Unwrapping Low-rank Textures on Generalized Cylindrical Surfaces

Zhengdong Zhang, Xiao Liang, Yi Ma
Visual computing Group, Microsoft Research Asia
zhangzdfaint@gmail.com, {v-ollian, mayi}@microsoft.com

Abstract

In this paper, we show how to reconstruct both 3D shape and 2D texture of a class of surfaces from a single perspective image. We consider the so-called the generalized cylindrical surfaces that are wrapped with low-rank textures. They can be used to model most curved building facades in urban areas or deformed book pages scanned for text recognition. Our method leverages on the recent new techniques for low-rank matrix recovery and sparse error correction and it generalizes existing techniques from planar surfaces to a much larger class of important 3D surfaces. As we will show with extensive simulations and experiments, the proposed algorithm can precisely rectify deformation of textures caused by both perspective projection and surface shape. It works for a wide range of symmetric or regular textures that are ubiquitous in images of urban environments, objects, or texts, and it is very robust to partial occlusion, noise, and saturation.

1. Introduction

One of the fundamental problems in computer vision is to recover from 2D images the 3D shape and pose of objects in a scene. In the past couple of decades, many effective algorithms and systems have been developed in the structure from motion (SfM) literature. Most of the methods require the use of multiple images and rely on correspondence of feature points, lines, or planes. Nowadays, an increasing number of applications require us to accurately recover 3D shape and pose of an object from a single image. For instance, from a single image taken by a mobile phone or an image on the Internet, we would like to recover the pose and texture of a building facade for recognition or augmented reality purposes; or from a single scanned copy, we would like to rectify the deformation of a book page and recognize the texts using conventional character recognition systems.

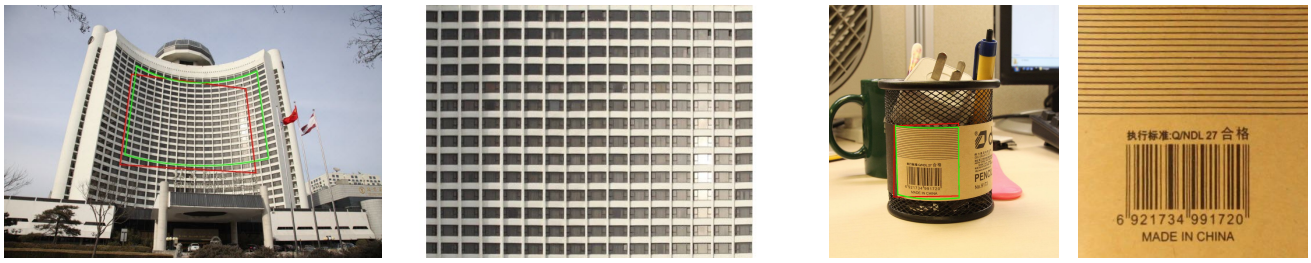
When there is only one image, additional priors are needed in order for the 3D recovery problem to be well defined. For instance, people have developed effective methods that make additional assumptions about the shape, say regular or symmetric [11, 8], or utilize additional information about the texture on the surface, say homogeneous random textures [3] or having certain low-rank structures [12], or actively acquire the shape of the surface using stereo or shape from lighting techniques [4], or simply make use of the known flat ground-truth and directly seek the deformation by optimization and searching techniques [10, 9].

In this paper, we consider the problem of recovering from a single image the shape and pose of a family of surfaces called *generalized cylindrical surfaces*. This is a very important family of shapes as they describe majority of curved building facades or deformed texts on curved surfaces, see Figure 1 for some examples. Such surfaces also have a very important property: Any generalized cylindrical surface is *isomorphic* to a planar surface.[1] All geometric and statistical properties of textures on the surface are preserved under such an isomorphism. Previous work has studied how to reconstruct such a surface if the textures have repetitive symmetric patterns as often seen in building facades [8, 7] or have a set of dominant parallel lines or salient boundaries as seen in scanned book pages [2, 6, 5].

While most existing techniques rely on statistical or geometrical relationships among local feature points or edges to rectify the surfaces, the recent advances in matrix rank minimization[1] have enabled people to effectively and efficiently harness global linear correlation in images. The work on *transform invariant low-rank textures* (TILT) [12] has shown that using such new tools, one can accurately recover the geometry of a planar surface given that the texture on the surface is low-rank. One distinctive feature of this new method is its holistic nature: it does not rely on any local features hence is very stable and robust to all kinds of nuisance factors in real images (see Figure 8 for some examples). In addition, the low-rank objective can harness many types of local or global regularities, including symmetry, parallelism, and orthogonality hence the method applies to a wide range of regular structures.

Contributions. In this paper, we show that the same rank-minimization technique can be applied to effectively rectify low-rank textures on arbitrary generalized cylindrical surfaces. This leads to a very efficient and effective algorithm

[1] Not every surface has this property. For instance, a sphere cannot be mapped to a plane without changing the metric on the surface.



(a) Input Image and Regions (b) Unwrapped Low-rank Texture (c) Input Image and Regions (d) Unwrapped Texture

Figure 1. **Unwrapping Textures from Curved Surfaces.** Images (a) and (c) are input images; Throughout this paper, windows with red border are the input region and windows with green border are the converged output of our method; The images (b) and (d) are the textures in the green window after the surface is unwrapped by our method. (**Images in this paper are best viewed in the electronic version.**)

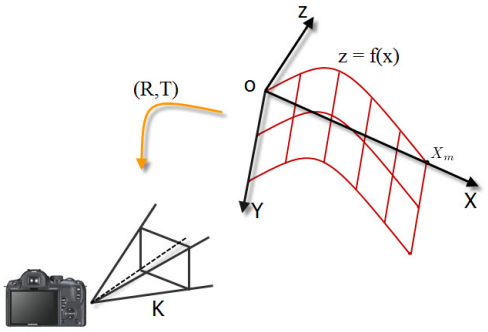


Figure 2. A generalized cylindrical surface C viewed by a perspective camera K.

that can accurately recover the 3D shape and 2D texture of the surface as well as the camera pose. As we will show with extensive simulations and experiments, the proposed algorithm works robustly for a fairly large range of view angle, surface deformation, initialization, and image quality. It handles both curved building facades and scanned documents in the same way, and enables a wide range of applications such as 3D reconstruction, augmented reality, and object (text) recognition.

2. Problem Formulation

In this section, we describe the mathematical model of our problem as well as the objective function used to solve the problem. Before we delve into the detailed mathematical formulation, Figure 2 illustrates the basic setup: a cylindrical surface, denoted as C, and its geometric pose, described by a rotation and translation pair $(R, T) \in SE(3)$, with respect to a perspective camera, with intrinsic parameters $K \in \mathbb{R}^{3\times3}$.

Model of the Curved Surface. We assume that our surface belongs to the class of *generalized cylindrical surfaces*. Mathematically, a surface is called a generalized cylindrical surface if it can be described as

$$\boldsymbol{c}(s,t) = t\boldsymbol{p} + \boldsymbol{h}(s) \quad \in \mathbb{R}^3,$$

where $s, t \in \mathbb{R}, \boldsymbol{p} \in \mathbb{R}^3$, $\boldsymbol{p} \perp \boldsymbol{h}'(s)$, and $\boldsymbol{h}(s) \in \mathbb{R}^3$ is generally a smooth function. So a generalized cylinder is a special class of *ruled surfaces* or *flat surfaces*.

Without loss of generality, we may choose a 3D coordinate frame (X, Y, Z) such that the center O is on the surface and Y-axis aligns with the direction of $\boldsymbol{p}$. We consider a "rectangular" section of the surface whose X-coordinate is in the interval $[0, X_m]$ (see Figure 2). Again without loss of generality, we may assume that the starting point of the section is O and the end point is $(X_m, 0, 0)$.[2] In this coordinate frame, the expression of the function $\boldsymbol{h}(\cdot)$ can be simplified and uniquely determined by a scalar function $Z = f(X)$. In this paper, we assume that the function can be modeled by a polynomial up to degree $d + 2$ (typically we choose $d \leq 4$ unless otherwise stated). So an explicit expression of the surface can be written as:

$$Z \doteq f_\tau(X) = X(X - X_m)\Big(\sum_{i=0}^{d} a_i X^i\Big), \qquad (1)$$

where we use $\tau = \{a_0, \ldots, a_d\}$ to denote the collection of parameters. Notice that when all a_i's are zero, the surface reduces to a planar surface $Z = 0$, which is considered by the original TILT paper [12]. We denote the so-defined surface as C_τ, using the subscript τ to indicate its dependency on the parameters τ.

Model of the Low-rank Texture on the Surface. Now we assume a 2D function $I_0(x, y)$ is defined on the surface C_τ between the section $X = 0$ and $X = X_m$ which models a texture wrapped on the surface.[3] Without loss of generality, we may assume that the y coordinate of the texture function is aligned with the Y-axis, and x is proportional to

[2] Note that we do not lose any generality by making this assumption as it helps fix the X-axis of the coordinate frame.

[3] In our implementation, X_m is always a fixed number, corresponding to the number of pixels of the unwrapped texture image (in the x-direction). In other words, to generate an texture image, we always uniformly take X_m samples along the surface in the x-direction.

(a) Input Image and Regions (b) Initial Input Window I (c) Unwrapped $I \circ g_\tau$ (d) Low-rank Part I_0 (e) Sparse Part E_0

Figure 3. **An Example for the Model** (6): a deformed low-rank texture I is unwrapped and decomposed into its low-rank part I_0 and sparse part E_0 by our algorithm.

the geodesic distance on the surface along the X direction.[4] Therefore, if $I_0(x, y)$ is a low-rank function as defined in [12], when the surface C_τ is unwrapped to a planar surface by an isomorphism, then I_0 represents a low-rank texture defined on that plane.

For any point (x, y) in the texture coordinates, we need to find its 3D coordinate (X_τ, Y_τ, Z_τ) on the cylindrical surface C_τ. Here we use the subscript τ to indicate the map's dependency on the parameters τ. Let $L_\tau \doteq \int_0^{X_m} \sqrt{1 + f_\tau'(X)^2} dX$ be the geodesic distance from the origin O to $(X_m, 0, 0)$ on the surface. Then the following set of equations uniquely determine the isomorphic (wrapping) map between (x, y) to (X_τ, Y_τ, Z_τ) that meets above specifications:

$$\begin{cases} x &= \frac{X_m}{L_\tau} \int_0^{X_\tau} \sqrt{1 + f_\tau'(X)^2} dX, \\ y &= Y_\tau, \\ Z_\tau &= f_\tau(X_\tau). \end{cases} \tag{2}$$

Model of the Camera. Suppose we have a camera with intrinsic parameters $K = \begin{pmatrix} f_x & \alpha & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3}$ and its position with respect to the surface coordinate frame (X, Y, Z) is specified by the Euclidean transform (R, T) where $R \in \mathbb{R}^{3\times 3}$ and $T \in \mathbb{R}^3$. With this camera, we take an image $I(u, v)$ of the surface C_τ.

A point (on the surface) with 3D coordinates (X_τ, Y_τ, Z_τ) is mapped to the image pixel coordinates (u, v) as

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} \frac{R_{1,1}X_\tau + R_{1,2}Y_\tau + R_{1,3}Z_\tau + T_1}{R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3} \\ \frac{R_{2,1}X_\tau + R_{2,2}Y_\tau + R_{2,3}Z_\tau + T_2}{R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3} \end{pmatrix}, \tag{3}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \begin{pmatrix} f_x x_n + \alpha y_n + o_x \\ f_y y_n + o_y \\ 1 \end{pmatrix}. \tag{4}$$

Recovery via Robust Rank Minimization. The image $I(u, v)$ is a transformed version of the low-rank texture $I_0(x, y)$ on the cylindrical surface C_τ. Let us denote the compounded map from the texture coordinate (x, y) to the image coordinate (u, v) as

$$g_\tau(x, y) : \ (x, y) \mapsto (u, v),$$

which depends on the parameters τ. Then ideally, we have $I \circ g_\tau = I_0$. However, in practice, the image can be a corrupted version of the low-rank texture, say due to occlusion etc, and this equation may not hold for a small fraction of the pixels. In other words, a more robust model for the imaging process is:

$$I \circ g_\tau = I_0 + E_0 \tag{5}$$

for some sparse matrix E_0. Figure 3 illustrates this model with a real example.

Therefore, given an image I, our goal is to recover all the unknown parameters τ (such as the polynomial coefficients), and rectify the image using g_τ so as to recover the low-rank texture I_0, despite some sparse errors E_0. Be aware that if the camera calibration K and pose (R, T) are unknown or only partially known, they are also part of the parameters τ that need to be recovered.

Similar to TILT, in order to recover the low-rank component I_0 and the sparse component E_0, we may aim to solve the following optimization problem:

$$\min_{\tau, L, S} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad I \circ g_\tau = L + S, \tag{6}$$

where L is the estimate for the low-rank component and S for the sparse component.

3. Solution and Algorithm Details

Although the objective function of the above optimization problem is convex, its constraint $I \circ g_\tau = L + S$ is highly nonlinear. Hence a common strategy is to linearize the constraint around the current estimate of the parameters τ and update the estimate iterative by solving the linearized version repeatedly:

$$\min_{\Delta\tau, L, S} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad I \circ g_\tau + J\Delta\tau = L + S, \tag{7}$$

where J is the Jacobian of the image against the unknown parameters τ. So, as long as we can evaluate the Jacobian at

[4] When the parameters of the curve change, the geodesic distance also changes. Nevertheless, such a parameterization always ensures that the unwrapping be isomorphic (up to a constant scale).

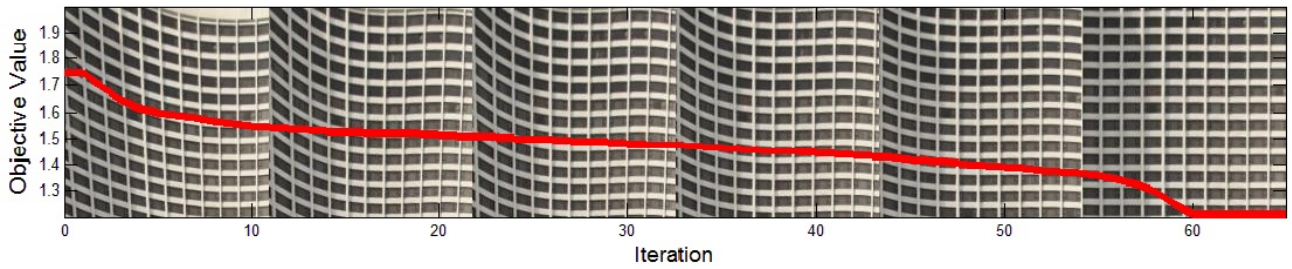


Figure 4. **Convergence Path for Optimizing (6).** We plot the value of the objective function against the number of iterations, as well as how the texture deforms at different stage of the process.

each iteration, the linearized problem is a convex program. We could use a similar ADM algorithm for TILT [12] for solving this convex program and then updating the estimate of $\tau \leftarrow \tau + \Delta\tau$ for the original problem (6). Figure 4 shows a representative convergence path of solving the optimization (6) for the image in Figure 1(a) with a very challenging initialization. With a more reasonable initialization, it normally takes much less (typically 20-30) iterations for the algorithm to converge. For a window of size 80×80, it takes about 10 to 20 seconds in our Matlab implementation.

3.1. Jacobian against Shape Parameters

Hence, the problem really boils down to how to evaluate the Jacobian of the image I against all the unknown parameters τ. Notice that the dependency of I on τ is all through the nonlinear map g_τ, hence the Jacobian is:

$$J = \frac{\partial I \circ g_\tau(x,y)}{\partial \tau} = \nabla I \cdot \frac{\partial g_\tau(x,y)}{\partial \tau}, \tag{8}$$

where ∇I is simply the gradient of the image I w.r.t. the image coordinates $g_\tau(x,y) = (u,v)$. It remains to compute:

$$\frac{\partial g_\tau(x,y)}{\partial \tau} = \frac{\partial}{\partial \tau}\begin{bmatrix} f_x x_n + \alpha y_n + o_x \\ f_y y_n + o_y \end{bmatrix} = \begin{bmatrix} f_x & \alpha \\ 0 & f_y \end{bmatrix}\begin{bmatrix} \frac{\partial x_n}{\partial \tau} \\ \frac{\partial y_n}{\partial \tau} \end{bmatrix}.$$

We continue to compute:

$$\begin{aligned}
\frac{\partial x_n}{\partial \tau} &= \frac{R_{1,1}\frac{\partial X_\tau}{\partial \tau} + R_{1,3}\frac{\partial Z_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)} \\
&\quad - \frac{R_{3,1}(R_{1,1}X_\tau + R_{1,2}Y_\tau + R_{1,3}Z_\tau + T_1)\frac{\partial X_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)^2} \\
&\quad - \frac{R_{3,3}(R_{1,1}X_\tau + R_{1,2}Y_\tau + R_{1,3}Z_\tau + T_1)\frac{\partial Z_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)^2}, \\
\frac{\partial y_n}{\partial \tau} &= \frac{R_{2,1}\frac{\partial X_\tau}{\partial \tau} + R_{2,3}\frac{\partial Z_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)} \\
&\quad - \frac{R_{3,1}(R_{2,1}X_\tau + R_{2,2}Y_\tau + R_{2,3}Z_\tau + T_1)\frac{\partial X_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)^2} \\
&\quad - \frac{R_{3,3}(R_{2,1}X_\tau + R_{2,2}Y_\tau + R_{2,3}Z_\tau + T_1)\frac{\partial Z_\tau}{\partial \tau}}{(R_{3,1}X_\tau + R_{3,2}Y_\tau + R_{3,3}Z_\tau + T_3)^2},
\end{aligned}$$

where one should notice that by choice Y_τ does not depend on τ.

Since $\frac{\partial Z_\tau}{\partial \tau} = \frac{\partial f_\tau(X_\tau)}{\partial \tau} = \frac{\partial f_\tau}{\partial \tau}|_{X=X_\tau} + f'_\tau(X_\tau)\frac{\partial X_\tau}{\partial \tau}$. So we only need to compute $\frac{\partial f_\tau}{\partial \tau}$ and $\frac{\partial X_\tau}{\partial \tau}$. From the definition of X_τ in (2), we have:

$$\int_0^{X_\tau} \sqrt{1 + f'_\tau(X)^2}\, dX = \frac{x}{X_m} L_\tau. \tag{9}$$

So differentiating both sides of (9) with respect to τ, we get:

$$\begin{aligned}
&\sqrt{1 + f'_\tau(X_\tau)^2}\frac{\partial X_\tau}{\partial \tau} + \int_0^{X_\tau} \frac{f'_\tau(X)}{\sqrt{1 + f'_\tau(X)^2}}\frac{\partial f'_\tau(X)}{\partial \tau} dX \\
= \quad & \frac{x}{X_m}\int_0^{X_m} \frac{f'_\tau(X)}{\sqrt{1 + f'_\tau(X)^2}}\frac{\partial f'_\tau(X)}{\partial \tau} dX.
\end{aligned}$$

So finally we have

$$\frac{\partial X_\tau}{\partial \tau} = \frac{\frac{x}{X_m}\int_0^{X_m} \frac{f'_\tau(X)}{\sqrt{1+f'_\tau(X)^2}}\frac{\partial f'_\tau(X)}{\partial \tau} dX - \int_0^{X_\tau} \frac{f'_\tau(X)}{\sqrt{1+f'_\tau(X)^2}}\frac{\partial f'_\tau(X)}{\partial \tau} dX}{\sqrt{1 + f'_\tau(X_\tau)^2}}, \tag{10}$$

where $\frac{\partial}{\partial \tau}$ against all the parameters $\{a_i\}$ in τ:

$$\begin{aligned}
\frac{\partial f_\tau(X)}{\partial a_i} &= X(X - X_m)X^i, \\
\frac{\partial f'_\tau(X)}{\partial a_i} &= (2X - X_m)X^i + X(X - X_m)iX^{i-1}.
\end{aligned}$$

By now we have all the ingredients for evaluating the Jacobian (8).

As we see from the above derivation, in order to evaluate the Jacobian, we need to evaluate two integrals in equation (10), which do not have closed-form expressions. Fortunately, we only need to compute the Jacobian once for each iteration at the current estimate τ. So we can simply evaluate these two integrals numerically. Compared to the rest of the algorithms where we need to compute SVD of the low-rank matrix L repeatedly, the cost for evaluating the two integrals is almost negligible and it does not affect the overall efficiency of the algorithm at all.

3.2. Updating Camera Parameters

In the above derivation, we have assumed that the camera parameters K and (R, T) are fixed. So we have to know K and (R, T) in advance.

Remember that as input to our algorithm, we require the user to specify a 4-sided polygon on the surface which are supposed to correspond to a "rectangular" region on the cylindrical surface as specified in Section 2. Ideally, this polygon should correspond to a rectangle in 3D, and we can use the orthogonality of its two sides to impose one linear constraint on the camera intrinsic parameters. Also, we can calculate the camera pose (R, T) from the homography between the rectangle and the image plane.

However, in practice, the four corners given by the user can be significantly off from such an ideal rectangular region. In such situation, we may well not even get a valid estimation of K, let alone (R, T). Fortunately focal length can be fetched from the EXIF information contained in the image file if it's taken by a modern digital camera. Further if we make fairly realistic assumption such as the pixel been square and the principle point being at the center of the image, we can obtain an approximate estimate of K, good enough for our purpose. Once K is known, the values for (R, T) can be derived from the homography between the rectangular region and the image. Be aware that the initial estimate of (R, T) can be incorrect as the initial window can be far off from a true rectangle in 3D. Hence we need to adjust them so that the window indeed corresponds to a rectangular region on the cylindrical surface. Thus, in our algorithm, we treat (R, T)[5] as part of the unknown parameters τ and update their estimates at each iteration too. The Jacobian of the image I against these parameters are fairly easy to derive and its has been used in the original work of TILT [12]. Due to the limit of space, we here do not elaborate.

As we will see from extensive simulation and experimental results, the algorithm has a significantly large region of convergence for the camera parameters. The initial input window does not have be very precise in order for the algorithm to converge onto an accurate solution.

4. Simulations and Experiments

In this section, we systematically evaluate the performance of the proposed algorithm on both synthetic and real images. Notice that our problem is highly nonlinear and the proposed algorithm is greedy in nature. So in order for the algorithm to be useful in practice, it needs to have a large range of convergence. We first provide careful simulations to validate the surprisingly large working range of the algorithm, then we test it on a variety of real images. Some comparisons with other related work and potential applications are also presented.

[5] We parameterize R by Rodrigues formula. To prevent the algorithm from shrinking to a point or exploding indefinitely, we assume that the distance between the camera and the surface is a fixed constant, i.e., the z-coordinate of T is fixed. This assumption is reasonable because it removes the ambiguity in the scale from a single image.

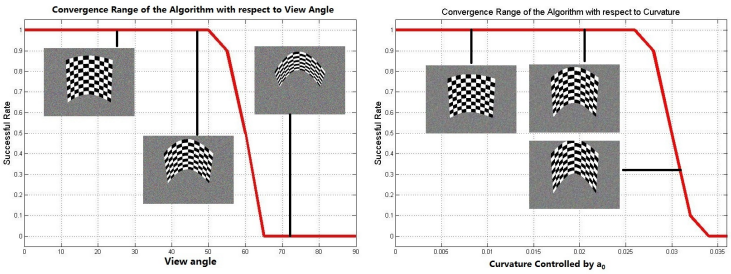


Figure 5. **Convergence Range of the Algorithm:** Success rates (y-axis) of the algorithm versus view angle (left) and curve curvature (right).

4.1. Evaluation on Synthetic Images

In this section, two simulations are carried out to examine the convergence range of the proposed algorithm. We synthesize a section of curved surface C_r with a standard checker-board texture (against a background of random noise). Standard ray-tracing techniques are used to generate all synthetic images of the surface from different viewpoints. See Figure 5 for examples.

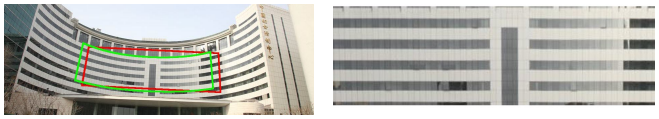
Range of View Angle. First we study the affect of changing the view direction. We first place the textured cylindrical surface C_r in front of the camera. The initial view direction is aligned with the surface normal, and then rotate the camera along the X-axis by an angle θ. As the rotation angle increases, we are simulating the effect of looking up a curved facade of a skyscraper with increasing tilt.

To enumerate θ, we divide $[0, \pi/2]$ uniformly into 18 intervals. In each interval i, we select N samples of θ according to i.i.d uniform distribution, then test on how many samples, say N_i, the algorithm converges to the ground truth. So N_i/N approximates the probability that the algorithm succeeds in region i.

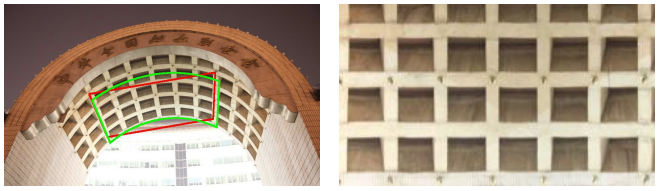
The plot of success rates is shown in Figure 5 left. From it, we can see that the algorithm succeeds with θ up to $60°$. Thus the algorithm tolerates a surprisingly large range of viewing direction.

Range of Curvature. Finally, we evaluate how "curved" a surface can be when our algorithm still rectifies it correctly. In this simulation, we change only one parameter that controls the curvature of a surface C_τ with $f_\tau(X) = X(X - X_m)(a_0 + a_1 X)$. We set a_1 to be a small and fixed value in order to give a realist shape to the curve. Then we enumerate a_0 in the range $[0, -0.04]$ and test the algorithm in a similar fashion with the previous simulations. The

(a) Input Image (b) Unwrapped (c) Input Image (d) Unwrapped



(e) Input Image (f) Unwrapped Texture



(g) Input Image (h) Unwrapped Texture

(i) Input Image (j) Top: initial; Bottom: unwrapped

Figure 6. **Rectification of Building Facades:** Images on the left column are the input, windows with red border are the original input, windows with green border are the output of our method; Images on the right are the recovered low-rank textures (in the green window after the deformation undone by our algorithm).

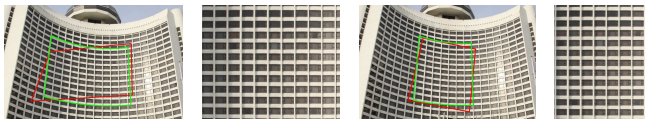
result is shown in Figure 5 right.

4.2. Experiments on Real Images

For all experiments, red windows are the input and green windows are the output.

Rectification of Curved Building Facades. We apply our algorithm to some representative images of curved building facades. The results are shown in Figure 6. We see that our method works well on a very diverse range of curved surfaces with all kinds of regular textures.

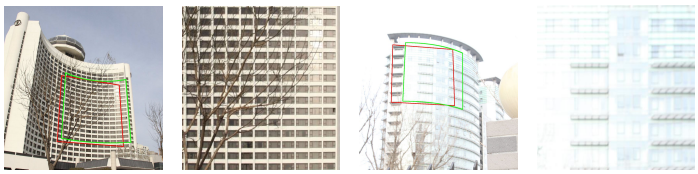
Stability to Initial Windows. One may be curious about how precise the user needs to specify the initial windows in order to obtain such good results that we have seen so far. For instance, what would happen if the user specifies a window far from a rectangular region on the surface? Using the same image we have seen in Figure 1(a), we show how the algorithm performs with different initial input windows in Figure 7.



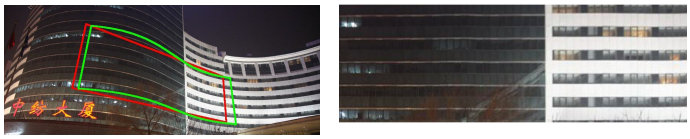
(a) Input (b) Unwrapped (c) Input (d) Unwrapped

(e) Input (f) Unwrapped (g) Input (h) Unwrapped

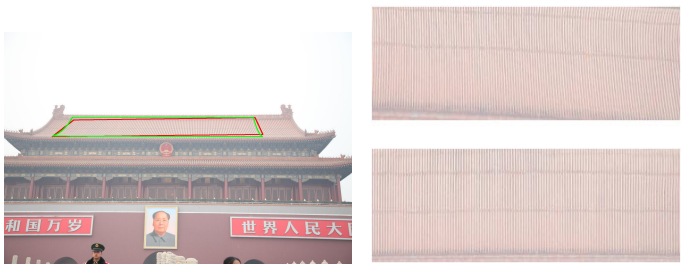
Figure 7. **Stability to Initial Windows.** These examples show the stability of our algorithm to initial windows. **(a)–(f)** show three examples that the algorithm succeeds despite very bad initializations. **(g)-(h)** shows the only failed case when the initial window is too narrow and extreme.



(a) Input Image (b) Unwrapped (c) Input Image (d) Unwrapped



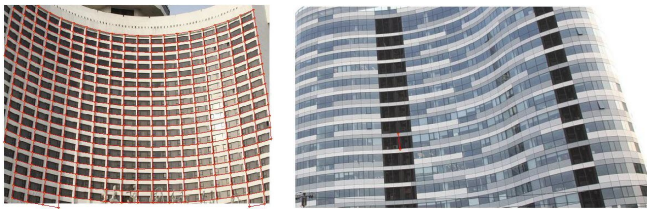
(e) Input Image (f) Unwrapped Texture



(g) Input Image (h) Top: initial; Bottom: unwrapped

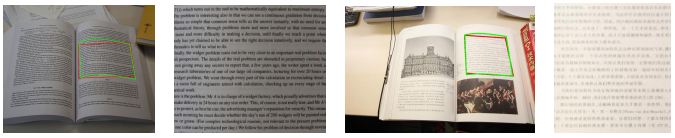
Figure 8. **Robustness of our Method.** This figure includes many challenging cases. The first one is partially occluded by tree branches. The second one is heavily over-exposed. The third one is taken at night and with two contrasting textures. The fourth one is very hazy. The results demonstrate the robustness of our algorithm to bad image quality.

Robustness to Corruptions. Examples shown in Figure 8 demonstrate the exceptional robustness of the proposed method: the same algorithm (with exact the same settings) works equally well for images that are over-exposed under bright daylight, under-exposed at night, noisy in a foggy day, or partially corrupted by occlusion.

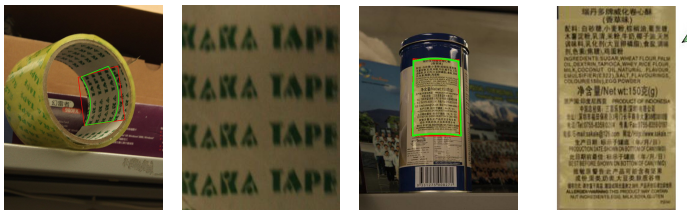


(a) Symmetry Lattice Detected (b) No Symmetry Lattice Detected

Figure 9. **Comparison with Symmetry Detection Method in [8]:** Left: a symmetry lattice detected by [8] on a building facade with clear repetitive patterns and benign view angle for the image in Figure 1(a); Right: but failed on a facade with a more general regular pattern for the image in Figure 3(a).



(a) Input Image (b) Unwrapped (c) Input Image (d) Unwrappe



(e) Input Image (f) Unwrapped (g) Input Image (h) Unwrapped

Figure 10. **Rectification of Deformed Texts on Curved Surfaces.**

Comparison with Symmetry Detection. Figure 9 shows the results of [8] on some of the images we have used. As we see, that method works on facades with clear repeated patterns, but failed on more general symmetric or regular patterns such as the one shown in Figure 3 and most of the images shown in Figure 6 and Figure 8. Many of the failures are due to lack of symmetry, large perspective distortion, or significant curvature. Since our method does not rely on symmetry detection, the results from our method can certainly benefit symmetry detection methods such as [8] since symmetry detection on the rectified textures would be an easier problem.

Rectification of Deformed Texts. Our algorithm not only works very well on all kinds of curved building facades, but also works on various deformed texts on curved surfaces such as texts on an opened book or labels on a bottle. Some representative results are shown in Figure 10. Clearly, our method can significantly improve the recognition performance of OCR engines for such deformed texts. Notice that from the examples on building facades and texts, our algorithm works for both convex and concave surfaces equally well.

Figure 11 compares our method with another recent method for rectifying documents [5]. That method relies on image segmentation and hence requires the region to be rectified has strong edges (such as the example shown in Figure 11 top). However, it does not work so well on images of texts whose boarders are not so evident (as the example in Figure 11 bottom shows). Nevertheless, our method relies on the regularity of the texture inside the region. In a way, our method is rather complementary to that work.

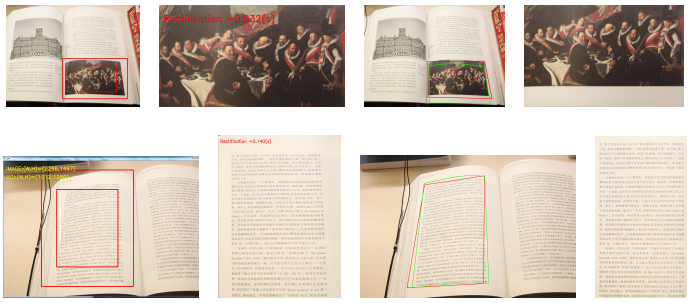


Figure 11. **Comparison with the Method in [5]:** Images on the left show the results of [5] on two examples: the bottom one is not precisely rectified. Images on the right show our results on the same images with similar inputs.

Figure 12. **Shape from Low-rank Textures**: 3D shape of the surface and camera pose (indicated by small pyramids) from a single image. Left: image in Figure 6(i) and Right: image in Figure 3(a).

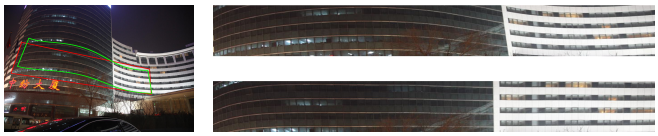
4.3. Applications and Extensions

Shape from (Low-rank) Textures. With all geometric information recovered from our algorithm, it's not difficult to retrieve the camera position (R, T) and the curved surface shape $\boldsymbol{c}(s,t)$ (from $f_\tau(X)$) in 3D that is consistent with the given image. Some representative results are shown in Figure 12. Hence, our method will certainly be useful for reconstructing 3D models of urban scenes where curved facades are abundant.

Augmented Reality. As we now can precisely estimate and then undo the distortion of the low-rank texture $I_0(x, y)$ caused by the perspective projection and the curved surface, we can perform many interesting editing or processing tasks on the given image and surface. For instance, we can superimpose virtual objects on the image or replace the texture of the surface while respecting all the camera and scene geometry. Please refer to Figure 13 for some illustrative examples.

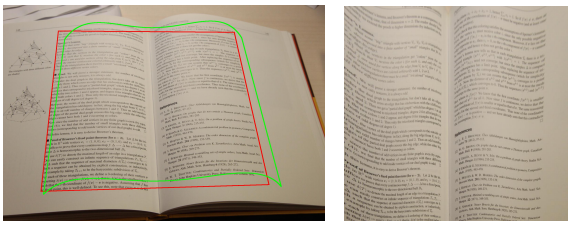
Figure 13. **Augmented Reality and Image Editing.** Superimpose new textures onto curved facades.

Challenges and Limitations. Although our algorithm works very robustly under very broad conditions and for images of a wide range of curved surfaces and textures, it may fail to generate accurate rectification if the conditions are too challenging or the polynomial surface model (1) is no longer valid. Figure 14 shows some of the cases: For the first example, the region is too large and the surface shape may have been beyond what can be approximated by the model – the unwrapping result is not as good as that in Figure 8 for a smaller region on the same surface. The second example is a facade consisting of three adjacent planes – our method approximates such a surface with a smooth surface and the rectification result is reasonable but not perfect. The third example contains two pieces of smooth surfaces and our model/algorithm failed to work on such an non-smooth surface. Hence, one possible topic for future study is how to extend the method to work on such piecewise-linear or piecewise-smooth surfaces, or even generalize to other classes of surfaces beyond generalized cylinders.



(a) Input Image (b) Top: initial; Bottom: unwrapped

(c) Input Image (d) Unwrapped Texture



(e) Input Image (f) Unwrapped Texture

Figure 14. **Results on Some Challenging Cases.** The algorithm does not rectify the deformation precisely and in some case failed.

5. Conclusions and Discussions

In this paper, we have presented a new method for rectifying low-rank textures on a generalized cylindrical surface from a single perspective image. The method relies on the new rank minimization techniques and has been evaluated with extensive simulations and experiments for its effectiveness and efficiency. Our method can accurately recover the shape of the surface and the pose of the camera hence it is very useful for 3D reconstruction of buildings. The recovered undistorted textures can be very useful for many tasks such as texture editing, synthesis, and text recognition. Our method currently is limited to handle a smooth cylindrical surface and in the future, we would like to investigate how to extend it to piecewise smooth surfaces.

References

[1] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *preprint*, 2009. 1

[2] K. Fujimoto, J. Sun, H. Takebe, M. Suwa, and S. Naoi. Curved paper rectification for digital camera document images by shape from parallel geodesics using continuous dynamic programming. In *Proc. of International Conference on Document Analysis and Recognition*, 2007. 1

[3] K. Ikeuchi. Shape from regular patterns. *Artificial Intelligence*, 22:49–75, 1984. 1

[4] K. Ikeuchi and B. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981. 1

[5] H. Koo and N. Cho. Rectification of figures and photos in document images using bounding box interface. In *CVPR*, 2010. 1, 7

[6] J. Liang, D. DeMenthon, and D. Doermann. Geometric rectification of camera-captured document images. *TPAMI*, 30:591–605, 2008. 1

[7] M. Park, K. Brocklehurst, R. Collins, and Y. Liu. Translation-symmetry-based perceptual grouping with applications to urban scenes. In *ACCV*, 2010. 1

[8] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *TPAMI*, 31:1804–1816, 2009. 1, 7

[9] M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-d tracking. In *ICCV*, 2007. 1

[10] Y. Tian and S. Narasimhan. A globally optimal data-driven approach for image distortion estimation. In *CVPR*, 2010. 1

[11] Y. M. W. Hong, Y. Yang. On symmetry and multiple view geometry: Structure, pose and calibration from a single image,. *IJCV*, 60(3):241–265, 2004. 1

[12] Z.Zhang, X. Liang, A. Ganesh, and Y. Ma. TILT: Transform invariant low-rank textures. In *ACCV*, 2010. 1, 2, 3, 4, 5