TILT: Transform Invariant Low-Rank Textures

Zhengdong Zhang · Arvind Ganesh · Xiao Liang · Yi Ma

Received: 9 December 2010 / Accepted: 2 January 2012 / Published online: 12 January 2012
© Springer Science+Business Media, LLC 2012

Abstract In this paper, we propose a new tool to efficiently extract a class of “low-rank textures” in a 3D scene from user-specified windows in 2D images despite significant corruptions and warping. The low-rank textures capture geometrically meaningful structures in an image, which encompass conventional local features such as edges and corners as well as many kinds of regular, symmetric patterns ubiquitous in urban environments and man-made objects. Our approach to finding these low-rank textures leverages the recent breakthroughs in convex optimization that enable robust recovery of a high-dimensional low-rank matrix despite gross sparse errors. In the case of planar regions with significant affine or projective deformation, our method can accurately recover both the intrinsic low-rank texture and the unknown transformation, and hence both the geometry and appearance of the associated planar region in 3D. Extensive experimental results demonstrate that this new technique works effectively for many regular and near-regular patterns or objects that are approximately low-rank, such as symmetrical patterns, building facades, printed text, and human faces.

Keywords Transform invariant · Low-rank texture · Sparse errors · Robust PCA · Rank minimization · Image rectification · Shape from texture · Symmetry

1 Introduction

One of the fundamental problems in computer vision is to identify certain feature points or salient regions in images. These points and regions are the basic building blocks for almost all high-level vision applications such as image matching, 3D reconstruction, object recognition, and scene understanding. Through the years, a large number of methods have been proposed in literature for extracting various types of feature points or salient regions. The detected points or regions typically represent parts of the image that have distinctive geometric or statistical properties such as Canny edges (Canny 1986), Harris corners (Harris and Stephens 1988), and textons (Leung and Malik 2001).

One of the important applications of detecting feature points or regions in images is to establish point-wise correspondences or measure similarity between different images of the same object. This problem is especially challenging if the images are taken from different viewpoints under different lighting conditions. Thus, it is desirable that the detected points/regions are somewhat stable or invariant under transformations incurred by changes in viewpoint or illumination. In the past two decades, numerous “invariant” features and descriptors have been proposed, studied, compared, and combined in the literature (see Mikolajczyk...
and Schmid 2005; Winder and Brown 2007 and references therein). Some of the earliest work in this genre were based on using a Markov model to study dependences between various wavelet subbands for rotation invariant textures (Cohen et al. 1991; Chen and Kundu 1994; Wu and Wei 1996; Do and Vetterli 2002). There has also been a lot of study in using different kinds of basis functions, such as Gabor wavelets, to filter the image and compute rotation invariant features from the filtered image (see Haley and Manjunath 1999; Greenspan et al. 1994; Madiraju and Liu 1994 and references therein).

A widely used invariant feature descriptor is the scale invariant feature transform (SIFT) (Lowe 2004), which to a large extent is invariant to changes in rotation and scale (i.e., similarity transformations). Nevertheless, if the images are shot from very different viewpoints, SIFT is not very successful in establishing reliable correspondences. This problem has been partially addressed by its affine-invariant version (Mikolajczyk and Schmid 2004; Morel and Yu 2009). However, even these extensions of SIFT are limited in practice since although the deformation of a small distant patch can be well-approximated by an affine transformation, projective transformations are necessary to describe the deformation of a large region viewed through a perspective camera. There has been relatively limited work on texture representation that is invariant to projective transformations (Chang et al. 1987; Kondepudy and Healey 1994). To the best of our knowledge, from a practical standpoint, there are no feature descriptors that are truly invariant (or even approximately so) under projective transformations or homographies. In addition, these methods normally do not deal with other concurrent nuisance factors such as partial occlusions, corruptions, and illumination changes that could severely undermine local feature extraction and matching in real images.

Despite tremendous effort in the past few decades to search for better and richer classes of invariant features in images, there seems to be a fundamental dilemma that none of the existing methods have been able to resolve: On the one hand, if we consider the typical classes of transformations incurred on the image domain by changing camera viewpoint and on the image intensity by changing contrast or illumination, then in a strict mathematical sense, invariants of the 2D image are extremely sparse and scarce—essentially only the topology of the extrema of the image function remains invariant, known as attributed Reeb tree (ART) (Sundaramoorthi et al. 2009). The numerous “invariant” image features proposed in the computer vision literature, including the ones mentioned above, are at best approximately invariant, and often only to a limited extent. On the other hand, a typical 3D scene is rich in regular structures that are full of invariants (with respect to 3D Euclidean transformations or other well-behaving deformation groups). For instance, in an urban environment, the scene is typically filled with man-made objects that have parallel edges, right-angled corners, regular shapes, symmetric structures, and repeated patterns. These geometric structures are rich in properties that are invariant under all types of subgroups of the 3D Euclidean group. As a result, their 2D (affine or perspective) images encode very rich and precise information about the 3D geometry and structure of the objects in the scene (Ma et al. 2004; Kosecka and Zhang 2005; Schindler et al. 2008).

In this paper, we propose a technique that aims to resolve the above dilemma about image invariants. We contend that instead of trying to seek local invariant features of the image that are either scarce or imprecise, we should aim to directly extract certain invariant structures in 3D through their 2D images by undoing the (affine or projective) domain transformations.

That is, we cast our quest for “transform-invariance” directly as an inverse problem of recovering certain invariant 3D structures from their (projected and deformed) 2D images. In this paper, we will not only make it precise what rich class of structures that we can recover but also introduce some new powerful computational tools that allow us to solve the associated inverse problems efficiently.

Many methods have been developed in the past to detect and extract all types of regular, symmetric patterns from images despite affine or projective transformations (see Park et al. 2008 for a recent evaluation). In particular, this problem has been studied extensively for the purpose of rectifying building facades in urban scenes (Liu and Collins 2001; Ma et al. 2004; Liu et al. 2010; Park et al. 2010). As symmetry is not a property that depends on a small neighborhood of a pixel, it can only be detected from a relatively large region of the image. However, almost all existing methods for detecting symmetric regions and patterns start by extracting and putting together local features such as corners and edges (Yang et al. 2005) or more advanced local features such as SIFT points (Schindler et al. 2008). As local feature detection and edge extraction themselves are sensitive to local image variations such as noise, occlusion, and illumination change, such symmetry detection methods inherently lack robustness and stability. In addition, as we will see in this paper, many regular structures and symmetric patterns do not even have distinctive local features. Thus, we need a more general and holistic way of detecting and extracting regular structures in images despite significant distortion and corruption.

Our goal in this paper is to extract invariant (geometric and textural) information from regions in a 2D image that correspond to a very rich class of regular or near-regular patterns on a planar surface in 3D, whose appearance can be modeled (approximately) as a “low-rank” matrix (see Fig. 1.
Fig. 1 (Color online) Low-rank Textures Automatically Rectified by Our Method. From left to right: a butterfly; a face; a tablet of Chinese characters; and the Leaning Tower of Pisa. Top: red windows denote the original input, green windows denote the deformed texture returned by our method; Bottom: textures in the green window rectified for display. We notice that the rank of the image matrix, denoted by $r$, is much lower for the rectified textures.

Fig. 2 Representative Examples of Low-rank Textures and Our Results. From left to right: an edge; a corner; a symmetric pattern, and a license plate. Top: deformed textures (high-rank as matrices); Bottom: the recovered low-rank representations.

for some examples). In some sense, many conventional features mentioned above such as edges, corners, symmetric patterns can all be considered as special instances of such low-rank textures (see Fig. 2). Clearly, when an image of such a texture undergoes some domain transformation (say affine or projective), the transformed texture is no longer low-rank, when viewed as a matrix. Nevertheless, by utilizing advanced convex optimization tools from matrix rank minimization, we will show how we can simultaneously recover such a low-rank texture from its deformed image and the associated deformation.

The novelty of our approach is that it directly uses all raw pixels of the image region of interest and there is no need for pre-extraction of any intermediate low-level, local features such as corners, edges, SIFT, Gabor, and DoG features. It is applicable to any image region with sufficiently low rank when considered as a matrix, regardless of the size of its spatial support. We are able to rectify not only small local patches around an edge or a corner but also larger globally symmetric regions such as an entire facade of a building. Thus, the method is truly holistic in nature. Furthermore, the proposed formulation and solution are inherently robust to gross errors caused by corruption, occlusion, or cluttered...
background as long as they affect only a relatively small fraction of the image pixels. We believe that this is a powerful new tool that allows people to accurately extract rich geometric and textural information about a 3D region from its 2D images, that are truly invariant to image domain transformations. We have also made a MATLAB implementation of our algorithm publicly available at the following webpage: http://perception.csl.uiuc.edu/matrix-rank/tilt.html.

**Organization of This Paper** The remainder of this paper is organized as follows: Sect. 2 gives a rigorous definition of “low-rank textures” as well as formulates the mathematical problem associated with extracting such textures. Section 3 gives an efficient and effective algorithm for solving the problem. We provide extensive experimental results to verify the efficacy of the proposed algorithm as well as the problem. We provide extensive experimental results to verify the efficacy of the proposed algorithm as well as the usefulness of the extracted low-rank textures in Sect. 4. In Sect. 5, we discuss some potential extensions and variations to the basic formulation.

### 2 Transform Invariant Low-Rank Textures

#### 2.1 Definition of Low-Rank Textures

In this paper, we consider a 2D texture as a function $I^0(x, y)$, defined on $\mathbb{R}^2$. We say that $I^0$ is a low-rank texture if the family of one-dimensional functions $\{I^0(x, y_0) | y_0 \in \mathbb{R}\}$ span a finite low-dimensional linear subspace, i.e.,

$$r \doteq \dim \left( \text{span} \{I^0(x, y_0) | y_0 \in \mathbb{R} \} \right) \leq k \quad (1)$$

for some small positive integer $k$. If $r$ is finite, then we refer to $I^0$ as a rank-$r$ texture. It is easy to see that a rank-1 function $I^0(x, y)$ must be of the form $g(x) \cdot h(y)$ for some functions $g(\cdot)$ and $h(\cdot)$; and in general, a rank-$r$ function $I^0(x, y)$ can be explicitly factorized as the combination of $r$ rank-1 functions:

$$I^0(x, y) \doteq \sum_{i=1}^{r} g_i(x) \cdot h_i(y). \quad (2)$$

Figure 2 shows some ideal low-rank textures: a vertical or horizontal edge (or slope) can be considered a rank-1 texture, and a corner can be considered a rank-2 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 typically lead to low-rank textures. Thus, the notion of low-rank texture encompasses a much broader range of “features” or regions than just corners and edges. However, we would like to point out that low-rank is not a necessary attribute of all symmetric textures or shapes occurring in natural images. In this work, we only consider those symmetrical textures that give rise to low-rank matrices.

Given a low-rank texture, obviously its rank is invariant under any scaling of the function, as well as scaling or translation in the $x$ and $y$ coordinates. That is, if $I(x, y) \doteq cI^0(ax + t_1, by + t_2)$

for some constants $a, b, c \in \mathbb{R}^+, t_1, t_2 \in \mathbb{R}$, then $I(x, y)$ and $I^0(x, y)$ have the same rank according to our definition in (1). For most practical purposes, it suffices to recover any scaled or translated version of the low-rank texture $I^0(x, y)$, as the remaining ambiguity left in the scaling can often be easily resolved in practice by imposing additional constraints on the texture (see Sect. 3.2). Hence, in this paper, unless otherwise stated, we view two low-rank textures equivalent if they are scaled and translated versions of each other:

$$I^0(x, y) \sim cI^0(ax + t_1, by + t_2),$$

for all $a, b, c \in \mathbb{R}^+, t_1, t_2 \in \mathbb{R}$. In homogeneous representation, this equivalence group of transformations consists of all elements of the form:

$$G \doteq \left\{ \begin{bmatrix} a & 0 & t_1 \\ 0 & b & t_2 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \right| a, b \in \mathbb{R}^+, t_1, t_2 \in \mathbb{R} \right\}. \quad (3)$$

In practice, images of 2D textures are not continuous functions on $\mathbb{R}^2$. Typically, we only have its values sampled on a finite discrete grid in $\mathbb{Z}^2$, of size $m \times n$ say. In this case, the 2D texture $I^0(x, y)$ is represented by an $m \times n$ matrix of real entries. 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 functions $g_i(\cdot)$ or $h_i(\cdot)$ defined in (2), the resulting matrix has the same rank as the continuous function. Thus, the 2D texture $I^0(x, y)$ when discretized as a matrix, also denoted by $I^0$ for convenience, has very low rank relative to its dimensions.

**Remark 1** (Low-rank Textures vs. Random Textures) Conventionally, the word “texture” is used to describe image regions that exhibit certain spatially stationary stochastic properties (for modeling things like grass, sand, and fabrics). Such textures can be considered as random samples from a

1The scale of the window needs to be large enough to meet this assumption.

2In other words, the resolution of the image cannot be too low.
stationary stochastic process (Levina and Bickel 2006) and the images generally have full rank when viewed as matrix. The low-rank “textures” defined here are complementary to such full-rank random textures. Here, low-rank textures correspond to regions in an image that have deterministic regular low-dimensional structures. Despite its obvious importance, there has been a lack of effective tools for analyzing this class of textures.

2.2 Deformed and Corrupted Low-Rank Textures

In most real images, we almost never see a perfectly low-rank texture, largely due to two factors:

1. The change in viewpoint induces a transformation on the domain of the texture function
2. The sampled values of the texture function are subject to many types of corruption such as quantization, noise, occlusions, etc.

In order to correctly extract the intrinsic low-rank textures from such deformed and corrupted image measurements, we must first carefully model these factors and then seek ways to eliminate them.

Deformed Low-Rank Textures Although many planar surfaces or structures in 3D exhibit low-rank textures, their images do not necessarily have low rank. Suppose that a low-rank texture \( I^0(x, y) \) lies on a planar surface in the scene. The image \( I(x, y) \) that we observe from a certain viewpoint is a transformed version of the original low-rank texture function \( I^0(x, y) \):\(^3\)

\[
I(x, y) = I^0 \circ \tau^{-1}(x, y) = I^0(\tau^{-1}(x, y)),
\]

where \( \tau : \mathbb{R}^2 \to \mathbb{R}^2 \) belongs to a certain Lie group \( \mathbb{G} \). In this paper, we assume that \( \mathbb{G} \) is either the 2D affine group \( \text{Aff}(2) \), or the homography group \( GL(3) \) acting linearly on the image domain.\(^4\) In general, the transformed texture \( I(x, y) \) no longer has low rank when viewed as a matrix. For instance, an ideal horizontal edge has rank one, but when rotated by 45°, it becomes a full-rank diagonal edge (see Fig. 2(a)).

Corrupted Low-Rank Textures In addition to domain transformations, the observed image of the texture might be corrupted by noise and occlusions or contain some pixels from the surrounding background. We can model such deviations by an error matrix \( E \) as follows:

\[
I = I^0 + E.
\]

As a result, the image \( I \) might no longer be a low-rank texture. In this paper, we assume that only a small fraction of the image pixels are corrupted by large errors, and hence, \( E \) is a sparse matrix.

Our goal in this paper is to recover the exact low-rank texture \( I^0 \) from an image that contains a deformed and corrupted version of it. More precisely, we aim to solve the following problem:

**Problem 1 (Recovery of Low-rank Texture)** Given a deformed and corrupted image of a low-rank texture: \( I = (I^0 + E) \circ \tau^{-1} \), recover the low-rank texture \( I^0 \) and the domain transformation \( \tau \in \mathbb{G} \).

The above formulation naturally leads to the following optimization problem:

\[
\min_{I^0, E, \tau} \text{rank}(I^0) + \gamma \|E\|_0 \quad \text{s.t.} \quad I \circ \tau = I^0 + E, \quad (4)
\]

where \( \|E\|_0 \) denotes the number of non-zero entries in \( E \). That is, we aim to find the texture \( I^0 \) of the lowest possible rank and the error \( E \) with the fewest possible nonzero entries that agrees with the observation \( I \) up to a domain transformation \( \tau \). Here, \( \gamma > 0 \) is a weighting parameter that trades off the rank of the texture versus the sparsity of the error. For convenience, we refer to the solution \( I^0 \) found to this problem as a Transform Invariant Low-rank Texture (TILT).\(^5\)

**Remark 2 (TILT vs. Affine-Invariant Features)** TILT is fundamentally different from the affine-invariant features or regions proposed in the literature (Mikolajczyk and Schmid 2004; Morel and Yu 2009). Essentially, those features are extensions to SIFT features in the sense that their locations are detected in a manner very similar to SIFT. The difference is that around each feature, an optimal affine transform is found that in some way “normalizes” the local statistics, say by maximizing the isotropy of the brightness pattern (Garding and Lindeberg 1996). On the other hand, TILT finds the best local deformation by minimizing the rank of the brightness pattern in a robust way. It works the same way for any image region of any size and for both affine and projective transforms (or even more general transformation groups

\(^3\)It helps to model any low-rank texture as a function defined on a continuous domain \( \mathbb{R}^2 \) since we can talk about domain transformation freely. Any image or matrix representation of the texture is only a discrete sampling of this function. This allows us to generate transformed images of a low-rank texture by interpolating between values of adjacent pixels.

\(^4\)Nevertheless, in principle, our method works for more general classes of domain deformations or camera projection models as long as they can be modeled well by a finite-dimensional parametric group. See Zhang et al. (2011a, 2011b) for examples.

\(^5\)By a slight abuse of terminology, we also refer to the procedure of solving the optimization problem as TILT.
that have a smooth parameterization (Zhang et al. 2011a, 2011b)). More importantly, as we will see in Sect. 4, our method is able to rectify all kinds of regions that are approximately low-rank (e.g. human faces, printed text) and the results match very well with human perception. Unlike SIFT features whose locations are difficult to predict or interpret by human vision, the low-rank textures computed by TILT match well with human visual perception.

**Remark 3 (TILT vs. RASL)** We note that the optimization problem (4) is very similar to the robust image alignment problem studied in Peng et al. (2010), known as RASL. This is because both RASL and TILT use the same mathematical framework (sparse and low-rank matrix decomposition with domain transformation) in their problem formulation. Although the formulation is similar, there are some important conceptual differences between the two problems. For instance, RASL treats each image as a vector and does not make use of any spatial structure within each image, whereas in this paper, TILT uses matrix rank and sparsity to study spatial structures within a 2D image. Thus, RASL and TILT are highly complementary to each other in that they try to capture temporal and spatial linear correlation in images, respectively. From an algorithmic point of view, TILT is simpler than RASL since it deals with only one image and one domain transformation whereas RASL deals with multiple images and multiple transformations, one for each image. We will propose many extensions to TILT to handle a wider range of textures and symmetries, most of which are not applicable to the image alignment problem that RASL strives to solve. Although beyond the scope of this paper, it remains to be seen in the future if TILT and RASL could be combined together to develop a richer class of tools for extracting more information from images.

**Remark 4 (TILT vs. Transformed PCA)** One might argue that the low-rank objective can be directly enforced, as in Transformed Component Analysis (TCA) proposed by Frey and Jojic (1999), which uses an EM algorithm to compute principal components, subject to domain transformations drawn from a known group. The TCA deals with Gaussian noise and essentially minimizes the Euclidean norm of the error term $E$. It is only natural to ask if such a “transformed principal component analysis” approach could apply to our image rectification problem here. Suppose that we ignore gross corruption or occlusion for the time being. We could attempt to recover a rank-$r$ texture by solving the following optimization problem:

$$\min_{I^0, \tau} \|I \circ \tau - I^0\|_F^2 \quad \text{s.t.} \quad \text{rank}(I^0) \leq r, \quad (5)$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm. It is possible to solve (5) by minimizing against the low-rank component $I^0$ and the deformation $\tau$ iteratively. That is, with $\hat{\tau}$ fixed, we estimate the rank-$r$ component $\hat{I}^0$ via PCA, and with $\hat{I}^0$ fixed, we compute the deformation $\hat{\tau}$ in a greedy fashion to minimize the least-squares objective.\(^6\)

Figure 3 shows some representative results of using such a “Transformed PCA” approach. However, even for simple patterns like the checker-board, it works only with a correct initial guess of the rank $r = 2$ beforehand. If we assume a wrong rank, say $r = 1$ or 3, solving (5) would not converge to a correct solution, even with a small initial deformation. For complex textures like a building facade shown in Fig. 3, whose rank is impossible to guess in advance, it is necessary to try all possibilities. Moreover, (5) can only handle small Gaussian noise. For images taken in real world, partial occlusion and other types of corruption are often present. The naive transformed PCA does not work robustly against such forms of corruption. As we will see in the rest of this paper, the TILT algorithm that we propose next can automatically

---

\(^6\)In fact, this simple iteration closely emulates the expectation-maximization (EM) procedure for solving the TCA problem proposed by Frey and Jojic (1999).
find the minimal matrix rank in an efficient manner and handle very large deformations and non-Gaussian errors of large magnitude.

3 Solution by Iterative Convex Optimization

As proposed in Peng et al. (2010), although the rank function and the $\ell^0$-norm in the original problem (4) 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 (Candès et al. 2011; Chandrasekaran et al. 2011): the matrix nuclear norm $\|\cdot\|_*$ for rank($I^0$) and the $\ell^1$-norm $\|\cdot\|_1$ for $\|\cdot\|_0$, respectively. Thus, we end up with the following optimization problem:

$$\min_{I^0, E, \tau} \|I^0\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad I \circ \tau = I^0 + E. \tag{6}$$

We note that although the objective function in the above problem is convex, the constraint $I \circ \tau = I^0 + E$ is nonlinear in $\tau \in \mathbb{G}$, and hence the problem is not convex. A common technique to overcome this difficulty is to linearize the constraint (Baker and Matthews 2004; Peng et al. 2010) around the current estimate and iterate. Thus, the constraint for the linearized version of our problem becomes

$$I \circ \tau + \nabla I \Delta \tau = I^0 + E, \tag{7}$$

where $\nabla I$ is the Jacobian (derivatives of the image with respect to the transformation parameters). The optimization problem in (6) reduces to

$$\min_{I^0, E, \Delta \tau} \|I^0\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad I \circ \tau + \nabla I \Delta \tau = I^0 + E. \tag{8}$$

The linearized problem above is a convex program and is amenable to efficient solution. Since the linearization is only a local approximation to the original nonlinear problem, we solve it iteratively in order to converge to a (local) minimum of the original non-convex problem (6). The algorithm has been summarized as Algorithm 1.

The iterative linearization scheme outlined above is a common technique in optimization to solve nonlinear problems. It can be shown that this kind of iterative linearization converges quadratically to a local minimum of the original non-linear problem. A complete proof is out of the scope of this paper. We refer the interested reader to Peng et al. (2011), Cromme (1978), Jittorntrum and Osborne (1980) and the references therein.

3.1 Fast Algorithm Based on Augmented Lagrange Multiplier Methods

The most computationally expensive part of Algorithm 1 is solving the convex program in the inner loop (Step 2). This can be cast as a semidefinite program and 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. Fortunately, there has been a recent flurry of work in developing fast, scalable algorithms for nuclear norm minimization (Cai et al. 2010; Toh and Yun 2010; Ganesh et al. 2009; Lin et al. 2009). To solve the linearized problem in (8), we use the Augmented Lagrange Multiplier (ALM) method (Bertsekas 2004; Lin et al. 2009). For the sake of completeness, in this section we explain how the ALM method can be adapted to solve our problem, and also comment on some implementation details for improving stability and range of convergence.

3.1.1 General Formulation of ALM

We first review the ALM algorithm in a more general setting, rather than for our specific problem. This will be useful later when we deal with different variations of the TILT algorithm that can all be solved under the same algorithmic framework described here.
Let us consider convex optimization problems of the form:

$$\min_{X} f(X) \text{ s.t. } A(X) = b,$$

where $f$ is a continuous, convex function, $A$ is a linear function, and $b$ is a vector of appropriate dimension. The basic idea of Lagrangian methods is to convert the above constrained optimization problem into an unconstrained problem that has the same optimal solution.

For the above problem (9), we define the augmented Lagrangian function as follows:

$$L_{\mu}(X, Y) = f(X) + \langle Y, b - A(X) \rangle + \frac{\mu}{2} \|b - A(X)\|_2^2,$$  

where $Y$ is a Lagrange multiplier vector of appropriate dimension, $\| \cdot \|_2$ denotes the Euclidean norm, and $\mu > 0$ denotes the penalty imposed upon infeasible points. The following result from Bertsekas (2004) establishes an important relation between the original problem (9) and its augmented Lagrangian function (10).

**Theorem 1 (Optimality of ALM)** Suppose that $\hat{X}$ is the optimal solution to (9). Then, for appropriate choice of $Y$ and sufficiently large $\mu$, we have

$$\hat{X} = \arg\min_{X} L_{\mu}(X, Y).$$

Thus, we could solve an unconstrained convex minimization problem in order to obtain the solution to the original constrained convex program (9). This result, while of theoretical importance, is not directly useful in practice since the choice of $Y$ and $\mu$ is not known a priori.

ALM methods are a class of algorithms that simultaneously minimize the augmented Lagrangian function and compute an appropriate Lagrange multiplier. The basic ALM iteration proposed in Bertsekas (2004) is given by

$$X_{k+1} = \arg\min_{X} L_{\mu_k}(X, Y_k),$$

$$Y_{k+1} = Y_k + \mu_k(b - A(X_k)),$$

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

where $(\mu_k)$ is a monotonically increasing positive sequence $(\rho > 1)$. Thus, we have reduced the original optimization problem (9) to a sequence of unconstrained convex programs.

The above iteration is computationally useful only if $L_{\mu}(X, Y)$ is easy to minimize with respect to $X$. For the problems encountered in this paper, this turns out to be the case indeed. This can be attributed to the following key property of the matrix nuclear norm and 1-norm:

$$\mathcal{S}_\mu(Y_1 + Y_2) = \arg\min_{X} \mu \|X\|_1 - \langle X, Y_1 \rangle + \frac{1}{2} \|X - Y_2\|_F^2,$$

$$US_{\mu}[\Sigma]V^T = \arg\min_{X} \mu \|X\|_1 - \langle X, W_1 \rangle + \frac{1}{2} \|X - W_2\|_F^2,$$

where $US_{\mu}[\Sigma]V^T$ is the Singular Value Decomposition (SVD) of $(W_1 + W_2)$, and $\mu$ is any non-negative real constant. Here, $\mathcal{S}[]$ represents the soft-thresholding or shrinkage operator which is defined on scalars as follows:

$$S_\mu(x) = \text{sign}(x) \cdot (\|x\| - \mu),$$

where $\mu \geq 0$. The shrinkage operator is extended to vectors and matrices by applying it elementwise. We now discuss how this iterative scheme can be applied to our linearized convex program (8).

### 3.1.2 Solving TILT by Alternating Direction Method

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

$$L_{\mu}(I^0, E, \Delta \tau, Y) = f(I^0, E) + \langle Y, R(I^0, E, \Delta \tau) \rangle + \frac{\mu}{2} \|R(I^0, E, \Delta \tau)\|_2^2,$$

where $\mu > 0$, $Y$ is a Lagrange multiplier matrix, $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, and $f(I^0, E) = \|I^0\|_+ + \lambda \|E\|_1$.

$$R(I^0, E, \Delta \tau) = I \circ \tau + V I \Delta \tau - I^0 - E.$$

From the above discussion, the basic ALM iteration scheme for our problem is given by

$$(I_{k+1}^0, E_k, \Delta \tau_k) = \arg\min_{I^0, E, \Delta \tau} L_{\mu_k}(I^0, E, \Delta \tau, Y_{k-1}),$$

$$Y_k = Y_{k-1} + \mu_{k-1} R(I_{k+1}^0, E_k, \Delta \tau_k).$$

Throughout the rest of the paper, we will always assume that $\mu_k = \rho^k \mu_0$ for some $\mu_0 > 0$ and $\rho > 1$, unless otherwise specified.

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 $I^0, E$ and $\Delta \tau$ simultaneously. So, we adopt a common strategy to solve it **approximately** by adopting an alternating minimizing scheme i.e., minimizing with respect to $I^0, E$ and $\Delta \tau$ one at a time:

$$I_{k+1}^0 = \arg\min_{I^0} L_{\mu_k}(I^0, E_k, \Delta \tau_k, Y_k),$$

$$E_{k+1} = \arg\min_{E} L_{\mu_k}(I_{k+1}^0, E, \Delta \tau_k, Y_k),$$

$$\Delta \tau_{k+1} = \arg\min_{\Delta \tau} L_{\mu_k}(I_{k+1}^0, E_{k+1}, \Delta \tau, Y_k).$$
Due to the special structure of our problem, each of the above optimization problems has a simple closed-form solution, and hence, can be solved in a single step. More precisely, the solutions to (15) can be expressed explicitly using the shrinkage operator as follows:

\[ I_{k+1}^0 \leftarrow U_k S_{\mu_k}[\Sigma_k] V_k^T, \]
\[ E_{k+1} \leftarrow S_{\mu_k}[I \circ \tau + \nabla I \Delta \tau_k - I_{k+1}^0 + \mu_k^{-1} Y_k], \]
\[ \Delta \tau_{k+1} \leftarrow (\nabla I)^\dagger (-I \circ \tau + I_{k+1}^0 + E_{k+1} - \mu_k^{-1} Y_k), \]

where \( U_k \Sigma_k V_k^T \) is the SVD of \((I \circ \tau + \nabla I \Delta \tau_k - E_k + \mu_k^{-1} Y_k)\), and \((\nabla I)^\dagger\) denotes the Moore-Penrose pseudo-inverse of \(\nabla I\).

From experiments, we observe that the above algorithm is much faster than all other alternative convex optimization schemes (such as the interior point method, accelerated proximal gradient, etc.). Although the convergence of the ALM method (11) has been well established in the optimization literature, its convergence has been well-studied and established proximal gradient, etc.). Although the convergence of the ADM scheme (15) remains an open problem (see also He 2009). However, the scheme proposed and proved in Yuan and Tao (2011) is slightly different from the direct ADM scheme (15) and is much slower in practice. The convergence of the ADM scheme (15) remains an open problem although in practice it converges to the desired solution in most cases. Recently, a variant of the alternating direction method with Gaussian back substitution for more than two sets of separable variables has been proposed in He et al. (2011).

We summarize the ADM scheme for solving (8) as Algorithm 2. We choose the sequence \(\{\mu_k\}\) to satisfy \(\mu_{k+1} = \rho \mu_k\) for some \(\rho > 1\). We note that the operations in each step of the algorithm are very simple with the SVD computation being the most computationally expensive step.\(^{10}\)

---

\(^{10}\)Empirically, we notice that for larger window sizes (over 100 \times 100 pixels), it is much faster to run the partial SVD instead of the full SVD, if the rank of the texture is known to be very low.

---

### Algorithm 2 (Solving Inner Loop of TILT)

**INPUT:** The current (deformed and normalized) image \(I \circ \tau \in \mathbb{R}^{m \times n}\) and its Jacobian \(\nabla I\) against deformation \(\tau\), and \(\lambda > 0\).

**Initialization:** \(k = 0, Y_0 = 0, E_0 = 0, \Delta \tau_0 = 0, \mu_0 > 0, \rho > 1\);

**WHILE** not converged DO

\[ (U_k, \Sigma_k, V_k) = \text{svd}(I \circ \tau + \nabla I \Delta \tau_k - E_k + \mu_k^{-1} Y_k); \]
\[ I_{k+1}^0 = U_k S_{\mu_k}[\Sigma_k] V_k^T; \]
\[ E_{k+1} = S_{\mu_k}^{-1}[I \circ \tau + \nabla I \Delta \tau_k - I_{k+1}^0 + \mu_k^{-1} Y_k]; \]
\[ \Delta \tau_{k+1} = (\nabla I)^\dagger (-I \circ \tau + I_{k+1}^0 + E_{k+1} - \mu_k^{-1} Y_k); \]
\[ Y_{k+1} = Y_k + \mu_k (I \circ \tau + \nabla I \Delta \tau_{k+1} - I_{k+1}^0 - \Delta \tau_{k+1}); \]
\[ \mu_{k+1} = \rho \mu_k; \]

**END WHILE**

**OUTPUT:** solution \((I^0, E, \Delta \tau)\) to problem (8).

### 3.2 Implementation Details

In the previous section, we described how the linearized and convexified TILT problem (6) can be solved efficiently using the ALM algorithm. However, there are a few caveats in applying it to real images. In this section, we discuss some possible ways to deal with these issues and make the problem well-defined. We also discuss some specific implementation details that could potentially improve the range of convergence of our algorithm.

**Constraints on the Transformations** As discussed in Sect. 2, there are certain ambiguities in the definition of low-rank texture. The rank of a low-rank texture function is invariant with respect to scaling in the pixel values, scaling in each of the coordinate axes, and translation along any direction. Thus, in order for the problem to have a unique, well-defined optimal solution, we need to eliminate these ambiguities. In Step 1 of Algorithm 1, the intensity of the image is renormalized in each iteration in order to eliminate the ambiguity of scale in the pixel values. Otherwise, the algorithm may tend to converge to a “globally optimal” solution by zooming into a black pixel or dark region of the image.

To deal with the ambiguities in the domain transformation, we could add some additional constraints to the problem. Let \(\tau(\cdot)\) represent the transformation. Suppose that the support of the initial image window \(\Omega\) is a rectangle (call the edges \(e_1\) and \(e_2\)) with the length of the two edges being \(L(e_1) = a\) and \(L(e_2) = b\), so that the total area \(S(\Omega) = ab\).

For affine transformations, to eliminate the ambiguity in translation, we typically enforce that the center \(x_0\) of the initial rectangular region \(\Omega\) remain fixed before and after the transformation i.e., \(\tau(x_0) = x_0\). This imposes a set of linear constraints on \(\Delta \tau\) of the form:

\[ A_1 \Delta \tau = 0. \] (17)
To eliminate the ambiguities in scaling the coordinates, we enforce (only for affine transformations) that the area and the ratio of edge length remain constant before and after the transformation, i.e. \( S(\tau(\Omega)) = S(\Omega) \) and \( L(\tau(e_1))/L(\tau(e_2)) = L(e_1)/L(e_2) \). In general, these equalities impose additional non-linear constraints on the desired transformation \( \tau \) in problem (6). Just as we had dealt with the non-linearity in the constraint in (6), we can linearize these additional constraints with respect to the transformation parameters \( \tau \) and obtain another set of linear constraints on \( \Delta \tau \) denoted by:

\[
A_\tau \Delta \tau = 0. \tag{18}
\]

We have provided a more detailed description of the derivation of the above-mentioned linear constraints in the Appendix.

For projective transformations, we typically fix two points, the two diagonal corners of the initial rectangular window or of the parallelogram if initialized with the result of the affine TILT. Notice that a homography matrix has a total of eight degrees of freedom. If the low-rank texture is associated with certain symmetric pattern that has two sets of parallel lines, the \( x \) and \( y \)-axes of the rectified low-rank texture then correspond to the two vanishing points. The two vanishing points and the two fixed points together impose exactly eight constraints and uniquely determine the homography. Hence, with this parameterization, there is no ambiguity in the optimal solution.

Thus, to eliminate the scaling and translation ambiguities in the solution, we simply add a set of linear constraints to the optimization problem (8). The resulting convex program can be solved again using the ALM algorithm. This would involve making very small modifications to Algorithm 2 to incorporate the additional linear constraints.

**Multi-resolution Approach** While the above formulation works reasonably well in practice, the presence of arbitrarily shaped sharp features or contours on an otherwise smooth low-rank texture can cause the TILT algorithm to converge to a local minima that is not the desired solution. Hence, to cope with large deformations, we adopt a multi-resolution approach. This is a common technique in many computer vision algorithms wherein we construct a pyramid of images, starting from the input image, by subsequently blurring and downsampling it. The problem is then solved at the lowest resolution first. The solution thus obtained is used to initialize the algorithm at the adjacent level of higher resolution, and this procedure is repeated for all levels. In practice, the multi-resolution approach not only improves the range of transformations that our algorithm can handle, but it also improves the running time of the algorithm significantly. This is because, the convex programs can be solved much faster at the lower resolutions, and since the initialization at the higher resolution is better, the number of iterations to convergence is typically very small (less than 20).

An important consideration while incorporating the multi-resolution approach for the TILT algorithm is the fact that the convex relaxation discussed in Sect. 3 is tight only at higher dimensions (or when the matrix size is large). Although it is very difficult to analytically estimate the minimum optimal size of the image, in practice, we find that our method works well for windows of size larger than \( 20 \times 20 \) pixels. In our implementation, we use a Gaussian kernel to blur the image and consider up to two levels of downsampling, each by a factor of 2 with respect to its adjacent higher level of resolution. We also ensure that the size of the image at the lowest resolution is at least \( 20 \times 20 \) pixels. We have tested the speed of this scheme in MATLAB on a 3 GHz PC. Fixing the initial window to have size \( 50 \times 50 \), the time taken is less than 6 seconds, averaged over 100 trials.

**Branch-and-Bound Scheme** We can increase the range of deformation that our algorithm can handle significantly by employing a branch-and-bound scheme. For instance, in the affine case, we initialize Algorithm 1 with different deformations (e.g., a combination search for all 4 degrees of freedom for affine transformations with no translation). Any affine transformation can be parametrized by \( [Ab] \in \mathbb{R}^{2 \times 2} \times \mathbb{R}^{2} \). Since we fix the center of the window, we effectively set \( b = 0 \). The remaining 4 parameters of the transformation denote the scaling along the \( x \) and \( y \)-axes, rotation, and skew. As discussed in Sect. 2, the scaling along the canonical axes does not change the rank of the texture, and hence, we ignore the ambiguity in it. Thus, we are left with two parameters—skew and rotation—that need to be determined. In other words, we can parametrize the affine transformation matrix \( A \) as:

\[
A(\theta, t) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \times \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.
\]

We partition the parameter space (rotation and skew) into multiple regions and perform a greedy search on the regions one-by-one. We first run TILT for various initializations of the rotation angle. We choose the one that minimizes the

---

11 In fact, one can use the same set of constraints as the affine case. But from our experience, the algorithm is more stable with the initialization of two points. In addition, as we will explain, the parameterization is more geometrically meaningful.

12 In practice, we almost always initialize the projective case with the result from the affine case.

13 We only have to introduce an additional set of Lagrangian multipliers and then revise accordingly the update equation associated with \( \Delta t_k+1 \).

14 The convex relaxation has a failure probability associated with it which typically decays as \( O(n^{-\alpha}) \), for some \( \alpha > 0 \), assuming that the matrices involved have size \( n \times n \).
cost function, and use this as an initialization to search for the skew parameters along the $x$-direction first, and subsequently along the $y$-direction. The parameters that minimize the cost function is the output of the branch-and-bound scheme.

A natural concern about such a branch-and-bound scheme is its effect on speed. Within the multi-resolution scheme, we only perform branch-and-bound at the level of lowest resolution, find the best solution, and use it to initialize the higher-resolution levels. Since Algorithm 1 is extremely fast for small matrices at the lowest-resolution level, running multiple instances with different initializations does not significantly affect the overall speed. In a similar spirit, to find the optimal projective transform (homography), we always find the optimal affine transformation first and then use it to initialize the algorithm to find the homography.15 From our experience, we found that with this initialization, we normally did not have to use the branch-and-bound scheme for the projective transformation case.

4 Experimental Results

In this section, we present the results of the proposed TILT algorithm on various natural and artificial low-rank textures. We first present some results quantifying the performance range of our algorithm. We then present examples from many different categories of natural images where TILT can recover the inherent symmetrical texture in the images. Finally, we present some examples where TILT does not recover the low-rank texture and examine the reasons for such failures.

4.1 Range of Convergence of TILT

For most low-rank textures, the proposed Algorithm 1 has a fairly large range of convergence, without using any branch-and-bound. In this section, we give a careful characterization of the range of convergence (ROC) of the proposed algorithm on a standard checker-board pattern.

Affine Case We deform a checker-board like pattern by a wide range of affine transformations of the form: $y = Ax + b$, where $x, y \in \mathbb{R}^2$, and test if the algorithm converges back to the correct solution. We parameterize the affine matrix $A$ as

$$A(\theta, t) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \times \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.$$
We vary \((\theta, t)\) in the range \(\theta \in [0, \pi/6]\) (in radians) with step size \(\pi/60\), and \(t \in [0, 1]\) with step size 0.05. We carry out 10 independent trials in each region and compute the success rate. Figure 4(b) shows the rate of success for all regions. We observe that the algorithm always finds the correct solution for up to \(\theta = 20^\circ\) of rotation and skew (or warp) of up to \(t = 0.4\). It is clear that due to its plain texture within each square block and sharp edges, the checker-board like pattern is a challenging case for “global” convergence as at many angles, its image corresponds to a local minimum that has relatively low rank. In practice, we find that for most symmetric patterns in urban scenes (as shown in Fig. 8), our algorithm converges over a much larger range of deformations even without the use of the branch-and-bound scheme.

Projective Case For the case of projective transformation (or homography), even if we fix two points in the image, there are still four degrees of freedom. It is difficult to illustrate the range of convergence across all four parameters together. So, here, we test the range of convergence over some of the most representative projective transformations that we normally encounter in real-world images, namely those equivalent to a planar low-rank pattern rotating in front of a perspective camera.

In this experiment, we place a standard checker-board pattern in front of a standard perspective camera—the image plane is the \(xy\)-plane and the optical axis is the \(z\)-axis. We rotate the pattern along a line in the \(xy\)-plane passing through the origin. We parametrize the axis of rotation by the angle it makes with the \(x\)-axis. We find the limits of the TILT algorithm by gradually increasing the amount of rotation along each axis (from \(0^\circ\) to \(90^\circ\) at steps of \(5^\circ\)). We also change the rotation axis from the \(x\)-direction (\(0^\circ\)) to the \(y\)-direction (\(90^\circ\)). Figure 5 shows the range of convergence of TILT under this setting. The curves indicate the case where TILT fails for the first time, or equivalently, TILT succeeds for all cases below the curves.

The two curves in the plot compare two different experiment settings. The first case (green curve) is just the basic projective TILT without any special initialization nor any branch-and-bound, and the second case (red curve) is the projective TILT initialized with the results from the affine TILT. From these results, we may conclude:

– The basic projective TILT works extremely well for the slanted checker-board like pattern—it converges even when the pattern has been rotated by up to \(50^\circ\) in all directions.

– Initialization with the affine TILT normally boosts the range of convergence for the projective TILT. With the affine initialization, the algorithm converges until the image has been rotated by more than \(65^\circ\) or rotation. In some cases, the increase in the range of convergence is by as much as \(20^\circ\), as compared to the basic projective TILT.

There are many possible ways to further improve the range of convergence for the TILT algorithm. So far, we have always used an upright rectangular window as the initial window. As we will see with experiments in later sections, TILT

\(^{16}\)The setting is symmetric and the pattern is symmetric so we only have to verify the range of convergence for the first quadrant.
Fig. 6 Robustness Tests of TILT on various low-rank textures. The textures on the left are ordered in descending order of being robust to random corruptions: from left to right, from top to bottom. The plot on the right shows the success rate of TILT many different textures at each level of corruption.

Fig. 7 Robustness of TILT. Top row: random corruption added to 60% pixels; Middle row: scratches added on a symmetric pattern; Bottom row: containing cluttered background

could work much better if the initial window is chosen in a way that is more adaptive to the orientation of the texture as well as the scale of the texture.

4.2 Robustness of TILT

In this experiment, we test the robustness of TILT on some representative synthetic and realistic low-rank patterns, shown in Fig. 6 (left). We introduce a small deformation to each texture (say rotation by 10°) and examine if TILT converges to the correct solution under different levels of random corruption. We randomly select a fraction (from 0% to 100%) of the pixels and assign them a random integer value between 0 and 255, both included. We run the TILT algorithm on such corrupted images and examine how
many images are correctly rectified by TILT at each level of corruption. The results are shown in Fig. 6 (right).

We observe that even when 30% of the pixels, randomly chosen, in the images are corrupted, TILT succeeds for a large majority (75%) of the images. For textures in the first row of Fig. 6 (left), TILT succeeds even when up to 50% of the pixels are corrupted. A closer look at the experimental results shows that TILT has low error tolerance for textures that either have very low contrast, or are rather sparse themselves, or have relatively high rank with respect to the window size.

Figure 7 show some more examples for the robustness of the proposed algorithm to random corruption, occlusions, and cluttered background, respectively. For the first two images in Fig. 7, TILT succeeds even if the branch-and-bound scheme is not used.

The above experiments demonstrate the robustness of TILT to randomly located corruptions. However, in some cases, we may have some idea about the part of the images that are likely to be corrupted or occluded. For instance, if the initial window is too close to the image boundary, the algorithm may converge to a region outside of the image boundary. In such cases, we know the locations of pixels in the region that are missing. This information about the location of some corrupted or missing pixels can help us to modify the algorithm and further improve its robustness. We will discuss this case in more detail in Sect. 5 when we study possible extensions to TILT.

Fig. 8 Shape from (Low-rank) Textures. Top left: The input grid of 60 × 60 windows. Top right: Low-rank textures detected by the TILT algorithm with affine transformations and the recovered local affine geometry. Middle left: Use homography to get the projective transformations. Middle right: the resulting image with the marked regions augmented with virtual objects. Bottom row: representative low-rank textures recovered from the marked regions of the buildings.
4.3 Shape from Low-Rank Textures

Obviously, the rectified low-rank textures found by our algorithm can facilitate many vision tasks, including establishing correspondences among images, recognizing text and objects, or reconstructing the 3D structure of a scene, etc. Due to limited space, we only illustrate how our algorithm can help extract precise, rich geometric and structural information from an image of an urban scene, as shown in Fig. 8 (top). This complements many existing “Shape from X” methods in the computer vision literature.

The size of the image shown in Fig. 8 is 1024 × 685 pixels and we simply run the TILT algorithm (with affine transformations) on a grid of 60 × 60 windows. If the rank of the resulting texture drops significantly from that of the original window, we say that the algorithm has “detected” a region with some low-rank structure.17 In Fig. 8, we have shown the resulting deformed windows, together with the local orientation and surface normal recovered from the recovered affine transformation. We note that for windows located in the interior of the building facades, TILT correctly recovers the local geometry for almost all of them. Even for patches located at the edge of the facades, one of the sides of the rectified patches always aligns precisely with the building’s edge. For patches located on the curved facade on the right, TILT still manages to recover the correct dominant geometry even though the surface is not perfect planar.18

It is also possible to initialize the size of the windows at different sizes or scales. For larger regions, it is clear that affine transformations are not sufficient to describe the deformation caused by a perspective projection accurately. For instance, the entire facade of the middle building in Fig. 8 (middle row) obviously exhibits significant projective deformation. Nevertheless, if we initialize the projective TILT algorithm with the output from the affine TILT algorithm on a small patch on the facade, the algorithm can easily converge to the correct homography and recover the low-rank textures correctly, as shown in Fig. 8 (middle row).

With both the low-rank texture and their geometry correctly recovered, we can easily perform many interesting tasks such as editing parts of the images without violating the true 3D shape and the correct perspective of the scene. Figure 8 (middle row) shows some examples which suggest that our method can be very useful for many augmented reality applications.

17Here, the image rank is computed by considering only those singular values that are at least 1/30 times the largest singular value. We also throw away regions whose largest singular value is too small, which typically correspond to smooth regions like the sky.

18For a more precise treatment of low-rank textures on a curved surface, one may refer to the more recent work of Zhang et al. (2011a).

4.4 Rectifying Many Categories of Low-Rank Textures

In this section, we test the efficacy of the TILT algorithm on natural images belonging to various categories. Besides some examples where TILT 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 limitations of the TILT algorithm.

We have seen that for artificial examples, the proposed TILT algorithm has a decent range of convergence for both affine and projective deformations, and it is also very robust to sparse corruption of the image pixels. Here, we demonstrate that it works remarkably well for a very broad range of patterns, regular structures, natural objects and even printed text that have an approximate low-rank structure. Figure 9 shows many such examples, from which we see that even when the initialization (with a rectangular window) is quite rough, our algorithm can converge precisely to the underlying low-rank structure of the images, despite occlusions, noise in background, illumination changes, and significant domain deformation.

Issues with More Challenging Cases  Our algorithm is expected to work well only when the low-rank and sparse structure assumptions, explained in Sect. 2, hold true. 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 section and the next section, we will discuss some of the limitations of TILT, as well as potential extensions that make it work better in some of the more challenging cases. Figure 10 shows some examples on which TILT does not perform as well as it did in previous examples. These examples are arguably more challenging than those shown in Fig. 9:

- Figure 10(a) is an example of choosing a very large initial window. Ideally, the correct solution is supposed to converge to a region beyond the image boundary. It stops once it hits the boundary which is a only partially correct solution. In the next section, we will show how this problem can be addressed by combining the basic TILT algorithm with techniques from low-rank matrix completion.
- Figure 10(b) shows a case where the algorithm manages to converge to an approximate solution despite the fact that there is a lack of regularity or precise low-rank structure in the printed text. TILT managed to correct the perspective distortion partially in this case.
- Figure 10(c) shows a case where the algorithm manages to correct the overall pose of the object despite fact that
Fig. 9  (Color online) Representative Results of TILT. The objects can be categorized as follows. Top two rows: regular patterns and textures; Middle two rows: signs, characters, and printed text; Bottom two rows: bar code, objects with bilateral symmetry. In each case, the red window denotes the input and the green window denotes the final output. The image enclosed by the green window is rectified and displayed to emphasize the low-rank structure.

– Figure 10(g) shows a failed case, where the perspective deformation is very large in the chosen input window and the texture is complex (i.e., the rank is relatively high). Nevertheless, with slightly better initialization, we expect the TILT algorithm to converge to the cor-

\[\text{Say by aggregating TILT results from smaller affine patches or by using rough manual input.}\]
Fig. 10 Challenging Cases.
TILT converges to an approximately correct solution at best for these examples. **Top:** from left to right: boundary problem, not enough regular texture, non-planar objects. **Bottom:** from left to right: large perspective distortion; too much random texture in background; sparse (binary image) low-rank structure.

rect solution. For example, as shown in Fig. 11 (top), if we simply shorten the width of the initial window along the direction in which the image is majorly deformed, the algorithm manages to find the correct solution.

- Figure 10(h) shows another failed case, where the initial window contains too many pixels from the background, which has the appearance of a random texture with little structure. Here, the algorithm converges to an undesirable local minimum. Nevertheless, with a slightly different initial window that contains fewer background pixels, the algorithm converges to the correct solution (see Fig. 11 (bottom)).

- Figure 10(i) shows a case where the low-rank texture itself is close to a sparse binary image. The algorithm only manages to converge to a partially correct transformation—the recovered texture is approximately symmetric along the horizontal direction. In this case, in order to improve the results, we may have to tune the weighting factor between the low-rank and sparse components in the cost function in (8), or enforce the symmetry of the desired solution explicitly in the form of additional constraints.

Expected Failures It should come as no surprise that when the assumptions of TILT are violated, it no longer finds the low-rank structure and the transformation correctly. Figure 12 shows some examples of TILT where it failed.

- The first example (Fig. 12(a)) shows the limitations of the “low-rank” assumption on some man-made structures: Two incompatible dominant low-rank structures (the facade and the shadow) are overlapped, which result in an overall high-rank region. TILT actually aligns the window along the orientation of the shadow. In order to succeed for this case, a simple “low-rank” promoting objective, like the one used in TILT, is no longer sufficient.
The second example (Fig. 12(b)) shows another limitation of the low-rank assumption. If the chosen window contains two adjacent low-rank regions each of which is distorted differently, the combined region might no longer be low-rank when subject to one global affine or projective transformation. To deal with this issue, proper segmentation of the different low-rank regions is needed before TILT can work correctly on each of the low-rank regions, or TILT has to be extended to simultaneously handle multiple domain transformations.

Although TILT is designed to be robust to corruptions or occlusions, it is effective only when the amount of corruption is not very large. As shown in Fig. 12(c), if there is too much occlusion, TILT cannot be expected to succeed, even though human vision is still capable of perceiving the building structures behind the tree. It remains to be seen whether the robustness of TILT can be improved to handle such challenging cases.

As mentioned earlier in Sect. 2, TILT is not designed to work on random textures occurring in nature, such as the one shown in Fig. 12(d). Although there has been work in the literature showing that it is possible to infer approximate orientation of the flower bed based on the statistical properties of the random texture, TILT is certainly not designed to handle such cases—it is effective for regular symmetric textures, but not for random textures.

5 Potential Modifications and Extensions

The TILT algorithm proposed in this paper is still rather rudimentary. Nevertheless, due to its simplicity, it can be easily modified or extended to handle more complex scenarios in natural images. In this section, we demonstrate this with three possible extensions. We do not claim that we have given the best possible solution to each problem discussed here. Instead, the goal is merely to show how TILT can be modified using some basic ideas. In fact, we believe that each of the following problems deserves a much more thorough investigation so that more effective and efficient algorithms could be developed in the future.

5.1 Matrix Completion for Boundary Effects

We note that in Step 3 of Algorithm 1, we update the transformation parameters $\tau$, and recompute the transformed im-

Fig. 11 (Color online) Effect of Initialization. For the examples in Fig. 10(g) and (h) where TILT had failed earlier, the correct solution is found with a slightly different initialization, in both cases by reducing the horizontal width of the initial (red) window.

(a) high-rank structures   (b) two low-rank regions   (c) too much occlusion   (d) random textures

Fig. 12 Failure Cases. TILT fails to recover the geometry of these images since they deviate from the assumptions under which TILT is designed to work. From left to right: two incompatible dominant low-rank structures, overlapped or adjacent; too much occlusion; random textures.
age \( I \circ \tau \) in Step 1 of the subsequent iteration. While this is conceptually sound, it poses a serious problem in practice. This is because real images always have finite support or size. So, if the window containing the texture of interest is close to the image boundary, then the transformed image window \( I \circ \tau \) might not be well-defined at all pixels. The conventional methods to treat this problem is to either assume that the region outside the image has a constant pixel value of zero, or to interpolate them from the boundary pixels ensuring some degree of smoothness. The former approach is ill-suited to our problem since it may destroy the low-rank structure of the texture inside the image (hence TILT may fail to converge to the correct solution as shown in Fig. 10(a)), while the latter introduces more free parameters to the algorithm, namely the choice of the interpolation function.

This problem can actually be handled in a more principled manner. We treat the pixels that fall outside the image boundary as missing entries of the low-rank matrix to be recovered. This formulation is in a similar spirit as the low-rank matrix completion problem that has been extensively studied recently (Recht et al. 2010; Candès and Recht 2009; Candès and Tao 2010). Let \( \Omega \) represent the set of pixels that are located inside the image boundary after transformation. Then, we modify the constraint in the linearized problem (8) as follows:

\[
\pi_\Omega (I \circ \tau + \nabla I \Delta \tau) = \pi_\Omega (I^0 + E), \tag{19}
\]

where \( \pi_\Omega (\cdot) \) denotes the projection operator onto the linear subspace of matrices with support in \( \Omega \). Thus, we apply the constraint only on the set of pixels at which the transformed image \( I \circ \tau \) is well-defined. Since \( \pi_\Omega (\cdot) \) is a linear operator, the resulting optimization problem is still a convex program and can be solved by the ALM algorithm outlined in Sect. 3.1.\(^{20}\) Figure 13 shows two examples of how matrix completion could improve the performance of TILT when the chosen window is too close to the boundaries of the image and the desired solution is required to converge to a window outside of the original image.

### 5.2 Enforcing Reflective Symmetry

As pointed out earlier, a low-rank matrix is merely the consequence of many types of regularities and symmetries in the image. However, a low-rank texture need not necessarily be symmetric. Hence, if we intend to recover a symmetric texture, the axis of symmetry is not necessarily always at the center of the recovered low-rank region. So, in order to ensure that the recovered low-rank region has such types of symmetries, additional constraints need to be imposed on TILT.

Suppose that \( I^0 \in \mathbb{R}^{m \times n} \) represents the image of a texture with reflective symmetry. Without any loss of generality, we may assume that the axis of symmetry is horizontal. Then, the reflective symmetry of \( I^0 \) can be expressed mathemati-

---

\(^{20}\)One can even handle small noise in this case, as shown in the work of Yuan and Tao (2011).
cally as

\[ I^0(i, j) = I^0(m + 1 - i, j), \quad \forall(i, j) \in [m] \times [n], \]  

(20)

where \([k]\) denotes the set of integers from 1 through \(k\), for any positive integer \(k\). In general, for most types of symmetry present in an image \(I^0\), we can find an invertible linear mapping \(g : \mathbb{R}^2 \rightarrow \mathbb{R}^2\) such that \(g(I^0) = I^0\).\(^{21}\) Thus, we may add any desired symmetry as an additional set of constraints to the linearized convex program (8) in the TILT framework. Since the constraints from symmetry are all linear in \(I^0\), we can easily use the ALM algorithm described in Sect. 3.1, with minor modifications, to solve the new constrained optimization problem.

We have implemented a modified version of TILT which enforces the recovered low-rank component \(I^0\) to have reflective symmetry in both \(x\) and \(y\)-directions.\(^{22}\) Figure 14 (top) shows the result of the modified algorithm on a checker-board with reflective symmetry enforced. Notice that the converged region is indeed symmetric in both directions. Figure 14 bottom shows the new converged results of the same stop sign example (in Fig. 10) with reflective symmetry enforced.

5.3 TILT for Rotational Symmetry

Many other structural properties may be converted to a low-rank objective. For instance, the image of a rotationally symmetric pattern need not be a low-rank matrix, but it can be converted to one. To deal with rotational symmetry, we will consider circular windows, instead of rectangular ones. Each circular window is uniquely determined by its center and its radius. Clearly, the image region enclosed by such a window is not a matrix. However, it can be converted to one by considering a Frieze-expansion pattern (FEP) of the region (Liu et al. 2004; Lee and Liu 2010).

Suppose that a matrix \(I^0 \in \mathbb{R}^{m \times n}\) is the FEP of a circular window in an image with center at the origin and radius \(R\). Then, the mapping \(\tau\) between an entry \((x_0, y_0)\) in \(I^0\) and its corresponding pixel in the image is given by

\[ \tau(x_0, y_0) = H \cdot \begin{bmatrix} \frac{R x_0}{m} \cos \left( \frac{2\pi y_0}{n} \right) & \frac{R x_0}{m} \sin \left( \frac{2\pi y_0}{n} \right) \end{bmatrix}, \]  

(21)

If the center and radius of the circular window are chosen correctly, then the above FEP mapping gives rise to a low-rank matrix. However, in practice, the exact position of the window is not known a priori. In addition, there could be an additional deformation of the pattern due to the viewpoint. Figure 15(a) shows a representative input image. Suppose that we model the deformation by an affine transformation. Then, the mapping \(\tau\) from the low-rank matrix to the input image can be rewritten as

\[ \tau(x_0, y_0) = H \cdot \begin{bmatrix} \frac{R x_0}{m} \cos \left( \frac{2\pi y_0}{n} \right) \\ \frac{R x_0}{m} \sin \left( \frac{2\pi y_0}{n} \right) \end{bmatrix}, \]  

(22)

\(^{21}\)For reflective symmetry, \(g\) is its own inverse.
\(^{22}\)In order to allow the low-rank region to move freely to a symmetric region, we have to remove the constraints on the translation parameters.
where $H$ represents an affine transformation in homogenous coordinates. We can easily modify TILT to deal with the combined deformation of the FEP and the affine map and the algorithm can simultaneously recover the correct center of symmetry and the affine deformation. We show the results of such an algorithm on one rotationally symmetric pattern in Fig. 15.

6 Conclusions and Future Directions

In this paper, we introduce a novel framework in which an image is viewed as a matrix and the rank of the matrix is used as a measure of textural simplicity in the image window. We have introduced a very effective way of extracting precise structure and geometry of low-rank textures from their images using iterative convex optimization techniques. The proposed algorithm works effectively and robustly for a wide range of symmetric patterns and regular structures in real images, suggesting that the \textit{transformed low-rank plus sparse structures model} can be very useful for modeling real images of urban environments and man-made objects. More importantly, the proposed tools are highly complementary to existing vision techniques that mainly focus on using local features. By leveraging powerful modern high-dimensional optimization techniques, the new tools allow us to extract structural and geometric information accurately and robustly from large image regions in a holistic fashion. The transform invariant low-rank textures and the associated geometric transformations recovered by TILT can be very useful for many image processing and computer vision tasks such as image compression, matching, segmentation, symmetry detection, reconstruction of 3D models of urban environments, and recognition of man-made objects.

The proposed TILT scheme is still quite rudimentary in its formulation and solution. Many aspects of it can still be modified, improved, and extended. In the last section, we have demonstrated how, for a few cases, TILT can be customized or extended by incorporating additional structural constraints or by considering different deformation models. In principle, there should be little difficulty in generalizing TILT from linear (affine or projective) domain transforms to other classes of possibly more complex non-linear deformations. During the preparation of this manuscript, several follow-up works have shown that the same scheme can be extended to handle deformations incurred by an uncalibrated camera lens (Zhang et al. 2011b) or induced for low-rank textures on a curved surface (Zhang et al. 2011a). It has also been demonstrated in Mobahi et al. (2011) that TILT can be extended to handle multiple low-rank structures in a scene, and hence, can become a new powerful tool for 3D reconstruction of urban scenes.

Besides applications in image processing and computer vision, this work also opens up interesting avenues for research in other areas. On the theoretical side, although this work strongly leverages the convex relaxation proposed in Candès et al. (2011), there are no strong theoretical guarantees for recovery in literature for the specific linearization and convex relaxation used in TILT. Several research groups, including some of the authors of this work, are currently working on deriving precise conditions (on the type of signals and the deformation groups) under which the TILT-like procedures are guaranteed to succeed. Such theoretical guarantees could provide insights into the kinds of images and signals for which we should expect TILT to succeed and could also provide clues how we can further improve TILT for broader classes of structures and transformations. Having a low-rank matrix structure is only a sufficient but not necessary property for many symmetric patterns and regular structures. Therefore, exploring more general class of measures or objective functions for rectifying richer classes of patterns and structures despite large classes of geometric
deformation would be an exciting area for future research. More generally, work in this direction could lead to the discovery of rich (and seemingly unlimited) classes of region-level invariants that can be extracted effectively and efficiently from images in a similar holistic fashion, without relying on conventional local (less robust) features.

Another interesting area of research is on developing more efficient algorithms. In particular, we have seen that TILT requires solving a sequence of convex programs. As discussed in this paper, scalable first-order methods can be employed to develop very efficient algorithms for TILT. Besides the ALM algorithm discussed in this paper, another alternative is the recently proposed TFOCS algorithm (Becker et al. 2011), which has a similar structure. However, a major bottleneck of these first-order methods is that they require a Singular Value Decomposition (SVD) computation every iteration. This issue has been recently addressed by Liu et al. (2011) for cases where the rank of the matrix to be recovered is very small compared to its size. The proposed method also advocates parallel implementation wherever possible to further speed up the algorithm. It is quite possible that these methods can be easily extended to design more efficient and scalable algorithms for TILT that could meet the requirements of real-time applications.

In this paper, we have only discussed methods to rectify a transformed low-rank region, but we have not fully addressed the issue of detecting the location and scale of candidate low-rank regions in an image. The use of human intervention in the initialization of the low-rank region is a major limitation of the current solution. As some of our experiments have suggested, better (user) initializations can significantly improve the performance and applicability of TILT to a broader range of situations. This leaves plenty of room for future investigation on how to improve and augment TILT with other computer vision techniques such as image segmentation and salient region detection, which can either assist human users in initialization, or perhaps even detect and extract low-rank regions fully automatically.

Acknowledgements We thank the editor and the reviewers for their valuable and insightful comments. We would like to thank Professor John Wright of Columbia University for many stimulating discussions on this topic and for suggesting the catchy acronym “TILT” for the proposed method. We would also like to thank Dr. Zhouchen Lin of Microsoft Research Asia, Professor Emmanuel Candès of Stanford University, and Professor Xiaoming Yuan of Hong Kong Baptist University for sharing their insights on convex optimization algorithms during the preparation of this manuscript. This work was partially supported by the grants ONR Navy N00014-09-1-0230, NSF CCF 09-64215, DARPA HRL 10036-100471, and NSF IIS 11-16012.

Appendix: Linearization of Constraints

In Sect. 3.2, we proposed to impose two sets of constraints on the deformation parameters to make the solution well-defined so as to avoid some pathological solutions. Here, we show a detailed derivation or linearization of these constraints. In particular, we present here the derivation for the case when the transformation group is the set of affine transformations. The derivation for the homography case is very similar in case such constraints need to be imposed.

Constraints on Translation (17) Our first constraint is that the center of the rectangular window be fixed i.e., if \( x_0 = [x_0(1) \ x_0(2)]^T \) is the initial center of the window and \( \tau \) is the optimal transformation, then \( \tau(x_0) = x_0 \). Since the transformation is affine, we have that \( \tau(x) = Ax + b \), where \( A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \) is an invertible matrix and \( b \in \mathbb{R}^2 \). Suppose we parameterize our transformation vector as

\[
\tau = \begin{bmatrix} A_{11} \\ A_{21} \\ A_{12} \\ A_{22} \\ b \end{bmatrix},
\]

then in (17) we have

\[
A_I = \begin{bmatrix} x_0(1) & 0 & x_0(2) & 0 & 0 \\ 0 & x_0(1) & 0 & x_0(2) & 0 \end{bmatrix}.
\]

Constraints on Scale (18) The second constraint ensures that the area covered by the window as well as its aspect ratio does not change drastically. We show how this constraint results in a linear constraint in \( \Delta \tau \). For a given affine transformation, we note that the size of a rectangle gets scaled by the same amount. Thus, without any loss of generality, we assume that the initial window is a unit square with the points \((0, 0)\) and \((1, 1)\) forming opposite diagonal vertices. Once again, we represent the affine transformation by \( \tau(x) = Ax + b \). Let \( S(A, b) \) denote the area of the window after transformation. Since the area of the window is unchanged by translation, we denote the area as \( S(A) \). Let \( e_1 \) and \( e_2 \) denote two adjacent edges (with the origin as the common vertex) of the initial square. After transformation, these edges can be represented by the vectors \( e_1 = (A_{11}, A_{21}) \) and \( e_2 = (A_{12}, A_{22}) \). Then, the area of the transformed window is given by

\[
S(A) = \frac{1}{2} \|e_1\|\|e_2\| \sin \theta,
\]

where \( \cos \theta = \frac{\langle e_1, e_2 \rangle}{\|e_1\|\|e_2\|} \). The above equation can be simplified to

\[
S(A) = \sqrt{(A_{11}A_{22} - A_{12}A_{21})^2}.
\]

Now suppose that the matrix \( A \) is perturbed by a small amount \( \Delta A \). Since we require that the new area \( S(A + \Delta A) \) is close to \( S(A) \), we impose the constraint that the first-order
term in the Taylor series expansion of \( S(A + \Delta A) \) be zero i.e.,
\[
\nabla_A S(A) \cdot \Delta A = 0.
\]
(26)

We now consider the second part of the constraint which is to minimize the rate at which the aspect ratio of the window changes. Since the aspect ratio is unity for the initial window, we essentially require that \(|e_1| = |e_2|\) for the transformed window, using the same notation as above. We define \( C(A) = |e_1|^2 - |e_2|^2 \). Then, ideally, we require \( C(A + \Delta A) \) to be close to zero. Once again, we impose the constraint that the first-order term in the Taylor series expansion be zero i.e.,
\[
\nabla_A C(A) \cdot \Delta A = 0.
\]
(27)

Combining (26) and (27), and denoting \( \tau \) as a vector of all the transformation parameters, it is easy to see that we get a linear constraint of the form \( A, \Delta \tau = 0 \), as given in (18).

References

Baker, S., & Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3), 221–255.

Becker, S., Candès, E., & Grant, M. (2011). Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3), 165–218.

Bertsekas, D. P. (2004). *Nonlinear programming*. Belmont: Athena Scientific.

Cai, J., Candès, E., & Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4), 1956–1982.

Candès, E., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6), 717–772.

Candès, E., & Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5), 2053–2080.

Candès, E., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58(3), 1:11–1:37.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679–698.

Chandrasekaran, V., Sanghavi, S., Parrilo, P., & Willsky, A. (2011). Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2), 572–596.

Chang, S., Davis, L., Dunn, S., Eklundh, J., & Rosenfeld, A. (1987). Texture discrimination by projective invariants. *Pattern Recognition Letters*, 5, 337–342.

Chen, J. L., & Kundu, A. (1994). Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden Markov model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2), 208–214.

Cohen, F., Fan, Z., & Patil, M. (1991). Classification of rotated and scaled texture images using Gaussian Markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 192–202.

Cromme, L. (1978). Strong uniqueness: A far-reaching criterion for the convergence analysis of iterative procedures. *Numerische Mathematik*, 29, 179–193.

Do, M., & Vetterli, M (2002). Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden Markov models. *IEEE Transactions on Multimedia*, 4(4), 517–527.

Eckstein, J., & Bertsekas, D. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 53, 295–318.

Frey, B., & Jojic, N. (1999). Transformed component analysis: Joint estimation of spatial transformations and image components. In *Proc. of IEEE international conference on computer vision*.

Gabay, D., & Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications*, 2, 17–40.

Ganesh, A., Lin, Z., Wright, J., Wu, L., Chen, M., & Ma, Y. (2009). Fast algorithms or recovering a corrupted low-rank matrix. In *Proceedings of third international workshop on computational advances in multi-sensor adaptive processing*.

Garding, J., & Lindeberg, T. (1996). Direct computation of shape cues using scale-adapted spatial derivative operators. *International Journal of Computer Vision*, 17(2), 163–191.

Glöwinski, R., & Marroco, A. (1975). Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité, d’une classe de problèmes de Dirichlet non linéaires. *Revue Française d’Automatique, Informatique et Recherche Opérationnelle*, 9, 41–76.

Greenspan, H., Belongie, S., Goodman, R., & Perona, P. (1994). Rotation invariant texture recognition using a steerable pyramid. In *Proc. of international conference on pattern recognition*.

Haley, G., & Manjunath, B. (1999). Rotation-invariant texture classification using a complete space-frequency model. *IEEE Transactions on Image Processing*, 8, 255–269.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Proc. of 4th alvey vision conference*.

He, B. (2009). Parallel splitting augmented Lagrangian methods for monotone structured variational inequalities. *Computational Optimization and Applications*, 42(2).

He, B., Tao, M., & Yuan, X. (2011). Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, under-revision.

Jitorntrum, K., & Osborne, M. (1980). Strong uniqueness and second order convergence in nonlinear discrete approximation. *Numerische Mathematik*, 34, 439–455.

Kondepudy, R., & Healey, G. (1994). Using moment invariants to analyze 3-D color textures. In *Proc. of IEEE international conference on image processing*.

Kosecka, J., & Zhang, W. (2005). Extraction, matching, and pose recovery based on dominant rectangular structures. *CVGIP. Image Understanding*, 100(3), 274–293.

Lee, S., & Liu, Y. (2010). Skewed rotation symmetry group detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 1659–1672.

Leung, T., & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textures. *International Journal of Computer Vision*, 43(1), 29–44.

Levina, E., & Bickel, P. J. (2006). Texture synthesis and nonparametric resampling of random fields. *Annals of Statistics*, 34(4), 1751–1773.

Lin, Z., Chen, M., Wu, L., & Ma, Y. (2009). The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices (UIUC Technical Report UIUC-ENG-09-2215).

Liu, R., Lin, Z., Wei, S., & Su, Z. (2011). Solving principal component pursuit in linear time via l1 filtering (Preprint).

Liu, Y., & Collins, R. (2001). Skewed symmetry groups. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Liu, Y., Collins, R., & Tsin, Y. (2004). A computational model for periodic pattern perception based on Frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3), 354–371.
Liu, Y., Hel-Or, H., Kaplan, C. S., & Gool, L. V. (2010). Computational symmetry in computer vision and computer graphics. *Foundations and Trends in Computer Graphics and Vision, 5*(1–2), 1–195.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*, 91–110.

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. (2004). *An invitation to 3D vision*. Berlin: Springer.

Madiraju, S., & Liu, C. (1994). Rotation invariant texture classification using covariance. In *Proc. of IEEE international conference on image processing*.

Mikolajczyk, K., & Schmid, C. (2004). Scale and affine invariant interest point descriptors. *International Journal of Computer Vision, 60*(1), 63–86.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*, 1615–1630.

Mobahi, H., Zhou, Z., Yang, A., & Ma, Y. (2011). Holistic 3D reconstruction of urban structures from low-rank textures. In *Third international IEEE workshop on 3D representation and recognition*.

Morel, J. M., & Yu, G. (2009). ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences, 2*(2), 438–469.

Park, M., Lee, S., Chen, P., Kashyap, S., Butt, A., & Liu, Y. (2008). Performance evaluation of state-of-the-art discrete symmetry detection algorithms. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Park, M., Broklehurst, K., Collins, R., & Liu, Y. (2010). Translation-symmetry-based perceptual grouping with applications to urban scenes. In *Asian conference on computer vision*.

Peng, Y., Ganesh, A., Wright, J., Xu, W., & Ma, Y. (2010). RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Peng, Y., Ganesh, A., Wright, J., Xu, W., & Ma, Y. (2011). RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, to appear*.

Recht, B., Fazel, M., & Parillo, P. (2010). Guaranteed minimum rank solution of matrix equations via nuclear norm minimization. *SIAM Review, 52*(3), 471–501.

Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., & Delaert, F. (2008). Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Sundaramoorthi, G., Petersen, P., Varadarajan, V. S., & Soatto, S. (2009). On the set of images modulo viewpoint and contrast changes. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Toh, K., & Yun, S. (2010). An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6, 615–640.

Winder, S., & Brown, M. (2007). Learning local image descriptor. In *Proc. of IEEE conference on computer vision and pattern recognition*.

Wu, W. R., & Wei, S. C. (1996). Rotation and gray-scale transform-invariant texture classification using spiral resampling, subband decomposition and hidden Markov model. *IEEE Transactions on Image Processing, 5*, 1423–1434.

Yang, A., Huang, K., Rao, S., & Ma, Y. (2005). Symmetry-based 3-D reconstruction from perspective images. *Computer Vision and Image Understanding, 99*(2), 210–240.

Yuan, X., & Tao, M. (2011). Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization, 21*(1), 57–81.

Zhang, Z., Liang, X., & Ma, Y. (2011a). Unwrapping low-rank textures on generalized cylindrical surfaces. In *Proc. of IEEE international conference on computer vision*.

Zhang, Z., Matsushita, Y., & Ma, Y. (2011b). Camera calibration with lens distortion from low-rank textures. In *Proc. of IEEE conference on computer vision and pattern recognition*.