TILT: Transform Invariant Low-rank Textures *

Zhengdong Zhang[†], Xiao Liang[†], Arvind Ganesh[‡], and Yi Ma[†,‡]

[†]Visual Computing Group, Microsoft Research Asia, Beijing
{v-kelviz, v-ollian, mayi}@microsoft.com
[‡]Coordinated Science Lab, University of Illinois at Urbana-Champaign
abalasu2@illinois.edu

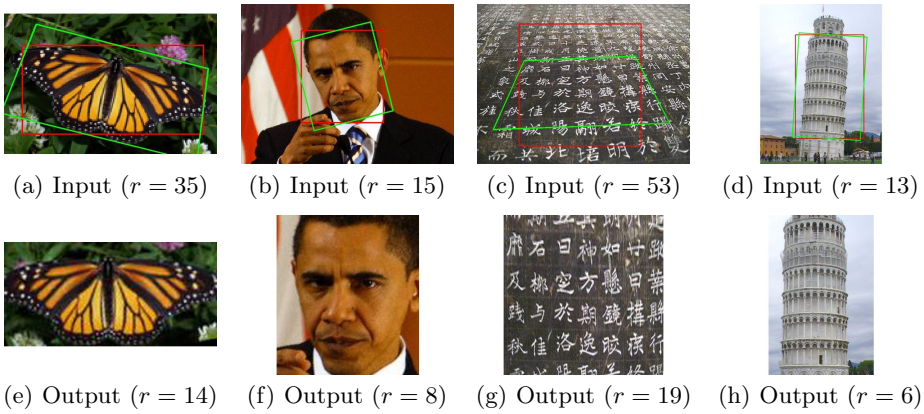
Abstract. In this paper, we show how to efficiently and effectively extract a rich class of low-rank textures in a 3D scene from 2D images despite significant distortion and warping. The low-rank textures capture geometrically meaningful structures in an image, which encompass conventional local features such as edges and corners as well as all 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 projective deformation, our method can accurately recover both the intrinsic low-rank texture and the precise domain transformation. Extensive experimental results demonstrate that this new technique works effectively for many near-regular patterns or objects that are approximately low-rank, such as human faces and text.

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 of almost all high-level vision tasks such as 3D reconstruction, object recognition, and scene understanding. Throughout the years, a large number of methods have been proposed in the computer vision literature for extracting various types of feature points or salient regions. The detected points or regions typically represent parts of the image which have distinctive geometric or statistical properties such as Canny edges, Harris corners, and textons.

One of the important applications of detecting feature points or regions is to establish correspondence or measure similarity across different images. For this purpose, 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 decade, numerous "invariant" features and descriptors have been proposed, studied, compared, and tuned in the literature (see [1, 2] and references therein). A widely used feature descriptor is the *scale invariant feature transform* (SIFT) [3], which to a large extent is invariant to changes in rotation and scale (*i.e.*, similarity transformations) and illumination. Nevertheless, if the images are shot from very different viewpoints, SIFT may fail to establish reliable correspondences and its affine-invariant version becomes a better choice [4, 5]. While deformation of a small distant patch can be well-approximated by an

* This work was supported by grants ONR N00014-09-1-0230, NSF CCF 09-64215, and NSF ECCS 07-01676.



(a) Input ($r = 35$) (b) Input ($r = 15$) (c) Input ($r = 53$) (d) Input ($r = 13$)

(e) Output ($r = 14$) (f) Output ($r = 8$) (g) Output ($r = 19$) (h) Output ($r = 6$)

Fig. 1. Low-rank Textures Automatically TILTed. From left to right: a butterfly; a face; a tablet of Chinese characters; and the Leaning Tower of Pisa. Top: windows with the red border are the original input, windows with the green border deformed texture returned by our method; Bottom: textures in the green window are matrices of much lower rank.

affine transform, projective transform becomes necessary to describe the deformation of a large region viewed through a perspective camera. 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.

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 ultimately: On the one hand, if we consider 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 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) [6]. The numerous "invariant" image features proposed in the vision literature, including the ones mentioned above, are at best approximately invariant, and often only to a limited extent. On the other hand, *the 3D scene is typically rich of regular structures that are full of invariants* (with respect to 3D Euclidean transformations). For instance, in an urban environment, the scene is typically filled with man-made objects that have parallel edges, right angles, regular shapes, symmetric structures, and repeated patterns. These geometric structures are rich of properties that are invariant under all types of subgroups of the 3D Euclidean group and as a result, their 2D (affine or perspective) images encode extremely rich 3D information about objects in the scene [7–9].

In this paper we propose a technique that aims to resolve the above dilemma about invariant features. We contend that instead of trying to seek invariants of the image that are either scarce or imprecise, we should

> *aim to directly detect and extract invariant structures of a scene through their images despite (affine or projective) domain transforms.*

Many methods have been developed in the past to detect and extract all types of regular, symmetric patterns from images under affine or projective transforms (see [10] for a recent evaluation). 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, most existing methods for detecting symmetric regions and patterns start by extracting and putting together local features such as SIFT points [9], corners, and edges [11]. As 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 features. Thus, we need a more general, effective, and robust way of detecting and extracting regular structures in images despite significant distortion and corruption.

Contributions of this Paper. In this paper, we aim to extract regions in a 2D image that correspond to a very rich class of regular patterns on a planar surface in 3D, whose appearance can be modeled as a "low-rank" matrix. 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. Clearly, an image of such a texture may be deformed by the camera projection and undergoes certain domain transformation (say affine or projective). The transformed texture in general is no longer low-rank in the image. Nevertheless, by utilizing advanced convex optimization tools from matrix rank minimization, we will show how to simultaneously recover such a low-rank texture from its deformed image and the associated deformation.

Our method directly uses raw pixel values of the image and there is no need of any pre-extraction of any low-level, local features such as corners, edges, SIFT, and DoG features. The proposed solution and algorithm are inherently robust to gross errors caused by corruption, occlusion, or cluttered background affecting a small fraction of the image pixels. Furthermore, our method applies to any image regions wherever such low-rank textures occur, regardless of the size of their spatial support. Thus, we are able to rectify not only small local features such as an edge and a corner but also large global symmetric patterns such as an entire facade of a building. We believe that this is a very powerful new tool that allows people to accurately extract rich structural and geometric information about the scene from its images, that are truly invariant of image domain transformations.

Organization of This Paper. The remainder of this paper is organized as follows: Section 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 usefulness of the extracted low-rank textures.

2 Transform Invariant Low-rank Textures

2.1 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 $\{I^0(x, y_0) \mid y_0 \in \mathbb{R}\}$ span a finite low-dimensional linear subspace *i.e.*,

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

for some small positive integer k. If r is finite, then we refer to I^0 as a rank-r texture. Figure 2 shows some ideal low-rank textures: a vertical or horizontal edge (or slope) can be considered as a rank-1 texture; and a corner can be considered as a rank-2 texture. By this definition, it is easy to see that the image of *regular symmetric patterns always lead to low-rank textures*.

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 $g(x, y) \doteq cI^0(ax + t_1, by + t_2)$ for some constants $a, b, c, t_1, t_2 \in \mathbb{R}_+$, then $g(x, y)$ 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 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 Section 3.2). Hence, in this paper, unless otherwise stated, we view two low-rank textures *equivalent* if they are scaled version of each other: $I^0(x, y) \sim cI^0(ax + t_1, by + t_2)$, for all $a, b, c, t_1, t_2 \in \mathbb{R}_+$.

In practice, we are never given the 2D texture as a continuous function in $\mathbb{R}^2$. Typically, we only have its values sampled on a finite discrete, say $m \times n$, grid in $\mathbb{Z}^2$. In this case, the 2D texture $I^0(x, y)$ is represented by an $m \times n$ real matrix. For a low-rank texture, we always assume that the size of the sampling grid is significantly larger than the intrinsic rank of the texture[1] *i.e.*,

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

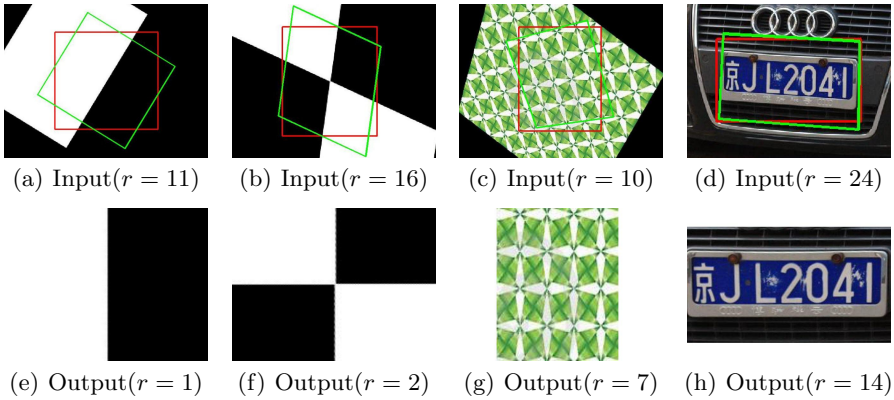
Thus, the 2D texture $I^0(x, y)$ (discretized) as a matrix has very low rank relative to its dimensions.

Remark 1 (Low-rank Textures versus Random Textures). Conventionally, the word "texture" is used to describe image regions that exhibit certain spatially stationary stochastic properties (e.g. grass, sand). Such a texture can be considered as a random sample from a stationary stochastic process [12] and is generally of full rank as a 2D function. The "low-rank textures" defined here are complementary to such random textures: It is supposed to describe regions in an image that have rather regular deterministic structures.

2.2 Deformed and Corrupted Low-rank Textures

In practice, we typically never see a perfectly low-rank texture in a real image, largely due to two factors: 1. the change of viewpoint usually induces a transformation on the domain of the texture function; 2. the sampled values of the

[1] It is easy to show that as long as the sampling rate is not one of the aliasing frequencies of the function I^0, the resulting matrix has the same rank as the continuous function.



(a) Input($r = 11$) (b) Input($r = 16$) (c) Input($r = 10$) (d) Input($r = 24$)

(e) Output($r = 1$) (f) Output($r = 2$) (g) Output($r = 7$) (h) Output($r = 14$)

Fig. 2. Representative examples of low-rank textures. 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 textures.

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 those factors and then seek ways to eliminate them.

Deformed Low-rank Textures. Although many surfaces or structures in 3D exhibit low-rank textures, their images do not! If we assume that such a texture $I^0(x, y)$ lies approximately 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 $I^0(x, y)$:

$$I(x, y) = I^0 \circ \tau^{-1}(x, y) = I^0 \left(\tau^{-1}(x, y)\right)$$

where $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ belongs to a certain Lie group $\mathbb{G}$. In this paper, we assume $\mathbb{G}$ is either the 2D affine group Aff(2) or the homography group $GL(3)$ acting linearly on the image domain.[2] In general, the transformed texture $I(x, y)$ as a matrix is no longer low-rank. For instance, a horizontal edge has rank one, but when rotated by 45°, it becomes a full-rank diagonal edge (see Figure 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 surrounding backgrounds. We can model such deviations as:

$$I = I^0 + E$$

for some error matrix E. As a result, the image I is potentially no longer 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:

[2] 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 family.

Problem 1 (Robust Recovery of Transform Invariant Low-rank Textures). 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:

$$\min_{I^0, E, \tau} \ \text{rank}(I^0) + \gamma \|E\|_0 \quad \text{subject to} \quad I \circ \tau = I^0 + E \tag{2}$$

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 rank and the error E of the fewest nonzero entries that agrees with the observation I up to a domain transformation τ. 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).[3]

Remark 2 (TILT versus Affine-Invariant Features). TILT is fundamentally different from the affine-invariant features or regions proposed in the literature [4, 5]. Essentially, those features are extensions to SIFT features in the sense that their locations are very much detected the same way as 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 [13]. Here 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 that have smooth parameterization). Probably most importantly, as we will see, our method is able to stratify all kinds of regions that are approximately low-rank (e.g. human faces, texts) and the results match extremely well with human perception.

Remark 3 (TILT versus RASL). We note that the optimization problem (2) is strikingly similar to the robust image alignment problem studied in [14], known as RASL. In some sense, TILT is a simpler problem as it only deals with one image and one domain transformation whereas RASL deals with multiple images and multiple transformations, one for each image. Thus, in the next section, we will follow a similar line of development to solve our problem as that in [14]. However, there are some important differences between TILT and RASL. For example, to make TILT work for a wide range of textures, we have to incorporate new constraints so that it achieves a large range of convergence. Moreover, we use a much faster convex optimization algorithm than the APG-based method used in [14], which will be described in the next section.

3 Solution by Iterative Convex Optimization

Although the formulation in (2) is intuitive, the rank function and the ℓ^0-norm are extremely difficult to optimize (in general NP-hard). Recent breakthroughs in convex optimization have shown that under fairly broad conditions, the cost

[3] By a slight abuse of terminology, we also refer to the procedure of solving the optimization problem as TILT.

function can be replaced by its convex surrogate [15]: the matrix nuclear norm $\|I^0\|_*$ (sum of all singular values) for $\text{rank}(I^0)$ and the ℓ^1-norm $\|E\|_1$ (the sum of absolute values of all entries) for $\|E\|_0$, respectively. As result, the objective function becomes:

$$\min_{I^0,E,\tau} \|I^0\|_* + \lambda\|E\|_1 \quad \text{subject to} \quad I \circ \tau = I^0 + E \tag{3}$$

where $\lambda > 0$ is a weighting parameter. Notice that although the objective function is now convex, the constraint $I \circ \tau = I^0 + E$ remains nonlinear in $\tau \in \mathbb{G}$. Theoretical considerations in [15] suggest that λ must be of the form $C/\sqrt{\max\{m,n\}}$, where C is a constant, typically set to unity, and $I^0 \in \mathbb{R}^{m \times n}$.

As suggested in [14], to deal with the nonlinear constraint effectively, we may assume that the deformation τ is small and so we can linearize the constraint $I \circ \tau = I^0 + E$ around its current estimate: $I \circ (\tau + \Delta\tau) \approx I \circ \tau + \nabla I \Delta\tau$, where ∇I represents the derivatives of the image w.r.t the transformation parameters.[4] Thus, locally the above optimization problem becomes a convex optimization subject to a set of linear constraints:

$$\min_{I^0,E,\Delta\tau} \|I^0\|_* + \lambda\|E\|_1 \quad \text{subject to} \quad I \circ \tau + \nabla I \Delta\tau = I^0 + E \tag{4}$$

As this linearization is only a local approximation to the original nonlinear problem, we solve it iteratively in order to converge to a (local) minima of the original problem. Although it is difficult to derive exact conditions under which this convex relaxation followed by linearization converges, in practice, we observe that the procedure does converge to a locally optimal solution, even when we start from a large initial deformation τ^0.

3.1 Fast Algorithm Based on Augmented Lagrangian Multiplier

In [14], the accelerated proximal gradient (APG) method was employed to solve the linearized problem (4). Recent studies have shown that the Augmented Lagrangian multiplier (ALM) method [16] is more effective for solving this type of convex optimization problems [15], and typically results in much faster convergence. For the sake of completeness, we will derive the ALM method to the linearized problem (4) and then summarize the overall algorithm for solving the original problem (3). We leave some detailed implementation issues for improving stability and range of convergence to the next subsection.

The Augmented Lagragian Multiplier method aims to solve the original constrained convex program (4) by instead minimizing the augmented Lagrangian given by:

$$L(I^0, E, \Delta\tau, Y, \mu) \doteq$$
$$\|I^0\|_* + \lambda\|E\|_1 + \langle Y, I \circ \tau + \nabla I \Delta\tau - I^0 - E\rangle + \frac{\mu}{2}\|I \circ \tau + \nabla I \Delta\tau - I^0 - E\|_F^2$$

[4] Strictly speaking, ∇I is a 3D tensor: it gives a vector of derivatives at each pixel whose length is the number of parameters in the transformation τ. When we "multiply" ∇I with another matrix or vector, it contracts in the obvious way which should be clear from the context.

where Y is a matrix of Lagrange multipliers, and $\mu > 0$ denotes the penalty for infeasible points. It is known from convex optimization literature [16] that the optimal solution to the original problem (4) can be effectively found by iterating the following two steps till convergence:

$$\begin{cases} (I^0_{k+1}, E_{k+1}, \Delta\tau_{k+1}) \leftarrow \min_{I^0,E,\Delta\tau} L(I^0, E, \Delta\tau, Y_k, \mu_k) \\ \mu_{k+1} \leftarrow \rho\mu_k, \;\; Y_{k+1} \leftarrow Y_k + \mu_k(I \circ \tau + \nabla I \Delta\tau_{k+1} - I^0_{k+1} - E_{k+1}) \end{cases} \tag{5}$$

for some $\rho > 1$.

In general, it might be expensive to find the optimal solution to the first step of (5) by minimizing over all the variables $I^0, E, \Delta\tau$ simultaneously. So in practice, to speed up the algorithm, we adopt an alternating minimization strategy as follows: [5]

$$\begin{cases} I^0_{k+1} \leftarrow \min_{I^0} L(I^0, E_k, \Delta\tau_k, Y_k, \mu_k) \\ E_{k+1} \leftarrow \min_{E} L(I^0_{k+1}, E, \Delta\tau_k, Y_k, \mu_k) \\ \Delta\tau_{k+1} \leftarrow \min_{\Delta\tau} L(I^0_{k+1}, E_{k+1}, \Delta\tau, Y_k, \mu_k) \end{cases} \tag{6}$$

Given the special structure of our Lagrangian function L, each of the above optimization problem has a very simple solution. Let $\mathcal{S}_t[\cdot]$ be the soft thresholding or *shrinkage* operator defined as follows:

$$\mathcal{S}_t(x) = \text{sign}(x) \cdot \max\{|x| - t, 0\} \tag{7}$$

where $t \geq 0$. When applied to vectors or matrices, the shrinkage operator acts element-wise. Suppose that $(U_k, \Sigma_k, V_k) \doteq \text{svd}(I \circ \tau + \nabla I \Delta\tau_k - E_k + \mu_k^{-1} Y_k)$. Then the optimization problems in (6) can be solved as follows:

$$\begin{cases} I^0_{k+1} \leftarrow U_k \mathcal{S}_{\mu_k^{-1}}[\Sigma_k] V_k^T \\ E_{k+1} \leftarrow \mathcal{S}_{\lambda\mu_k^{-1}}[I \circ \tau + \nabla I \Delta\tau_k - I^0_{k+1} + \mu_k^{-1} Y_k] \\ \Delta\tau_{k+1} \leftarrow (\nabla I^T \nabla I)^{-1} \nabla I^T (-I \circ \tau + I^0_{k+1} + E_{k+1} - \mu_k^{-1} Y_k) \end{cases} \tag{8}$$

We summarize the ALM approach to solving the problem in (3) as Algorithm 1.

3.2 Additional Constraints and Implementation Details

The previous section lays out the basic ALM algorithm for solving the TILT problem (3). However, there are a few caveats in applying it to real images of low-rank textures. In this section, we discuss some additional constraints which make the solution to the problem well-defined and some special implementation details that improve the range of convergence.

Constraints on the Transformations. As we have discussed in Section 2.1, 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 its value, scaling in each of the coordinates, and translation in each of the coordinates. Thus, in order for the problem to have a unique, well-defined optimal solution, we need to eliminate these ambiguities. In the first step of Algorithm 1, the intensity of the image is renormalized at each iteration in order to eliminate the scale ambiguity

[5] It can be shown that under fairly broad conditions, this does not affect the convergence of the algorithm.

Algorithm 1 (TILT via ALM)

Input: Initial rectangular window $I \in \mathbb{R}^{m \times n}$ in the input image, initial transformations τ in a certain group $\mathbb{G}$ (affine or projective), $\lambda > 0$.
While not converged **Do**
Step 1: normalize the image and compute the Jacobian w.r.t. transformation:

$$I \circ \tau \leftarrow \frac{I \circ \tau}{\|I \circ \tau\|_F}, \quad \nabla I \leftarrow \frac{\partial}{\partial \zeta}\left(\frac{I \circ \zeta}{\|I \circ \zeta\|_F}\right)\Big|_{\zeta=\tau};$$

Step 2: solve the linearized convex optimization (4):

$$\min_{I^0, E, \Delta\tau} \|I^0\|_* + \lambda\|E\|_1 \quad \text{subject to} \quad I \circ \tau + \nabla I \Delta\tau = I^0 + E,$$

with the initial conditions: $Y_0 = 0, E_0 = 0, \Delta\tau_0 = 0, \mu_0 > 0, \rho > 1, k = 0$:
While not converged **Do**

$$\begin{aligned}
(U_k, \Sigma_k, V_k) &\leftarrow \text{svd}(I \circ \tau + \nabla I \Delta\tau_k - E_k + \mu_k^{-1} Y_k),\\
I^0_{k+1} &\leftarrow U_k \mathcal{S}_{\mu_k^{-1}}[\Sigma_k] V_k^T,\\
E_{k+1} &\leftarrow \mathcal{S}_{\lambda\mu_k^{-1}}[I \circ \tau + \nabla I \Delta\tau_k - I^0_{k+1} + \mu_k^{-1} Y_k],\\
\Delta\tau_{k+1} &\leftarrow (\nabla I^T \nabla I)^{-1} \nabla I^T (-I \circ \tau + I^0_{k+1} + E_{k+1} - \mu_k^{-1} Y_k),\\
Y_{k+1} &\leftarrow Y_k + \mu_k (I \circ \tau + \nabla I \Delta\tau_{k+1} - I^0_{k+1} - E_{k+1}),\\
\mu_{k+1} &\leftarrow \rho\mu_k,
\end{aligned}$$

End While
Step 3: update transformations: $\tau \leftarrow \tau + \Delta\tau_{k+1}$;
End While
Output: I^0, E, τ.

in pixel value. Otherwise, the algorithm may tend to converge to a "globally optimal" solution by zooming into a black pixel.

To resolve the ambiguities in the domain transformation, we also need some additional constraints. For simplicity, we assume that the support of the initial image window Ω is a rectangle with the length of the two edges being $L(e_1) = a$ and $L(e_2) = b$, so that the total area $S(\Omega) = ab$. To eliminate the ambiguity in translation, we can fix the center x_0 of the window *i.e.*, $\tau(x_0) = x_0$. This imposes a set of linear constraints on $\Delta\tau$ given by :

$$A_t \Delta\tau = 0 \tag{9}$$

To eliminate the ambiguities in scaling the coordinates, we enforce (typically only for affine transforms) 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 conditions impose additional nonlinear constraints on the desired transformation τ in problem (3). As outlines earlier, we can linearize these constraints against the transformation τ and obtain another set of linear constraints on $\Delta\tau$:

$$A_s \Delta\tau = 0 \tag{10}$$

As a result, to eliminate both scaling and translation ambiguities, all we need to do is to add two sets of linear constraints to the optimization problem

(4). This results in very small modifications to Algorithm 1 to incorporate those additional linear constraints.[6]

Multi-Resolution and Branch-and-Bound. To further improve both the speed and the range of convergence, we adopt the popular multi-resolution approach in image processing. For the given image window I, we build a pyramid of images by iteratively blurring and downsampling the window by a factor of 2 until the size of the matrix reaches a threshold (say, less than 30×30 pixels[7]). Starting from the top of the pyramid, we apply our algorithm to the lowest-resolution image first and always initialize the algorithm with the deformation found from the previous level. We found that in practice, this scheme significantly improves the range of convergence and robustness of the algorithm since in the low-resolution images, small details are blurred out and the larger structures of the image drive the updates of the deformation. Moreover, it can speed up Algorithm 1 by hundreds of times. We tested the speed of our algorithm in MATLAB on a PC with a 3 Ghz processor. With input matrices of size 50×50, the average running time over 100 experiments is less than 6 seconds.

Apart from the multi-resolution scheme, we can make Algorithm 1 work for a large range of deformation by using a branch-and-bound approach. 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 transforms with no translation). A natural concern about such a branch-and-bound scheme is its effect on speed. Nevertheless, within the multi-resolution scheme, we only have to perform branch-and-bound at the 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 times 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 transform first and then use it to initialize the algorithm. We observed that with such an initialization, the branch-and-bound step becomes unnecesary for the projective transformation case.

Results in all examples and experiments shown in this paper are found by Algorithm 1 using both the multi-resolution and branch-and-bound schemes, unless otherwise stated.

4 Experiments and Applications

4.1 Range of Convergence of TILT

For most low-rank textures, Algorithm 1 has a fairly large range of convergence, even without using any branch-and-bound. To illustrate this, we show the result of the algorithm with a checkerboard image undergoing different ranges of affine transform: $y = Ax + b$, where $x, y \in \mathbb{R}^2$. We parameterize the affine matrix A as

[6] By introducing an additional set of Lagrangian multipliers and then appropriately revising the update equation associated with $\Delta\tau_{k+1}$.

[7] In order for the convex relaxation (3) to be tight enough, the matrix size cannot be too small. In practice, we find that our method works well for windows of size larger than 20×20.

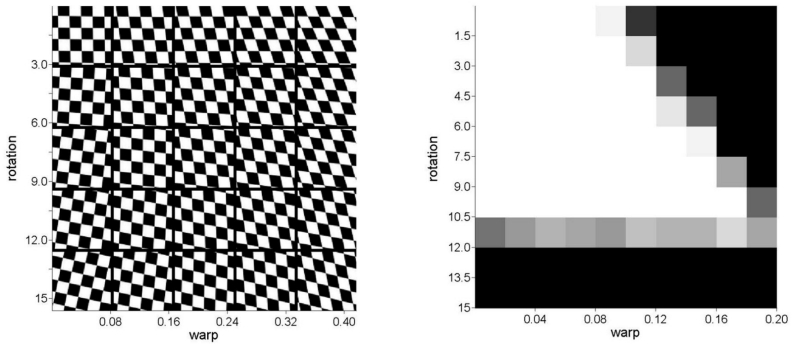
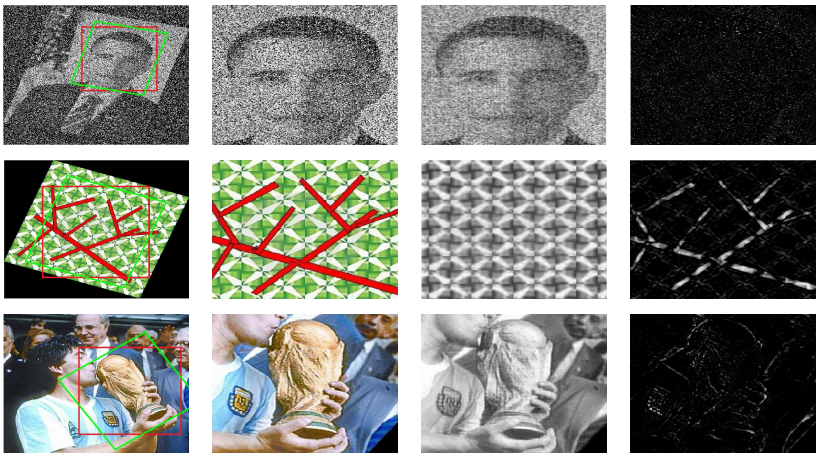


Fig. 3. Convergence of TILT. Left: representative input images in different regions; Right: the range of convergence (# of successes out of 20 random trials in each region).



(i) Input I (j) Output $I \circ \tau$ (k) Low rank I^0 (l) Sparse error E

Fig. 4. Robustness of TILT. Top: random corruption added to 60% pixels; Middle: scratches added on a symmetric pattern; Bottom: containing cluttered background.

$A(\theta, t) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$. We change (θ, t) within the range $\theta \in [0, \pi/6]$ with step size $\pi/60$, and $t \in [0, 0.3]$ with step size 0.03. We observe that the algorithm always converges up to $\theta = 10^\circ$ of rotation and skew (or warp) up to $t = 0.2$. Due to its rich symmetries and sharp edges, the checkerboard is a challenging case for "global" convergence since there are multiple local minima possible. In practice, we find that for most symmetric patterns in urban scenes (as shown in Figure 5), our algorithm converges for the entire tested range without any branch-and-bound.

4.2 Robustness of TILT

The results shown in Figure 4 demonstrate the striking robustness of the proposed algorithm to random corruptions, occlusions, and cluttered background, respectively. For the first two experiments, the branch-and-bound scheme was not used.

4.3 Shape from Low-rank Texture Detection

Obviously, the rectified low-rank textures found by our algorithm can better facilitate almost all high-level vision tasks than existing feature or texture detectors, including establishing correspondences among images, recognizing texts and objects, or reconstructing 3D shape or structure of a scene, etc. Due to limited space, we show a few examples in Figure 5 (left) to illustrate how our algorithm can extract rich geometric and structure information from an image of an urban scene.

The image size in this experiment is 1024×685 pixels and we use affine transformations on a grid of 60×60 windows to obtain the low-rank texture. If the rank of the resulting texture drops significantly from that of the original window, we say that the algorithm has "detected" a salient region.[8] In Figure 5, we have plotted the resulting deformed windows, together with the local orientation and surface normal recovered from the optimal affine transformation. Notice that for windows inside the building facades, our algorithm correctly recovers the local geometry for almost all of them; even for patches on the edge of the facades, one of its sides always aligns precisely with the building's edge.

Of course, one can initialize the size of the windows at different sizes or scales. But for larger regions, affine transformations will not be accurate enough to describe the deformation. In this case, we use projective transformations. For instance, the entire facade of the middle building in Figure 5 (left) obviously exhibits significant projective deformation. Nevertheless, if we initialize the projective TILT algorithm with the affine transform of 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 Figure 5 (middle).

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 using the true 3D orientation and the correct perspective. Figure 5 (right) illustrates this application with an example.

4.4 Rectifying Different Categories of Low-rank Textures

Since the proposed algorithm has a very large range of convergence for both affine and projective transformations and it is also robust to sparse corruptions, we observed that it works remarkably well for a very broad range of patterns, regular structures, and objects that arise in natural images or paintings. Figure 6 shows a few examples. We observe that with a simple initialization with a very rough rectangular window, our algorithm can converge precisely onto the underlying low-rank structures of the images, despite significant deformation.

References

1. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. PAMI **27** (2005) 1615–1630

[8] The image rank is computed by thresholding the singular values at 1/30 times the largest one. We also throw away regions whose largest singular value is too small, which typically correspond to a smooth region such as the sky.

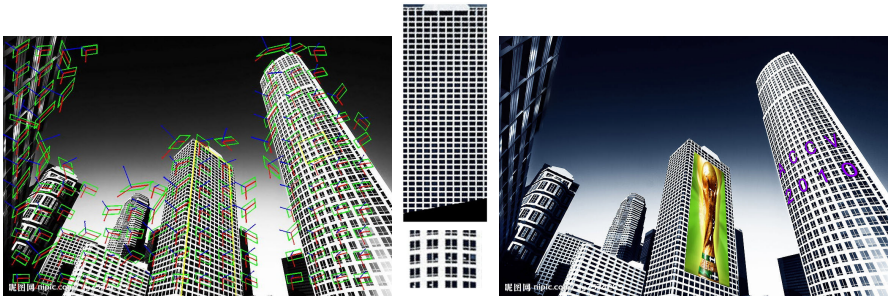


Fig. 5. Left: Low-rank textures detected by the TILT algorithm with affine transform on a grid of 60×60 windows and the recovered local affine geometry. Middle: low rank textures recovered by TILT with projective transform, which correspond to the regions marked with yellow lines; Right: the resulting image with the marked regions edited.

2. Winder, S., Brown, M.: Learning local image descriptor. In: Proc. of CVPR. (2007)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110
4. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point descriptors. IJCV **60** (2004)
5. Morel, J.M., Yu, G.: Asift: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences **2** (2009)
6. Sundaramoorthi, G., Petersen, P., Varadarajan, V.S., Soatto, S.: On the set of images modulo viewpoint and constrast changes. In: Proc. of CVPR. (2009)
7. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.: An Invitation to 3D Vision. Springer (2004)
8. Kosecka, J., Zhang, W.: Extraction, matching, and pose recovery based on dominant rectagular structures. CVGIP: Image Understanding **100** (2005) 274–293
9. Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., Dellaert, F.: Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In: Proc. of CVPR. (2008)
10. Park, M., Lee, S., Chen, P., Kashyap, S., Butt, A., Liu, Y.: Performance evaluation of state-of-the-art discrete symmetry detection algorithms. In: Proc. of CVPR. (2008)
11. Yang, A., Huang, K., Rao, S., Ma, Y.: Symmetry-based 3-D reconstruction from perspective images. Computer Vision and Image Understanding **99** (2005) 210–240
12. Levina, E., Bickel, P.J.: Texture synthesis and non-parametric resampling of random fields. Annals of Statistics **34** (2006) 1751–1773
13. Garding, J., Lindeberg, T.: Direct computation of shape cues using scale-adapted spatial derivative operators. IJCV **17** (1996) 163–191
14. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In: Proc. of CVPR. (2010)
15. Candès, E., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? preprint (2009)
16. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (2004)

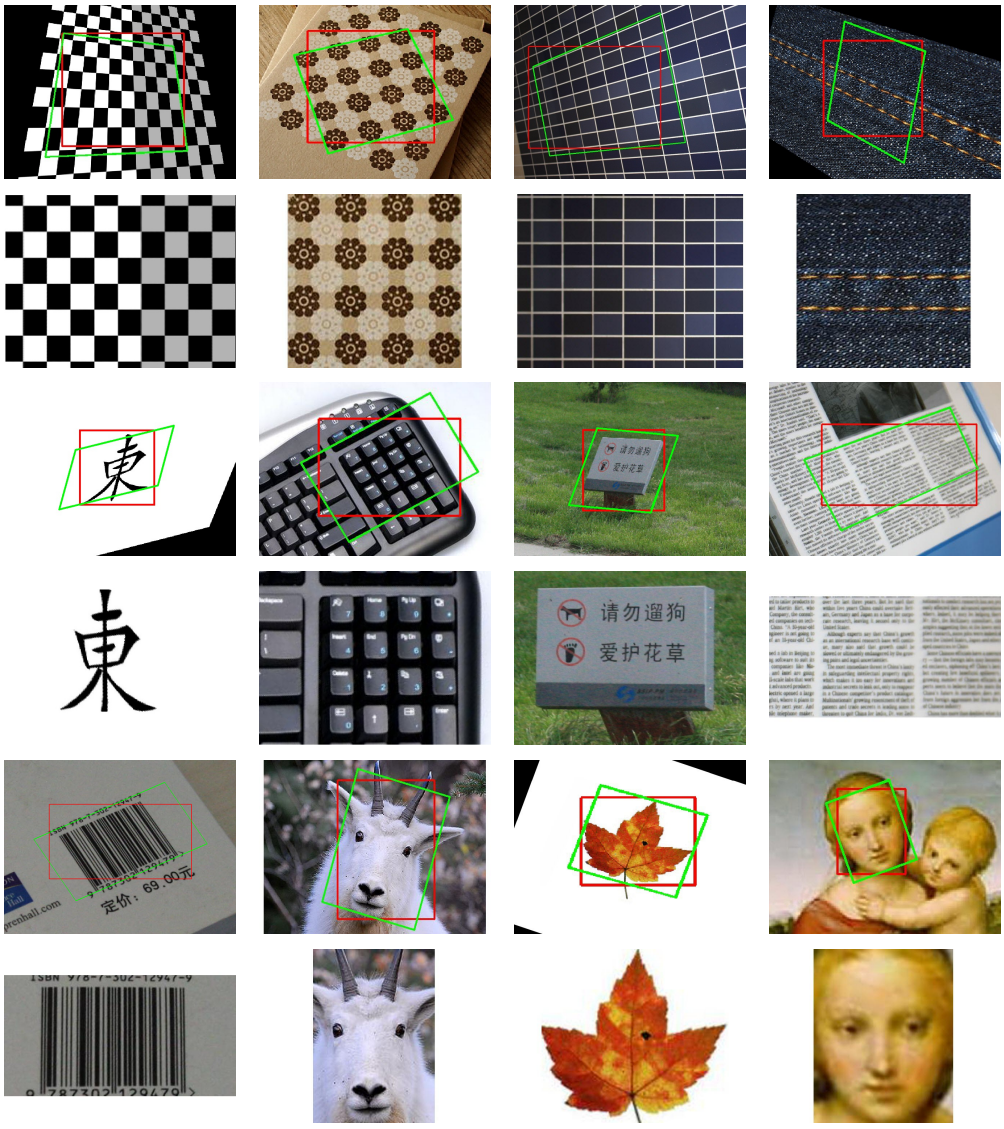


Fig. 6. Representative results of our method. Top: various patterns and textures; Middle: various texts and signs; Bottom: objects with bilateral symmetry.