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 relies 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 rely 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], [16, 17, 5]. While these methods are designed to work for random textures, Liu *et al.* [7] shows 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 works done specifically for image completion. Criminisi *et al.* [6] employs an exemplar-based texture synthesis technique modulated by a unified scheme for determining the fill order of the target region. Sun *et al.* [5] utilize user interaction to specify the structure information in the image and help to do image completion. Komodakis *et al.* [4] treats 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 such patch-based techniques, including Patch Match (PM) that used by Photoshop [18], Image Completion with Structure Propagation (SP) developed by Microsoft [5], and Shift Map (SM) [19].

2 Problem Formulation and Technical Approach

In this section, we describe the mathematical model for the class of *sparse low-rank textures* considered in this paper. We formulate the objective of completing or repairing a sparse low-rank texture and show how to minimize the objective via its convex surrogate. We employ the efficient Linearized Alternating Direction Method (LADM) [20] for solving the convex optimization program.

2.1 Sparse Low-rank Texture Inpainting

Consider a 2D texture as a matrix $I \in \mathbb{R}^{m\times n}$, it is called a low-rank texture if $r \ll \min(m,n)$, where $r = \operatorname{rank}(I)$. All regular, symmetric patterns belong to this class of textures. For such textures, image completion becomes a low-rank matrix completion problem. Suppose $D \in \mathbb{R}^{m\times n}$ is the given image and Ω is the set of observed entries (pixels). The goal is is to recover an image I of the lowest possible rank that agrees with D on all the observed pixels. This completion problem is equivalent to following problem:

$$\min_{I} \ \operatorname{rank}(I), \quad \text{s.t.} \ \ I_{ij} = D_{ij}, \forall (i,j) \in \Omega, \tag{1}$$

Although this is in general an NP-hard problem, in a recent seminal paper [21], Candès *et al.* has 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:

$$\min_{I} \|I\|_*, \quad \text{s.t.} \ \ P_\Omega(I) = P_\Omega(D), \tag{2}$$

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 belong to Ω.

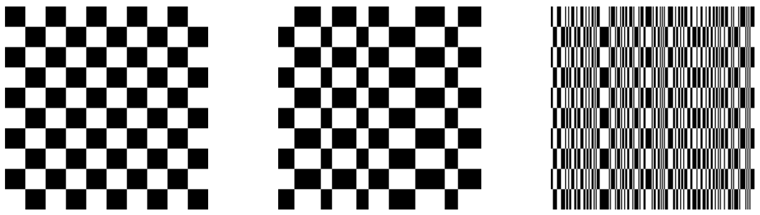


Fig. 2. An example of rank's insufficiency. These three images have exactly the same rank, but they go from purely regular, to nearly regular, and to nearly random textures.

Although being low-rank is a necessary condition for most regular, structured images, it is certainly *not sufficient*. Figure 2 shows three images that have exactly the same rank. Obviously the first two are more smooth, realistic textures than the third one. If we consider a rank-r texture as a 2D function $I(x,y)$, define on $\mathbb{R}^2$. Then $I(x,y)$ can be factored as $\sum_{i=1}^{r} g_i(x)h_i(y)$. If I represents a more realistic regular or near regular pattern, it is typically piecewise smooth. Hence, the 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).

So, if we factorize the matrix I as $I = UV^T$, where U and V can be represented as $U = B_1X_1$ and $V = B_2X_2$ for some bases (B_1, B_2). If the bases are properly chosen, both X_1 and X_2 will be sparse. Or equivalently, the matrix $W = X_1X_2^T$ will be a sparse matrix, which has the same (low) rank as I since $I = B_1WB_2^T$.

Hence, if we want the recovered image to be both low-rank and sparse (in certain transformed domain), we could modify the above convex program (2) as follow to impose additional spatial structures:

$$\min_{I,W} \|I\|_* + \lambda\|W\|_1 \quad \text{s.t.} \quad P_\Omega(I) = P_\Omega(D), \; I = B_1WB_2^T, \tag{3}$$

where λ is a weighting parameter which trades off the rank and sparsity of the recovered image. If we further assume that the bases used are orthonormal, we have $\|I\|_* = \|B_1WB_2^T\|_* = \|W\|_*$. The convex program (3) is equivalent to:

$$\min_{W} \quad \|W\|_* + \lambda\|W\|_1 \quad \text{s.t.} \quad P_\Omega(B_1WB_2^T) = P_\Omega(D). \tag{4}$$

2.2 Sparse Low-rank Texture Repairing

Almost all previous methods for image inpainting or completion (e.g., [8], [12], [13], etc.) need information about the support Ω of the corrupted regions. This information is usually obtained through 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 E, and solve the following optimization:

$$\min_{W} \quad \|W\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 \quad \text{s.t.} \quad P_\Omega(B_1 W B_2^T + E) = P_\Omega(D). \tag{5}$$

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 [22], the above convex program will decompose the image into a low-rank component and a sparse one. Of course, the nonzero entries in estimated E can help us to further refine the support Ω. For instance, we could simple set:

$$\text{supp}(E) \doteq \begin{cases} (i,j) \in \Omega^c, & |E_{ij}| > \varepsilon; \\ (i,j) \in \Omega, & |E_{ij}| \leq \varepsilon. \end{cases} \tag{6}$$

for some threshold $\varepsilon > 0$. Or we could estimate the support of E using more sophisticated model to encourage additional structures such as spatial continuity. We will discuss one such option in Section 2.4.

We could further iterate between the image completion and support estiomation:

$$\begin{aligned} (W^i, E^i) &= \text{argmin}_{W,E} \|W\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 \\ &\qquad\qquad \text{subject to} \quad P_{\Omega^i}(B_1 W B_2^T + E) = P_{\Omega^i}(D), \\ \Omega^{i+1} &= \Omega^i - \text{supp}(E^{i+1}), \end{aligned} \tag{7}$$

where α is a weighting parameter between sparsity and low-rankness.

We could iterate the above process till convergence and obtain the repaired image $I^* = B_1 W^* B_2^T$. In practice, we also notice a good side effect of adding the additional E term, we can not only obtain the support of the corrupted regions but also can reduce noise on the repaired texture image I^*.

2.3 Solution via Linearized Alternating Direction Method

Our optimization requires simultaneously minimizing the nuclear norm and 1-norm of a matrix, which has been explored in the work of [23] for data clustering. Hence we could utilize the same algorithm suggested by that work: the linearized alternating direction method with adaptive penalty[24]. This method has proven to be one of the fastest algorithms for solving various low-rank matrix completion and recovery problems. To adopt LADMAP method to our problem, we need to make our objective function separable. Thus we introduce an auxiliary variable A to replace W in the low-rank term, which turns the optimization problem into the following:

$$\min_{A,E,W} \|A\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 \quad \text{s.t.} \quad A = W, \; P_\Omega(B_1 W B_2^T + E) = P_\Omega(D). \tag{8}$$

LADM works by minimizing the augmented Lagrangian function of the above problem:

$$\begin{aligned} L_\mu(A, W, E, Y_1, Y_2) = \|A\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 + \langle Y_1, A - W\rangle + \langle Y_2, P_\Omega(B_1 W B_2^T \\ + E) - P_\Omega(D)\rangle + \tfrac{\mu}{2}\|A - W\|_F^2 + \tfrac{\mu}{2}\|P_\Omega(B_1 W B_2^T + E) - P_\Omega(D)\|_F^2, \end{aligned} \tag{9}$$

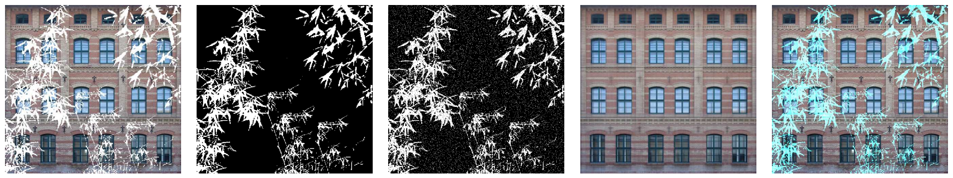


Fig. 3. Robustness to imperfect input support. This figure shows that with 5% support map corrupted, our algorithm recovers the image while the method based on low-rank matrix completion LRTC [25] fails.

with respect to the unknown variables A, W, E one at a time. Here Y_1, Y_2 are the Lagrange multipliers, $\langle\cdot,\cdot\rangle$ is the inner product, $\|\cdot\|_F$ is the Frobenius norm, and $\mu > 0$ is a penalty parameter. The LADM updates all the variables as follows:

$$\begin{aligned}
A_{k+1} &= \operatorname*{argmin}_A L_{\mu_k}(A, W_k, E_k, Y_{1k}, Y_{2k}), && (10)\\
W_{k+1} &= \operatorname*{argmin}_W L_{\mu_k}(A_{k+1}, W, E_k, Y_{1k}, Y_{2k}), && (11)\\
E_{k+1} &= \operatorname*{argmin}_E L_{\mu_k}(A_{k+1}, W_{k+1}, E, Y_{1k}, Y_{2k}), && (12)\\
Y_{1k+1} &= Y_{1k} + \mu_k \cdot P_\Omega(B_1 W_{k+1} B_2^T + E_{k+1} - D), &&\\
Y_{2k+1} &= Y_{2k} + \mu_k \cdot (A_{k+1} - W_{k+1}), &&\\
\mu_{k+1} &= \rho \cdot \mu_k, &&
\end{aligned}$$

where $\rho > 1$ is a constant. Subproblem (10) has closed-form solution:

$$A_{k+1} = \operatorname*{argmin}_A \|A\|_* + \frac{\mu_k}{2}\big\|A - W_k + \frac{1}{\mu_k}Y_{1k}\big\|_F^2 = S_{(\mu_k)^{-1}}\big(W_k - \frac{1}{\mu_k}Y_{1k}\big), \qquad (13)$$

where $\boldsymbol{S}_\mu(\cdot)$ is the singular value shrinkage operator: $\boldsymbol{S}_\varepsilon(W) = U\boldsymbol{T}_\varepsilon(\Sigma)V^T$ in which $U\Sigma V^T$ is the SVD of W and $\boldsymbol{T}_\varepsilon(x) = \operatorname{sgn}(x)\max(|x| - \varepsilon, 0)$ is the scalar shrinkage operator.

Solving subproblems (12) and (11) need extra modification on the objective function since there are linear operator on variables E and W. By [20] and [24], 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 W and E. This results in the following updating scheme:

$$W_{k+1} = \boldsymbol{T}_{\frac{\lambda}{\mu_k \eta_1}}\Big(W_k - \frac{B_1^T(P_\Omega(B_1 W B_2^T + E_k - D + \frac{1}{\mu_k}Y_{2k})B2 + W_k - A_{k+1} - \frac{1}{\mu_k}Y_{1k}}{\eta_1}\Big), \qquad (14)$$

and

$$E_{k+1} = \boldsymbol{T}_{\frac{\alpha}{\mu_k \eta_2}}\Big(E_k - \frac{P_\Omega(E + B_1 W_{k+1} B_2^T - D) + \frac{1}{\mu_k}Y_{2k}}{\eta_2}\Big), \qquad (15)$$

where $\eta_1 > 0$ and $\eta_2 > 0$ are some parameters (cf. [24]), and we set $\eta_1 = 3, \eta_2 = 3$ in our paper.

To show the advantage of adding the robust error term E for support detection, we do a comparison with another low-rank texture completion method based on matrix completion [25]. In Figure 3, we first corrupted an input low-rank image with a ground truth support Ω, then we generate a new support $\widetilde{\Omega}$

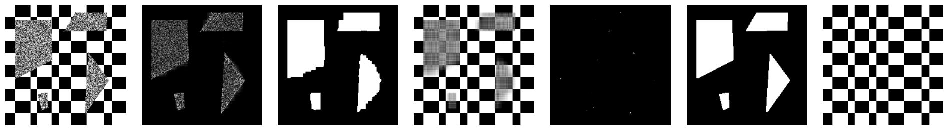

(a) Input D (b) E^1 (c) Ω^1 (d) I^1 (e) E^2 (f) Ω^2 (g) I^2

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.

by randomly switching 5% of the support Ω (from 1 to 0 or from 0 to 1). We use $\widetilde{\Omega}$ as the input support for our method and the LRTC method [25] and compare their completion results. In the comparison, we set parameters $\lambda = 0.001$, $\alpha = 0.85$ for our algorithm and use DCT as the basis for B_1 and B_2. For the LRTC method, all parameters are set as the same values as those in the public code of LRTC. The results in Figure 3 clearly show that our method can handle erroneous input support while other matrix completion methods would fail.

2.4 Handling Error E with Contiguous Support

In a real image, very often the regions we need to repair have contiguous supports – occluded or corrupted pixels are likely to be adjacent to each other in the image. The simple thresholding used in (6) 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 supp(E), as in [26]. We do not make any other assumptions about the locations, sizes, shapes, or the number of occluded regions.

Following a similar strategy in [26] to estimate a continuity-prone support, we only have to replace the support update in Eq. (7) by the following:

$$\mathrm{supp}(E) = \mathrm{argmin}_{\Omega \in \{-1,1\}^{m \times n}} \textstyle\sum_{(l,n)\in M} \gamma \mathbf{s}[i]\mathbf{s}[j] + \sum_{l \in V} \log\big(p(\mathbf{e}^i[l]|\mathbf{s}[l])\big), \quad (16)$$

with

$$\begin{aligned} \log\big(p(\mathbf{e}[l]|\mathbf{s}[l]=-1)\big) &= \begin{cases} -\log\beta, & |\mathbf{e}[l]| \le \beta; \\ \log\beta, & |\mathbf{e}[l]| > \beta. \end{cases} \\ \log\big(p(\mathbf{e}[l]|\mathbf{s}[l]=1)\big) &= \begin{cases} 0, & |\mathbf{e}[l]| > \beta; \\ \log\beta, & |\mathbf{e}[l]| \le \beta. \end{cases} \end{aligned} \quad (17)$$

where $G = (V, M)$ is a graph which represents the image domain, $\mathbf{s} = \mathrm{vec}(\Omega) \in \mathbb{R}^{mn}$, $\mathbf{e} = \mathrm{vec}(E) \in \mathbb{R}^{mn}$, $V = \{1, ..., mn\}$ denotes the set of $m \times n$ pixels and M denotes the edges connecting neighboring pixels. Here, the parameter β indicate the level of error we would accept before considering an entry of the image as occluded. The 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.

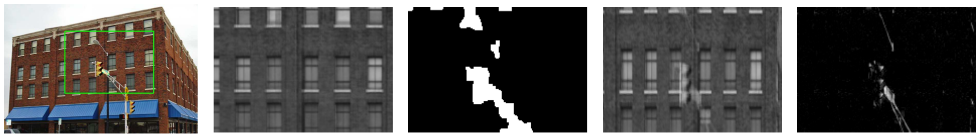

(a) Input image (b) Our result (c) Our support (d) TILT result (e) E of TILT

Fig. 5. Comparison with TILT. Notice that TILT does not (intend to) fully repair the occluded parts while our method does.

Figure 4 shows how MRF works. We used $\gamma = 0.015, \beta = 5$ and $\lambda = 0.001, \alpha = 0.85$ in the example (empirically set since they bring best performance). All other experiments use the same parametric setting unless stated otherwise.

2.5 Handling Deformed Texture *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 I agrees with the observed texture D up to certain transformation. To recover I, we then need to solve the following problem instead:

$$\min_{W,E,\tau} \|W\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 \quad \text{s.t.} \quad P_\Omega(B_1 W B_2^T + E) = P_\Omega(D \circ \tau), \tag{18}$$

where $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ belongs to a certain group of transforms, e.g., affine transform, perspective transforms, and general cylindrical transform [27].

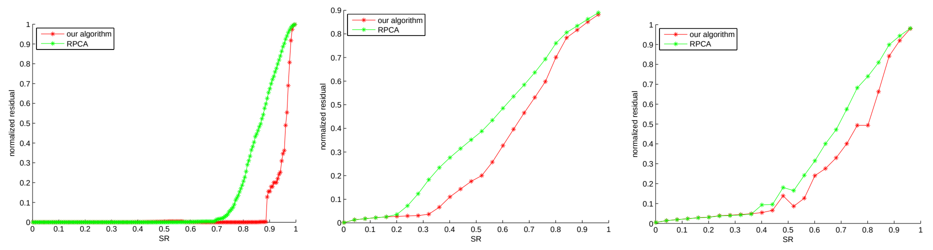
Above problem is not a convex program as the constraint is now nonlinear in the unknowns. Similar to the work of TILT by Zhang *et al.* [28], we could linearize $D \circ \tau$ at the previous τ^i as $D \circ (\tau^i + \Delta\tau) \approx D \circ \tau^i + J\Delta\tau$ where J is the Jacobian: derivative of the image with respect to the transformation parameters. Then, to handle deformation, our method can be easily modified as follows:

$$\begin{aligned} (W^{i+1}, E^{i+1}, \Delta\tau^{i+1}) &= \operatorname{argmin}_{W,E,\Delta\tau} \|W\|_* + \lambda\|W\|_1 + \alpha\|E\|_1 \\ &\qquad \text{s.t.} \quad P_{\Omega^i}(B_1 W B_2^T + E) = P_{\Omega^i}(D \circ \tau^i + J\Delta\tau), \\ \tau^{i+1} &= \tau^i + \Delta\tau^{i+1}, \\ \Omega^{i+1} &= \Omega^i \cup \operatorname{supp}(E^{i+1}). \end{aligned} \tag{19}$$

The algorithm runs in the same spirit as TILT, except that it further imposes sparsity of the recovered texture and repair the image once the locations of errors are detected. Nevertheless the improvement of our method in repairing the rectified texture is very significant. Figure 5 shows an example that compares the results of our method with the TILT code released by the authors of [28].

3 Simulations and Experiments

A. Comparison with Low-rank Matrix Recovery. Here we conduct simulations to study the working range of our method. We corrupt different per-



(a) random corruption (b) one disk (c) random blocks

Fig. 6. Quantitative comparisons with RPCA. This figure shows normalized residual error versus the sampling rate of our method and RPCA [22] on different kinds of corruptions.

centages 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, 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-art low-rank matrix completion or recovery method: Robust Principal Component Analysis (RPCA) [22] (The object function is $\min_A \|A\|_*$ s.t. $P_\Omega(A) = P_\Omega(D)$, which does not exploit the sparse prior at al-l). This is also the method which some of the latest texture completion methods such as LRTC [25] are based upon.

In this simulation, we restrict to the simpler case that the 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 residual $\|A - M\|_F / \|M\|_F$ to measure the recovery error, where A is the recovered image, and M is the ground truth.

The parameter setting for our algorithm and RPCA is $\lambda = 0.001, \alpha = 0$. We use DCT as basis in all simulations and experiments unless otherwise stat-ed. We have tested three kind 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 approxi-mately low-rank.

We test the two algorithms for input images with SR growing from 0 to 99.9%. Figure 6 shows quantitative comparison between the two methods with different levels and kinds of corruptions. Figure 7 further show some qualitative results that compare the two methods. These results clearly show that the sparse prior on the image indeed helps low-rank texture completion in all cases.

B. 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

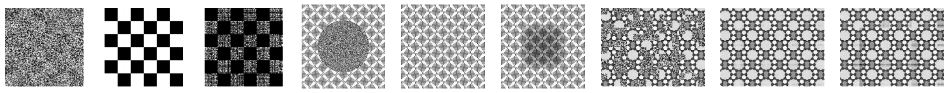


Fig. 7. Qualitative comparisons with RPCA on simulated images. First one (columns 1-3) is a checkerboard texture 91% randomly corrupted; second one (columns 4-6) is a real texture 30% corrupted on a disk like region; and third one (columns 7-9) is a real texture 40% corrupted by random blocks. In each case, the first image is input, the second is our result, and the third is result by RPCA [22].

some highly engineered commercial systems: Patch Match (PM) used by Photoshop ([18], [29]), 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. However, this experiment is 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.

Unlike our method, these image completion systems typically require the user 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 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 Figure 8 forth row first column 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 Figure 9). The repairing result can be significantly improved. As our method is inherently robust, the provided support needs not 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 can not handle images with perspective deformation. In our method, the deformation can be estimated by solving (19). The estimated τ is then used to rectify the input region and we then solve (8) to repair the texture, typically with two iterations (first to estimate the error support, second to complete the image with a partially known error support). If the image has no deformation, we can simply skip the deformation estimation step. For a color image, three matrices associated with the three RGB channels are repaired independently. In this experiment, we set $\alpha = 0.85$, and all other parameters remain the same as previous simulations and experiments.

In Figure 8, we present comparisons with Shift-Map method [19], Patch Match (PM) ([18], [29]) and Structure Propagation (SP) [5] on four different images: a simulated non-uniform low-rank texture, a uniform building facade,

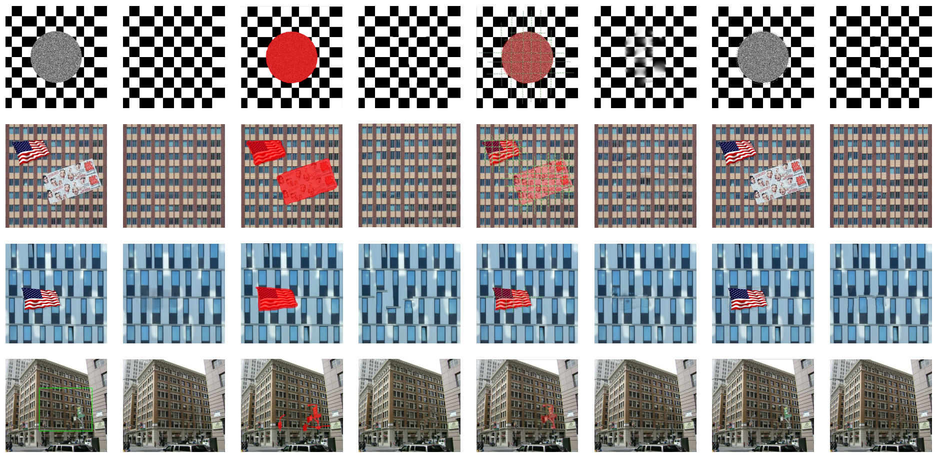


Fig. 8. Comparison results with Shift-Map [19], SP [5] and Photoshop [18]. Columns 1-2: input and result of our method; Columns 3-4: input and result of Shift-Map; Columns 5-6: input and result of SP; Columns 7-8: input and result of Photoshop. For rectified textures, our method can take the whole image as input, while in the deformed case, a green window indicates our input.

a somewhat less uniform building facade and a real building facade image with perspective deformation, which correspond to five rows in Figure. 8, respectively. As we see, these three method fail to recover precisely the consistent structural pattern of the original image.

C. More Examples of Image Repairing. To show how our method can handle realistic image repairing tasks, Figure 9 shows 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. As we see in these results, our algorithm could recover very well the low-rank and sparse structures in regular or near regular textures in natural images. However, as mentioned earlier in Section 2, our algorithm is not intended to work for random or near random textures that may violate the low-rank and sparse assumption, such as the last example shown in Figure9. It may "over-smooth" such textures hence the results might not be on par with local patch-based methods that are more suitable for such textures. Finally, notice that the recovered images of our method are not as sharp as the original input image because they are essentially estimated, not synthesized with original pixels. Nevertheless, it should not be difficult to improve the quality of our results by incorporating with local patch-based synthesis methods. We leave this for future study.

Acknowledgement. This work was partially supported by the funding of ONR N00014-09-1-0230, NSF CCF 09-64215, NSF IIS 11-16012, and DARPA KECoM 10036- 100471.

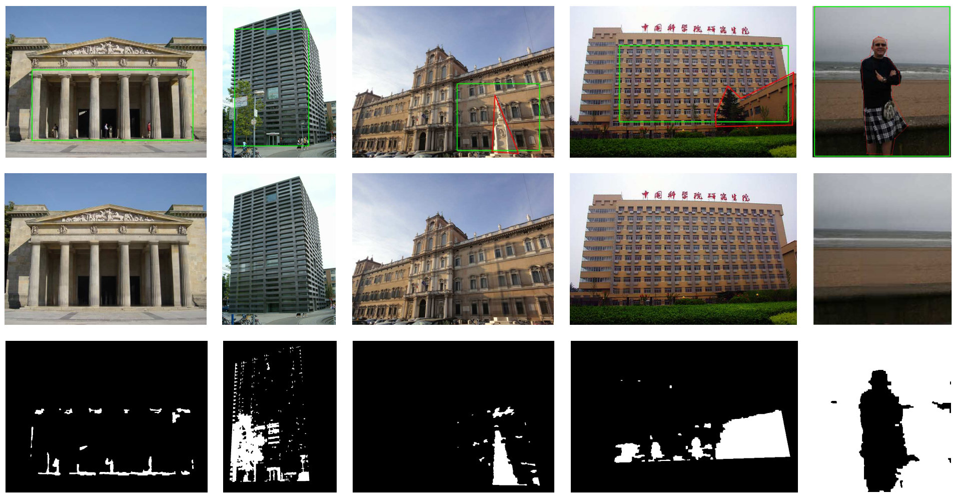


Fig. 9. Realistic repairing results. First row: input images where the green windows denote the input windows to our system, and red contours denote the initial error supports to our system. Second row: repairing images by our method. Third row: estimated final error supports by our method.

References

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

12. Mairal, J., Elad, M., Sapiro, G.: Sparse representation for color image restoration. the IEEE Trans. on Image Processing (TIP) **17** (2007) 53–69
13. Fadili, M.J., l. Starck, J., Murtagh, F.: Inpainting and zooming using sparse representations. The Computer Journal (2009) 6479
14. Elad, M., l. Starck, J., Querre, P., Donoho, D.: Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). The Computer Journal (2005) 340–358
15. Nigeau, A., Bertalmio, M., Caselles, V., Sapiro, G.: A comprehensive framework for image inpainting. the IEEE Trans. on Image Processing (TIP) **19** (2010) 2634 – 2645
16. Kwatra, V., Schodl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. ACM Transactions on Graphics (SIGGRAPH) **22** (2003) 277–286
17. Hays, J., Efros, A.A.: Scene completion using millions of photographs. ACM Transactions on Graphics (SIGGRAPH) **26** (2007)
18. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (SIGGRAPH) **28** (2009)
19. Pritch, Y., Kav-Venaki, E., Peleg, S.: Shift-map image editing. Proceedings of the Internetional Conference on Computer Vision (ICCV) (2009)
20. 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 (2010)
21. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational Mathematics **9** (2009) 717–772
22. Candès, E., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of the ACM **58** (2011)
23. Zhuang LS, Gao HY, L.Z.M.Y.Z.X.Y.N.: Non-negative low rank and sparse graph for semi-supervised learning. (2012)
24. Lin, Z., Liu, R., Su, Z.: Linearized alternating direction method with adaptive penalty for low-rank representation. Advances in Neural Information Processing Systems (NIPS) (2011)
25. Liu, J., Musialski, P., Wonka, P., Ye, J.: Tensor completion for estimating missing values in visual data. (2009) 2114–2121
26. Zhou, Z., Wagner, A., Mobahi, H., Wright, J., Ma, Y.: Face recognition with contiguous occlusion using markov random fields. in Proceedings of IEEE International Conference on Computer Vision (ICCV) (2009)
27. Zhang, Z., Liang, X., Ma, Y.: Unwrapping low-rank textures on generalized cylindrical surfaces. International Conference on Computer Vision (ICCV) (2011)
28. Zhang, Z., Ganesh, A., Liang, X., Ma, Y.: TILT: Transform-invariant low-rank textures. to appear in International Journal of Computer Vision (IJCV) (2011)
29. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized PatchMatch correspondence algorithm. European Conference on Computer Vision (ECCV) (2010)