Int J Comput Vis (2013) 104:1–14
DOI 10.1007/s11263-013-0611-6

Linearized Alternating Direction Method with Adaptive Penalty and Warm Starts for Fast Solving Transform Invariant Low-Rank Textures

Xiang Ren · Zhouchen Lin

Received: 26 May 2012 / Accepted: 6 January 2013 / Published online: 24 January 2013
© Springer Science+Business Media New York 2013

Abstract Transform invariant low-rank textures (TILT) is a novel and powerful tool that can effectively rectify a rich class of low-rank textures in 3D scenes from 2D images despite significant deformation and corruption. The existing algorithm for solving TILT is based on the alternating direction method. It suffers from high computational cost and is not theoretically guaranteed to converge to a correct solution to the inner loop. In this paper, we propose a novel algorithm to speed up solving TILT, with guaranteed convergence for the inner loop. Our method is based on the recently proposed linearized alternating direction method with adaptive penalty. To further reduce computation, warm starts are also introduced to initialize the variables better and cut the cost on singular value decomposition. Extensive experimental results on both synthetic and real data demonstrate that this new algorithm works much more efficiently and robustly than the existing algorithm. It could be at least *five times faster* than the previous method.

Keywords Transform invariant low-rank texutres · Linearized alternating direction method with adaptive penalty · Warm start · Singular value decomposition

X. Ren
Department of Computer Science, University of Illinois at Urbana-champaign, Urbana, IL 61801, USA
e-mail: xren7@illinois.edu

Z. Lin (✉)
Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, Beijing 100871, People's Republic of China
e-mail: zlin@pku.edu.cn

1 Introduction

Extracting invariants in images is a fundamental problem in computer vision. It is key to many vision tasks, such as 3D reconstruction, object recognition, and scene understanding. Most of the existing methods start from low level local features, such as SIFT points (Schindler et al. 2008), corners, edges, and small windows, which are inaccurate and unrobust. Recently, Zhang et al. (2012b) proposed a holistic method called the transform invariant low-rank textures (TILT) that can recover the deformation of a relatively large image patch so that the underlying textures become regular (see Figs. 1a, b). This method utilizes the global structure in the image patch, e.g. the regularity, such as symmetry, rectilinearity and repetition, that can be measured by low rankness, rather than the low level local features, hence can be very robust to significant deformation and corruption.

TILT has been applied to many vision tasks and been extended for many computer vision applications. For example, Zhang et al. (2011b) used TILT to correct lens distortion and do camera calibration, without detecting feature points and straight lines. Zhang et al. (2012a) applied TILT to rectify texts in natural scenes to improve text recognition on mobile phones. Zhao and Quan (2011)) proposed a method for detecting translational symmetry using TILT. Zhang et al. (2011a) further generalized the transforms allowed by TILT to polynomially parameterized ones so that the shape and pose of low rank textures on generalized cylindrical surfaces can be recovered. Mobahi et al. (2011) used the low rank textures recovered by TILT as the building block for modeling urban scenes and reconstructing the 3D structure.

TILT was inspired by Robust Alignment by Sparse and Low-rank decomposition (RASL) (Peng et al. 2010), which has been successfully applied to align faces and video frames. TILT and RASL share the same mathematical model, namely

Fig. 1 TILT for rectifying textures. *Left column* The original image patches, specified by *red rectangles*, and the transforms found by TILT, indicated by *green quadrilaterals*. *Right column* Rectified textures using the transforms found. *Top row* Results by our method for solving TILT. *Bottom row* Results by the original method in Zhang et al. (2012b) for solving TILT. The original method does not converge to a correct solution. (Images in this paper are best viewed on screen!)

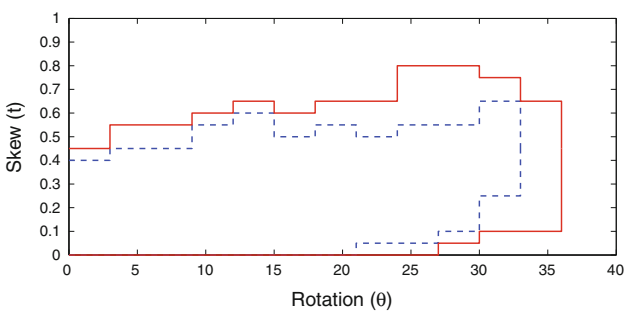

Fig. 2 Range of convergence for affine transform. The x-axis and y-axis stand for the rotation angle θ and the skew value t in an affine transform, respectively. Regions inside the curves are affine transforms that are successfully recovered in all trials. The *blue dashed curve* and the *red solid curve* are the boundaries of the parameters of affine transforms that can be successfully recovered by ADM and LADMAP, respectively

after an appropriate transform a data matrix can be decomposed into a low rank component and a sparse one (see (1)). The only difference between TILT and RASL is that the data matrix in TILT is an image patch, while that in RASL is multiple images or frames, each being a column of the matrix. So in the sequel, we just focus on TILT.

The existing most efficient solver for TILT is based on the alternating direction method (ADM) (Zhang et al. 2012b; Lin et al. 2009). It has been adopted by all the researchers that use TILT for their problems. However, it still requires multiple seconds to rectify a small sized image patch, making the applications of TILT far from being interactive. Another critical drawback of the existing solver is that it was derived out of intuition by simply mimicing the ADM for the robust principal component analysis (RPCA) problem presented in Lin et al. (2009). In the literature of ADM in the Gauss-Seidel

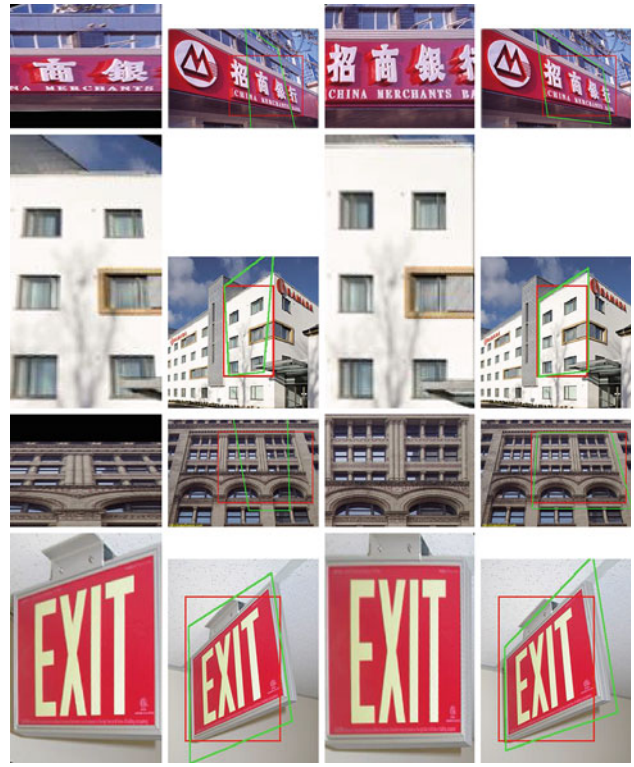

Fig. 3 Visual comparison between the results by ADM and LADMAP. *First and second columns* Rectification results by ADM. *Third and fourth columns* Results by LADMAP. *First and third columns* Rectified regions of interest. *Second and fourth columns* The initial transform (illustrated by *red rectangles* which are manually prescribed) and the computed transform (illustrated by *green quadrilaterals*)

fashion, almost all the convergence results were proven under the case of *two* variables, while the inner loop of TILT problem (see (3)) has *three* variables. Hence the convergence of ADM for the inner loop of TILT is not theoretically guaranteed. In our experiments, we did find many examples that the existing solver for TILT failed to produce a correct solution (e.g., see Figs. 1c, d, 2, 3 and 4). Consequently, to make the algorithm workable, its parameters have to be carefully tuned in order to trade off between convergence speed and robustness at the best (c.f. last paragraph of Sect. 2.2), which is difficult for different applications. The above drawbacks of the existing algorithm motivated us to design a more efficient algorithm for TILT, with a theoretical guarantee on the convergence of its inner loop.

We observe that the original method does not always converge to a correct solution because the inner loop of the original TILT problem have three variables. This motivates us to cancel one of the variables and solve an equivalent TILT problem that has only two variables in its inner loop. Then using the recently developed linearized alternating direction method with adaptive penalty (LADMAP) (Lin et al. 2011) and some good properties of the annihilation matrix, the inner loop can be solved efficiently and with a convergence guarantee. We further speed up the computation by warm starts in

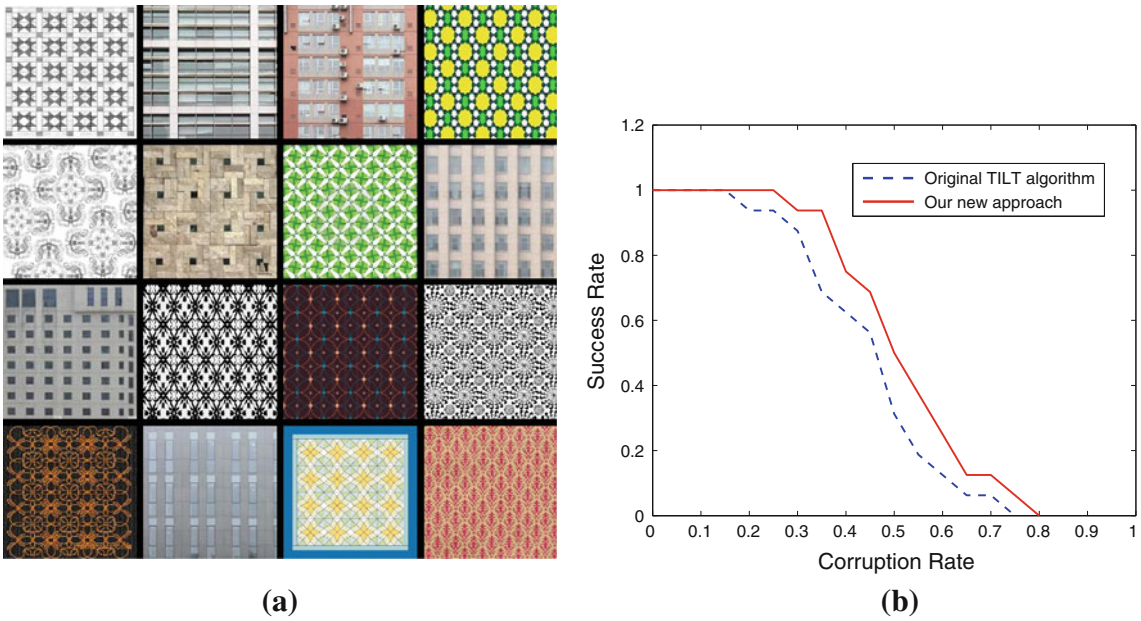

Fig. 4 Robustness of ADM and LADMAP under different levels of corruption. (**a**) Sample textures. (**b**) Success rates of ADM and LADMAP under different levels of corruption

two ways. First, we initialize the variables in the inner loop by their values when they exit the previous inner loop. This gives the variables very good initial values. So the number of iterations in the inner loop can be greatly cut. Second, as singular value decomposition (SVD) is the major bottleneck of computation, we also solve SVD by warm start, where the singular vectors and singular values are initialized as those in the previous iteration. The update of SVD is also based on a recently developed technique of optimization with orthogonality constraints (Wen and Yin 2013). As a result, our new algorithm, called LADMAP with warm starts (LADMAP+WS), can be at least *five times faster* than the original method and has a convergence guarantee for the inner loop.

The rest of this paper is organized as follows. In Sect. 2 we review the TILT problem and the ADM method for solving it. In Sect. 3 we introduce the LADMAP method for solving the inner loop of TILT and the variable warm start technique. In Sect. 4, we present some important details of applying LADMAP to TILT so that the computations can be made efficient. In Sect. 5, we show the warm start technique to compute SVD. Experimental results on both simulated and real data are reported in Sect. 6. Finally, we conclude our paper in Sect. 7.

2 Transform Invariant Low-rank Textures

In this section, we first briefly review the mathematical model of TILT. Then we introduce its corresponding convex surrogate and the existing ADM based method for solving it.

2.1 Mathematical Model

Consider a 2D texture as a matrix $A \in \mathbb{R}^{m\times n}$. It is called a low-rank texture if $r \ll \min(m,n)$, where $r = \operatorname{rank}(A)$. However, a real texture image is hardly an ideal low-rank texture, mainly due to two factors (1) it undergoes a deformation, e.g., a perspective transform from 3D scene to 2D image; (2) it may be subject to many types of corruption, such as noise and occlusion.

So if we could rectify a deformed texture D with a proper inverse transform τ and then remove the corruptions E, then the resulting texture A will be low rank. This inspires the following mathematical model for TILT (Zhang et al. 2012b):

$$\min_{A,E,\tau} \operatorname{rank}(A) + \lambda\|E\|_0, \quad s.t.\ D\circ\tau = A + E, \tag{1}$$

where $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ belongs to a certain group of transforms, e.g., affine transforms, perspective transforms, and general cylindrical transforms (Zhang et al. 2011a), $\|E\|_0$ denotes the number of non-zero entries in E, and $\lambda > 0$ is a weighting parameter which trades off the rank of the underlying texture and the sparsity of the corruption.

2.2 The Existing Method for Solving TILT

The above problem (1) is not directly tractable, because the rank of a matrix and the ℓ_0-norm are discrete and nonconvex, thus solving the optimization problem is NP-hard. As a common practice, rank and ℓ_0-norm could be replaced by the nuclear norm (Candés et al. 2011) (the sum of the singular values) and ℓ_1-norm (Donoho 2004) (the sum of the absolute

values of entries), respectively. This yields the following optimization problem with a convex objective function (Zhang et al. 2012b):

$$\min_{A,E,\tau} \|A\|_* + \lambda\|E\|_1, \quad s.t.\ D \circ \tau = A + E. \tag{2}$$

The above problem is *not* a convex program as the constraint is nonconvex. So Zhang et al. (2012b) proposed to linearize $D \circ \tau$ at the previous τ^i as $D \circ (\tau^i + \Delta\tau) \approx D \circ \tau^i + J\Delta\tau$ and determine the increment $\Delta\tau$ in the transform by solving

$$\min_{A,E,\Delta\tau} \|A\|_* + \lambda\|E\|_1, \quad s.t.\ D \circ \tau^i + J\Delta\tau = A + E, \tag{3}$$

where J is the the Jacobian: derivative of the image with respect to the transform parameters. J is full column rank. We call the iterative procedure to solve (3) the inner loop for solving TILT. When the increment $\Delta\tau$ is computed, the transform is updated as

$$\tau^{i+1} = \tau^i + \Delta\tau.$$

In the following, when we are focused on the inner loop, for simplicity we may omit the superscripts $i+1$ or i of variables. This should not cause ambiguity by referring to the context.

Zhang et al. (2012b) proposed an ADM based method to solve (3). It is to minimize the augmented Lagrangian function of problem (3):

$$\begin{aligned} L(A, E, \Delta\tau, Y, \mu) = \|A\|_* + \lambda\|E\|_1 + \langle Y, D \circ \tau + J\Delta\tau \\ -A - E\rangle + \frac{\mu}{2}\|D \circ \tau + J\Delta\tau \\ -A - E\|_F^2, \end{aligned}$$

with respect to the unknown variables *alternately* (hence the name ADM), where Y is the Lagrange multiplier, $\langle A, B\rangle \equiv \mathrm{tr}(A^T B)$ is the inner product, $\|\cdot\|_F$ is the Frobenius norm, and $\mu > 0$ is the penalty parameter. The ADM in Zhang et al. (2012b) goes as follows:

$$A_{k+1} = \underset{A}{\operatorname{argmin}}\, L(A, E_k, \Delta\tau_k, Y_k, \mu_k), \tag{4}$$

$$E_{k+1} = \underset{E}{\operatorname{argmin}}\, L(A_{k+1}, E, \Delta\tau_k, Y_k, \mu_k), \tag{5}$$

$$\Delta\tau_{k+1} = \underset{\Delta\tau}{\operatorname{argmin}}\, L(A_{k+1}, E_{k+1}, \Delta\tau, Y_k, \mu_k), \tag{6}$$

$$\begin{aligned} Y_{k+1} &= Y_k + \mu_k(D \circ \tau + J\Delta\tau_{k+1} - A_{k+1} - E_{k+1}), \\ \mu_{k+1} &= \rho\mu_k, \end{aligned}$$

where $\rho > 1$ is a constant. All subproblems (4)–(6) have closed form solutions (Zhang et al. 2012b) (cf. (24) and (26)). More complete details of the algorithm can be found as Algorithm 1 in Zhang et al. (2012b).

Although the above ADM in the Gauss-Seidel fashion empirically works well in most cases, there is no theoretical guarantee on its convergence. There are two factors that may result in its non-convergence. Namely, all the previous convergence results of ADM in the Gauss-Seidel fashion were proven under the conditions that the number of unknown variables are only *two*[1] and the penalty parameter is *upper bounded*. In contrast, the inner loop of TILT has three variables and its penalty parameter is not upper bounded. As one will see, the naive ADM algorithm above may not produce a correct solution. In particular, the choice of ρ is critical to influence the number of iterations in the inner loop and the quality of solution. If ρ is small, there will be a lot of iterations but the solution is very likely to be correct. If ρ is large, the number of iterations is small but an incorrect solution may be produced. So one has to tune ρ very carefully so that the number of iterations and the quality of solution can be best traded off. This is difficult if the textures are different. So in this paper we aim at addressing both the computation speed and the convergence issue of the inner loop in the original method.

3 Solving the Inner Loop by LADMAP

In this section, we show how to apply a recently developed method LADMAP to solve the inner loop of TILT. We first reformulate (3) so that $\Delta\tau$ is canceled and hence LADMAP can be applied.

3.1 Sketch of LADMAP

LADMAP (Lin et al. 2011) aims at solving the following type of convex programs:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}), \quad s.t.\ \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) = \mathbf{c}, \tag{7}$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{c}$ could be either vectors or matrices, f and g are convex functions, and $\mathcal{A}$ and $\mathcal{B}$ are linear mappings.

If the original ADM is applied to (7), $\mathbf{x}$ and $\mathbf{y}$ are updated by minimizing the augmented Lagrangian function of (7), resulting in the following updating scheme:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\mu_k}{2}\|\mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c} + \gamma_k/\mu_k\|^2, \tag{8}$$

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + \frac{\mu_k}{2}\|\mathcal{B}(\mathbf{y}) + \mathcal{A}(\mathbf{x}_{k+1}) - \mathbf{c} + \gamma_k/\mu_k\|^2, \tag{9}$$

$$\gamma_{k+1} = \gamma_k + \mu_k[\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c}], \tag{10}$$

where γ is the Lagrange multiplier and μ is the penalty parameter. In many problems, f and g are vector or matrix norms, and the subproblems (8) and (9) have closed-form solutions when $\mathcal{A}$ and $\mathcal{B}$ are identity mappings (Cai et al. 2010; Donoho

[1] For the case of more than two variables, ADM with some appropriate modifications, such as introducing a dummy variable and treating all the original variables as a super block (Bertsekas and Tsitsiklis 1997) or introducing a Gaussian back substitution (He et al. 2012), can be proven to converge.

1995; Liu et al. 2010; Yang and Zhang 2011), hence easily solvable. However, when $\mathcal{A}$ and $\mathcal{B}$ are not identity mappings, subproblems (8) and (9) may not be easy to solve. So Lin et al. (2011) proposed linearizing the quadratic penalty term in the augmented Lagrangian function and adding a proximal term in (8) and (9) for updating $\mathbf{x}$ and $\mathbf{y}$, resulting in the following updating scheme:

$$\begin{aligned}\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\mu_k \eta_A}{2}\|\mathbf{x}-\mathbf{x}_k \\ &\quad + \mathcal{A}^*(\gamma_k + \mu_k(\mathcal{A}(\mathbf{x}_k)+\mathcal{B}(\mathbf{y}_k)-\mathbf{c}))/(\mu_k\eta_A)\|^2, \quad (11)\\ \mathbf{y}_{k+1} &= \arg\min_{\mathbf{y}} g(\mathbf{y}) + \frac{\mu_k \eta_B}{2}\|\mathbf{y}-\mathbf{y}_k \\ &\quad + \mathcal{B}^*(\gamma_k + \mu_k(\mathcal{A}(\mathbf{x}_{k+1})+\mathcal{B}(\mathbf{y}_k)-\mathbf{c}))/(\mu_k\eta_B)\|^2, \quad (12)\end{aligned}$$

and λ is still updated as (10), where $\eta_A > 0$ and $\eta_B > 0$ are some parameters. Then one can see the new subproblems (11) and (12) can have closed-form solutions again when f and g are norms. Lin et al. also proposed a strategy to adaptively update the penalty parameter μ [2] and proved that when μ is non-decreasing and upper bounded, and $\eta_A > \|\mathcal{A}\|^2$ and $\eta_B > \|\mathcal{B}\|^2$, $(\mathbf{x}_k, \mathbf{y}_k)$ converges to an optimal solution to (7), where $\|\mathcal{A}\|$ and $\|\mathcal{B}\|$ are the operator norms of $\mathcal{A}$ and $\mathcal{B}$, respectively. For more details, please refer to Lin et al. (2011).

3.2 Reformulating the Inner Loop of TILT

As almost all existing convergence results of ADM or linearized ADM in the Gauss-Seidel fashion are proven in the case of two variables, while the inner loop of TILT has three variables, we aim at canceling one variable by taking advantage of the special structure of the problem.

We notice that $\Delta\tau$ does not appear in the objective function of (3) and it is easy to see that if (A^*, E^*) is an optimal solution, then the optimal $\Delta\tau$ must be

$$\Delta\tau^* = (J^T J)^{-1} J^T (A^* + E^* - D \circ \tau), \quad (13)$$

as $(A^*, E^*, \Delta\tau^*)$ must satisfy the linear constraint in (3). So we have to find an optimization problem to obtain (A^*, E^*).

By plugging (13) into the linear constraint in (3), one can see that (A^*, E^*) must satisfy the following constraint:

$$J^\perp A + J^\perp E = J^\perp (D \circ \tau), \quad (14)$$

where $J^\perp = I - J(J^T J)^{-1} J^T$ and I is the identity matrix. So we have an optimization problem for (A^*, E^*):

$$\min_{A,E} \|A\|_* + \lambda\|E\|_1, \quad s.t.\ J^\perp A + J^\perp E = J^\perp(D \circ \tau). \quad (15)$$

We can prove the following proposition.

Proposition 1 *Problems (3) and (15) have the same optimal solution* (A^*, E^*).

Proof We only have to prove that the constraints for (A, E) do not change. The constraints for (A, E) in (3) and (15) can be written as

$$S_1 = \{(A, E) | \exists \Delta\tau \text{ such that } D \circ \tau + J\Delta\tau = A + E\},$$

and

$$S_2 = \{(A, E) | J^\perp A + J^\perp E = J^\perp (D \circ \tau)\},$$

respectively. If $(A, E) \in S_1$, then we can multiply $J^\perp$ to both sides of $D \circ \tau + J\Delta\tau = A + E$ to see that $(A, E) \in S_2$. If $(A, E) \in S_2$, then $D \circ \tau - A - E \in \text{null}(J^\perp)$. Since $\text{null}(J^\perp) = \text{span}(J)$, there exists $\Delta\tau$ such that $D \circ \tau - A - E = J\Delta\tau$. Thus $(A, E) \in S_1$. □

One can easily check that $J^\perp$ has the following nice properties:

$$(J^\perp)^T = J^\perp \text{ and } J^\perp J^\perp = J^\perp. \quad (16)$$

Moreover, denote the operator norm of $J^\perp$ as $\|J^\perp\|$. Then we have

Proposition 2 $\|J^\perp\| = 1$.

Proof From (16) we have $(J^\perp)^2 = J^\perp$. So the eigenvalues λ of $J^\perp$ satisfies $\lambda^2 - \lambda = 0$. Thus, the eigenvalues of $J^\perp$ are either 1 or 0. As $J^\perp$ is symmetric, $\|J^\perp\| = 1$. □

Applying LADMAP ((11), (12), and (10)) directly to (15) (where $\mathbf{x}$, $\mathbf{y}$, and γ are A, E, and Y, respectively), with some algebra we have the following updating scheme:

$$A_{k+1} = \operatorname*{argmin}_A \|A\|_* + \frac{\mu_k}{2}\|A - M_k\|_F^2, \quad (17)$$

$$E_{k+1} = \operatorname*{argmin}_E \lambda\|E\|_1 + \frac{\mu_k}{2}\|E - N_k\|_F^2, \quad (18)$$

$$Y_{k+1} = Y_k + \mu_k J^\perp (A_{k+1} + E_{k+1} - D \circ \tau), \quad (19)$$

$$\mu_{k+1} = \min(\mu_{\max}, \rho \cdot \mu_k), \quad (20)$$

where

$$M_k = A_k - J^\perp(A_k + E_k - D \circ \tau + Y_k/\mu_k), \quad (21)$$

$$N_k = E_k - J^\perp(A_{k+1} + E_k - D \circ \tau + Y_k/\mu_k), \quad (22)$$

$\mu_{\max}$ is an upper bound of μ_k and

$$\rho = \begin{cases} \rho_0, \text{ if } \mu_k \cdot \max(\|A_{k+1} - A_k\|_F, \\ \qquad \|E_{k+1} - E_k\|_F)/\|J^\perp(D \circ \tau)\|_F < \varepsilon_2, \\ 1, \quad \text{otherwise}, \end{cases} \quad (23)$$

[2] Please refer to (20) and (23). For succinctness, we do not repeat it here. Disregarding the adaptive penalty, LADMAP is a special case of the generalized ADM (Deng and Yin 2012) and is closely related to the predictor corrector proximal multiplier method (Chen and Teboulle 1994) which updates variables parallelly rather than alternately.

in which $\rho_0 \geq 1$ is a constant and $\varepsilon_2 > 0$ is a small threshold. Note that in (17)–(18) we have utilized the properties of $J^\perp$ in (16) and $\|J^\perp\| = 1$. The updating scheme (20) and (23) for μ comes from the adaptive penalty strategy in LADMAP (Lin et al. 2011).

The closed form solution to (17) is as follows (Cai et al. 2010):

$$A_{k+1} = S_{\frac{1}{\mu_k}}(M_k), \tag{24}$$

where $S(\cdot)$ is the singular value shrinkage operator:

$$S_\varepsilon(W) = UT_\varepsilon(\Sigma)V^T, \tag{25}$$

in which $U\Sigma V^T$ is the SVD of W and $T_\varepsilon(\cdot)$ is the scalar shrinkage operator defined as (Donoho 1995):

$$T_\varepsilon(x) = \operatorname{sgn}(x)\max(|x| - \varepsilon, 0).$$

Subproblem (18) also has a closed form solution as follows (Yang and Zhang 2011):

$$E_{k+1} = T_{\frac{\lambda}{\mu_k}}(N_k). \tag{26}$$

The iterations (17)–(20) stop when the following two conditions are satisfied:

$$\|J^\perp(A_{k+1} + E_{k+1} - D \circ \tau)\|_F / \|J^\perp(D \circ \tau)\|_F < \varepsilon_1 \tag{27}$$

and

$$\mu_k \cdot \max(\|A_{k+1} - A_k\|_F, \|E_{k+1} - E_k\|_F)/\|J^\perp(D \circ \tau)\|_F < \varepsilon_2. \tag{28}$$

These two stopping criteria are set based on the KKT conditions of problem (15). Details of the deduction can be found in Lin et al. (2011).

3.3 Warm Starting Variables in the Inner Loop

Since the number of iterations in the inner loop greatly affects the efficiency of the whole algorithm, we should provide good initial values to the variables in the inner loop so that the number of iterations can be reduced.

The original algorithm initializes A as the input $D \circ \tau$ and E and Y as 0. Such a cold start strategy does not utilize any information from the previous loop. We observe that solutions from the previous inner loop are good initializations for next inner loop, because the difference in $D \circ \tau$ in successive inner loops becomes smaller and smaller. So their solutions should be close to each other. This motivates us to use the following warm start strategy for the variables:

$$A_0^{i+1} = A_\infty^i, \quad E_0^{i+1} = E_\infty^i, \quad \text{and } Y_0^{i+1} = Y_\infty^i, \tag{29}$$

where the subscripts and superscripts are indices of the inner and outer loops, respectively. The subscripts 0 and ∞ stand for the initial and final values in an inner loop, respectively.

We summarize our LADMAP with variable warm start (LADMAP+VWS) approach for solving TILT in Algorithm 1. Note that we only change the part of the inner loop. The rest of the algorithm is inherited from that in Zhang et al. (2012b).

Algorithm 1 (TILT via LADMAP+VWS)

Input: A rectangular window $D \in \mathbb{R}^{m\times n}$ in an image, the initial parameters τ^0 of the transform, weight parameter $\lambda > 0$, $\rho_0 \geq 1$ and $\mu_{\max} > 0$.
Initialize: $A^0 = D \circ \tau^0$, $E^0 = 0$, $Y^0 = 0$, and $i = 0$.
While not converged **Do**

Step 1: normalize the image and compute the Jacobian w.r.t. the transform:

$$D \circ \tau^i \leftarrow \frac{D \circ \tau^i}{\|D \circ \tau^i\|_F}, \quad J^{i+1} \leftarrow \frac{\partial}{\partial \zeta}\left(\frac{D \circ \zeta}{\|D \circ \zeta\|_F}\right)\Big|_{\zeta=\tau^i};$$

Step 2: solve the linearized convex optimization:

$$\min_{A,E} \|A\|_* + \lambda\|E\|_1, \quad \text{s.t.} \quad (J^{i+1})^\perp(D\circ\tau^i) = (J^{i+1})^\perp(A+E),$$

with the initial conditions: $A_0^{i+1} = A_\infty^i$, $E_0^{i+1} = E_\infty^i$ and $Y_0^{i+1} = Y_\infty^i$, $\mu_0 > 0$, $k = 0$:

While (27) or (28) are not satisfied **Do**

$$A_{k+1}^{i+1} = S_{\frac{1}{\mu_k}}\left(A_k^{i+1} - (J^{i+1})^\perp(A_k^{i+1} + E_k^{i+1} - D \circ \tau^i + \mu_k^{-1}Y_k^{i+1})\right),$$
$$E_{k+1}^{i+1} = T_{\frac{\lambda}{\mu_k}}\left(E_k^{i+1} - (J^{i+1})^\perp(A_{k+1}^{i+1} + E_k^{i+1} - D \circ \tau + \mu_k^{-1}Y_k^{i+1})\right),$$
$$Y_{k+1}^{i+1} = Y_k^{i+1} - \mu_k(J^{i+1})^\perp(A_{k+1}^{i+1} + E_{k+1}^{i+1} - D \circ \tau^i),$$
$$\mu_{k+1} = \min(\mu_{\max}, \rho \cdot \mu_k), \text{ where } \rho \text{ is given by (23)},$$
$$k \leftarrow k + 1.$$

End While

Step 3: compute optimal $\Delta\tau^*$ using Eq. (13), where $A^* = A_\infty^{i+1}$, $E^* = E_\infty^{i+1}$;

Step 4: update transform: $\tau^{i+1} = \tau^i + \Delta\tau^*$, $i \leftarrow i+1$;

End While
Output: converged solution (A^*, E^*, τ^*).

4 Implementation Details

In the previous section we introduce the basic ideas of LADMAP with variable warm start (LADMAP+VWS) algorithm for solving the inner loop of TILT (15). However, there are still some details that need to be handled carefully so that the computation can be the most efficient. In this section, we first show how to compute with $J^\perp$ efficiently, then we discuss how to modify the algorithm in order to accommodate some additional constraints when putting TILT to real applications.

4.1 Multiplying with $J^\perp$

In Algorithm 1, J is actually an order-3 tensor (Zhang et al. 2012b). When computing J is actually arranged into an $mn \times p$ matrix, where $m \times n$ is the size of $D \circ \tau$ and p

is the number of variables to parameterize the transform τ. When multiplying $J^{\perp}$ with a matrix M we actually mean to rearrange the matrix M into an $mn \times 1$ vector and then multiply it with $J^{\perp}$. However, $J^{\perp}$ is an $mn \times mn$ matrix, which is huge and often cannot be fit into the memory. Moreover, the computation cost will be $O((mn)^2)$, which is also unaffordable. So we cannot form $J^{\perp}$ explicitly and then multiply it with a vectorized matrix. Recall that $J^{\perp} = I - J(J^T J)^{-1} J^T$, we may overcome this difficulty by multiplying successively:

$$J^{\perp} v = v - J \cdot ((J^T J)^{-1} \cdot (J^T v)), \tag{30}$$

whose computation cost is only $O(pmn)$. Note that $(J^T J)^{-1}$ is only a $p \times p$ small matrix which can be pre-computed and saved. So the above strategy is much more efficient in both memory and computation.

4.2 Initializing Y

In Algorithm 1, we have to multiply $(J^{i+1})^{\perp}$ with three different matrices. If we initialize Y_0^{i+1} in the subspace spanned by $(J^{i+1})^{\perp}$, then Y_k^{i+1} is always in the subspace spanned by $(J^{i+1})^{\perp}$ during the iteration. Then thanks to the idempotency of $(J^{i+1})^{\perp}$, we have $(J^{i+1})^{\perp} Y_k^{i+1} = Y_k^{i+1}$ and hence A and E can be updated as

$$\begin{aligned} A_{k+1}^{i+1} &= S_{\frac{1}{\mu_k}}\Big(A_k^{i+1} - (J^{i+1})^{\perp}(A_k^{i+1} + E_k^{i+1} - D \circ \tau^i) \\ &\quad - \mu_k^{-1} Y_k^{i+1}\Big), \\ E_{k+1}^{i+1} &= T_{\frac{\lambda}{\mu_k}}\Big(E_k^{i+1} - (J^{i+1})^{\perp}(A_{k+1}^{i+1} + E_k^{i+1} - D \circ \tau^i) \\ &\quad - \mu_k^{-1} Y_k^{i+1}\Big). \end{aligned}$$

Then one can see that we only have to multiply $(J^{i+1})^{\perp}$ with two different matrices, $A_k^{i+1} + E_k^{i+1} - D \circ \tau^i$ and $A_{k+1}^{i+1} + E_k^{i+1} - D \circ \tau^i$, because $(J^{i+1})^{\perp}(A_{k+1}^{i+1} + E_{k+1}^{i+1} - D \circ \tau^i)$ is used for updating both Y_{k+1}^{i+1} and A_{k+2}^{i+1}. This saves one multiplication of multiplying with $(J^{i+1})^{\perp}$ for each iteration.

To combine with the warm start technique, Y should be initialized as

$$Y_0^{i+1} = (J^{i+1})^{\perp} Y_{\infty}^i. \tag{31}$$

Because the dual problem of the inner loop (15) is

$$\max_Y \langle J^{\perp} Y, D \circ \tau \rangle, \quad s.t.\ \|J^{\perp} Y\|_2 \le 1, \|J^{\perp} Y\|_{\infty} \le \lambda^{-1},$$

where $\|\cdot\|_{\infty}$ is the maximum absolute value in a matrix, we see that the optimal Y can be chosen in $\text{span}(J^{\perp})$. So constraining Y in $\text{span}(J^{\perp})$ does not affect the convergence of LADMAP+VWS to the optimal solution. As $Y_{\infty}^i \in \text{span}((J^i)^{\perp})$, when J^{i+1} is close to J^i, $Y_0^{i+1} \approx Y_{\infty}^i$. So (31) makes good combination of warm starting Y and making $Y \in \text{span}(J^{\perp})$.

4.3 Handling Additional Constraints on τ

As is discussed in Zhang et al. (2012b), additional constraints should be imposed on τ so as to eliminate degenerate or trivial solutions, e.g., τ^* being 0. Typical constrains include that both the center and area of the rectangle being fixed. These additional constraints can be formulated as l linear constraints on $\Delta\tau$ (Zhang et al. 2012b):

$$Q \cdot \Delta\tau = 0, \tag{32}$$

where $Q \in \mathbb{R}^{l \times p}$.

Following the same idea as that in Sect. 3.2, we aim at eliminating $\Delta\tau$ with the l additional constraints. As $\Delta\tau$ needs to satisfy both the linear constraints in Eq. (32) and that in problem (3), the overall constraints for $\Delta\tau$ are

$$\begin{bmatrix} J \\ Q \end{bmatrix} \Delta\tau = \begin{bmatrix} A + E - D \circ \tau \\ 0 \end{bmatrix}. \tag{33}$$

Similar to the deductions in Sect. 3.2, we can have an equivalent problem:

$$\min_{A,E} \|A\|_* + \lambda \|E\|_1, \quad s.t.\ WA + WE = W(D \circ \tau), \tag{34}$$

where

$$W = \begin{bmatrix} I - J(J^T J + Q^T Q)^{-1} J^T \\ -Q(J^T J + Q^T Q)^{-1} J^T \end{bmatrix}.$$

Matrix W also enjoys a nice property similar to $J^{\perp}$:

$$W^T W = I - J(J^T J + Q^T Q)^{-1} J^T, \tag{35}$$

which can be utilized to reduce the computational cost. Moreover, $\|W\|_2 = 1$. Then with some algebra LADMAP applied to the above problem goes as follows:

$$\begin{aligned} A_{k+1} &= S_{\frac{1}{\mu_k}}(M_k), \\ E_{k+1} &= T_{\frac{\lambda}{\mu_k}}(N_k), \\ Y_{k+1} &= Y_k + \mu_k \cdot W(A_{k+1} + E_{k+1} - D \circ \tau), \\ \mu_{k+1} &= \min\big(\rho \cdot \mu_k, \mu_{\max}\big), \end{aligned}$$

where

$$\begin{aligned} M_k &= A_k - \mu_k^{-1} W^T Y_k - (W^T W)(A_k + E_k - D \circ \tau), \\ N_k &= A_{k+1} - \mu_k^{-1} W^T Y_k - (W^T W)(A_{k+1} + E_k - D \circ \tau), \end{aligned}$$

and ρ is still computed as (23).

Thanks to (35), the multiplication of $W^T W$ with a vectorized matrix can be done similarly as (30). To further reduce the computational cost, we introduce $\tilde{Y}_k = W^T Y_k$ and initialize it in $\text{span}(W^T W)$. Then M_k, N_k and $\tilde{Y}_k$ are computed as follows:

$$\begin{aligned} M_k &= A_k - \mu_k^{-1} \tilde{Y}_k - (W^T W)(A_k + E_k - D \circ \tau), \\ N_k &= A_{k+1} - \mu_k^{-1} \tilde{Y}_k - (W^T W)(A_{k+1} + E_k - D \circ \tau), \\ \tilde{Y}_{k+1} &= \tilde{Y}_k + \mu_k \cdot (W^T W)(A_{k+1} + E_{k+1} - D \circ \tau). \end{aligned} \tag{36}$$

Again, $(W^T W)(A_{k+1} + E_{k+1} - D \circ \tau)$ is used to update both M_{k+1} and $\tilde{Y}_{k+1}$. In this way, the iterations for the inner loop can be computed very efficiently.

Now the warm start of Y is replaced by that of $\tilde{Y}$, which is:

$$\tilde{Y}_0^{i+1} = \left[(W^{i+1})^T W^{i+1}\right] \tilde{Y}_\infty^i. \tag{37}$$

As $l \ll mn$, $(W^{i+1})^T W^{i+1}$ is actually very close to $(J^{i+1})^\perp$. So (37) is both a good combination of warm starting $\tilde{Y}$ and making $\tilde{Y} \in \text{span}(W^T W)$.

5 Warm Starting for SVD in the Inner Loop

As shown in (24)–(25), to update A we have to compute the SVD of M_k. Unlike other low-rank recovery problems (Cai et al. 2010; Lin et al. 2009; Toh and Yun 2009), partial SVD cannot be used here. This is because partial SVD is faster than full SVD only when the rank of A_{k+1} is less than $\min(m, n)/5$ (Lin et al. 2009), while when rectifying general textures this condition is often violated. So computing the full SVD of M_k is very costly as its complexity is $O(mn \min(m, n))$. Without exaggeration, the efficiency of computing the full SVD dominates the computing time of solving TILT. So we have to reduce the computation cost on full SVD as well.

We observe that, except for the first several steps in the inner loop, the change in M_k between two iterations is relatively small. So we may expect that the SVDs of M_k and M_{k-1} in two successive iterations may be close to each other. This naturally inspires us to utilize the SVD of M_{k-1} to estimate the SVD of M_k.

To do so, we first formulate the SVD problem as follows:

$$\begin{aligned} &(U^*, \Sigma^*, V^*) = \text{argmin}_{U,\Sigma,V}\, F(U, \Sigma, V), \\ &s.t.\ U^T U = I,\ \ \Sigma \text{ is diagonal, and } V^T V = I, \end{aligned} \tag{38}$$

where $F(U, \Sigma, V) = \frac{1}{2}\|M - U\Sigma V^T\|_F^2$ and $M \in \mathbb{R}^{m\times n}$ is the matrix to be decomposed. Without loss of generality we may assume $m \geq n$. $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ are columnly orthonormal and orthogonal matrices, respectively. As we can negate the columns of U or V, we need not require the diagonal entries of Σ to be nonnegative.

5.1 Optimization with Orthogonality Constraints

The usual method to solve a general orthogonality constrained optimization problem is to search along the geodesic of the Stiefel manifold [3] along the direction of the gradient of the objective function projected onto the tangent plane of the manifold (Edelman et al. (1999)). This may require SVDs in order to generate feasible points on the geodesic. Fortunately, Wen and Yin (2013) recently developed a technique that does not rely on SVDs, making our warm start for SVD possible.

Denote the unknown variable as X. Suppose the gradient of the objective function at X is G, then the projection of G onto the tangent plane of the Stiefel manifold at X is $P = GX^T - XG^T$ (Edelman et al. 1999; Wen and Yin 2013). Instead of parameterizing the geodesic of the Stiefel manifold along direction P using the exponential function, Wen and Yin (2013) proposed generating feasible points by the Cayley transform:

$$X(\tau) = C(\tau)X, \quad \text{where } C(\tau) = \left(I + \frac{\tau}{2}P\right)^{-1}\left(I - \frac{\tau}{2}P\right).$$

It can be verified that $X(\tau)$ has the following properties:

1. $X(\tau)$ is smooth in τ;
2. $\left(X(\tau)\right)^T X(\tau) = I, \forall \tau \in \mathbb{R}$, given $X^T X = I$;
3. $X(0) = X$;
4. $\frac{d}{d\tau} X(0) = -P$.

So when $\tau > 0$ is sufficiently small, $X(\tau)$ can result in a smaller objective function value than X.

$X(\tau)$ could be viewed as reparameterizing the geodesic with τ, which does not exactly equal to the geodesic distance. However, when τ is small it is very close to the geodesic distance as it is the length of the two segments enclosing the geodesic (Wen and Yin (2013)). When computing $X(\tau)$, no SVD is required. A matrix inversion and some matrix multiplications are required instead, which is of much lower cost than SVD. However, as both matrix multiplication/inversion and SVD are of the same order of complexity, we have to control the number of matrix multiplications and inversions, so that our warm start based method can be faster than directly computing the full SVD.

5.2 SVD with Warm Start

Now for our SVD problem (38), we can compute the gradient (G_U, G_Σ, G_V) of the objective function $F(U, \Sigma, V)$ with respect to (U, Σ, V) and search on a geodesic on the constraint manifold $C_U \times C_\Sigma \times C_V$ in the gradient direction for the next best solution, where C_U is the Stiefel manifold of all columnly orthonormal $m \times n$ matrices, C_Σ is the subspace of all $n \times n$ diagonal matrices, and C_V is the Stiefel manifold of all $n \times n$ orthogonal matrices.

[3] A Stiefel manifold is a set of columnly orthonormal matrices whose geodesic distance is induced from the Euclidian distance of its ambient space.

We search on the constraint manifold on the following curve:

$$\begin{aligned} U(t) &= \left(I + \frac{t}{2}P_U\right)^{-1}\left(I - \frac{t}{2}P_U\right)U, \\ \Sigma(t) &= \Sigma - t \cdot P_\Sigma, \\ V(t) &= \left(I + \frac{t}{2}P_V\right)^{-1}\left(I - \frac{t}{2}P_V\right)V, \end{aligned} \tag{39}$$

where $P_U = G_U U^T - U G_U^T$ and $P_V = G_V V^T - V G_V^T$ are the projection of G_U and G_V onto the tangent planes of C_U and C_V at U and V, respectively, and $P_\Sigma = \mathrm{diag}(G_\Sigma)$ is the projection of G_Σ onto C_Σ. The details of computing the gradients can be found in Appendix.

Then we may find the optimal t^* such that

$$t^* = \underset{t}{\mathrm{argmin}}\, f(t) = \frac{1}{2}\|M - U(t) \cdot \Sigma(t) \cdot V(t)^T\|_F^2. \tag{40}$$

As $f(t)$ is a complicated function of t, the optimal t^* has to be found by iteration. This will be costly as many matrix inversions and multiplication will be required. So we choose to approximate $f(t)$ by a quadratic function via Taylor expansion:

$$f(t) \approx f(0) + f'(0) \cdot t + \frac{1}{2}f''(0) \cdot t^2, \tag{41}$$

where $f'(0)$ and $f''(0)$ are the first and second order derivatives of $f(t)$ evaluated at 0, respectively. These two derivatives can be computed efficiently. Details of the deductions can be found in Appendix. Then we can obtain an approximated optimal solution $\tilde{t}^* = -f'(0)/f''(0)$ and approximate the SVD of M as $U(\tilde{t}^*)\Sigma(\tilde{t}^*)V(\tilde{t}^*)^T$.

The warm start SVD method is summarized in Algorithm 2. It is called only when the difference between M_k and M_{k-1} is smaller than a pre-defined threshold ε_{svd}.

Algorithm 2 (SVD with Warm Start)

Input: The decomposed matrices U_{k-1}, Σ_{k-1} and V_{k-1} of the SVD of M_{k-1}, and a matrix M_k.

If $\|M_k - M_{k-1}\| < \varepsilon_{svd}$ (**Else** do full svd)

Step 0: $M = M_k$, $U(0) = U_{k-1}$, $\Sigma(0) = \Sigma_{k-1}$, and $V(0) = V_{k-1}$.

Step 1: compute the projected gradients P_U, P_Σ, and P_V of F at $U(0)$, $\Sigma(0)$, and $V(0)$ using (42) to (47), respectively.

Step 2: compute $f'(0)$, $f''(0)$, and $\tilde{t}^* = -f'(0)/f''(0)$ using (48) to (49).

Step 3: compute $U_k = U(\tilde{t}^*)$, $\Sigma_k = \Sigma(\tilde{t}^*)$, $V_k = V(\tilde{t}^*)$ using (39).

Output: U_k, Σ_k, and V_k as the decomposed matrices in the SVD of M_k.

Although $U(\tilde{t}^*)\Sigma(\tilde{t}^*)V(\tilde{t}^*)^T$ is an approximate SVD of M, our final goal is to compute the singular value shrinkage of M_k in order to update A_{k+1} (see (17), (24) and (25)), not the SVD of M_k itself. We can show that when M_k is close enough to M_{k-1}, computing the SVD of M_k approximately still produces a highly accurate A_{k+1}. The corner stone of our proof is the following pseudocontraction property of the singular value shrinkage operator:

$$\begin{aligned} \|S_\varepsilon(W_1) - S_\varepsilon(W_2)\|_F^2 &\le \|W_1 - W_2\|_F^2 \\ &\quad - \|[S_\varepsilon(W_1) - W_1] - [S_\varepsilon(W_2) - W_2]\|_F^2, \end{aligned}$$

thanks to Lemma 3.3 of Pierra (1984) and the fact that $S_\varepsilon(\cdot)$ is the proximal mapping of the nuclear norm (Cai et al. 2010). Then we have:

$$\begin{aligned} &\|S_\varepsilon\big(U(\tilde{t}^*)\Sigma(\tilde{t}^*)V(\tilde{t}^*)^T\big) - S_\varepsilon\big(M_k\big)\|_F \\ &\quad \le \|U(\tilde{t}^*)\Sigma(\tilde{t}^*)V(\tilde{t}^*)^T - M_k\|_F \\ &\quad \le \|U(0)\Sigma(0)V(0)^T - M_k\|_F = \|M_{k-1} - M_k\|_F. \end{aligned}$$

Since we switch to our warm start technique for SVD in the inner loop only when M_k is very close to M_{k-1} (i.e., $\|M_{k-1} - M_k\| < \varepsilon_{svd}$), it is guaranteed that:

$$\|S_\varepsilon\big(U(\tilde{t}^*)\Sigma(\tilde{t}^*)V(\tilde{t}^*)^T\big) - S_\varepsilon\big(M_k\big)\|_F < \varepsilon_{svd}.$$

Hence our approximate SVD for M_k still results in a highly accurate A_{k+1}.

6 Experiments

In this section, we conduct several experiments on both the inner loop only and the complete TILT problem to evaluate the performance of our proposed LADMAP with warm starts method. Numerical experiments on the inner loop only are conducted by using synthetic data in order to demonstrate the effectiveness of LADMAP and the warm start for SVD. For the complete TILT problem, we conduct experiments on images with simulated and real deformations to further test the efficiency and robustness of LADMAP and the warm start techniques. The images we use are from a low-rank textures data set and a natural images data set, respectively.

The code for the original ADM based method is provided by the first author of Zhang et al. (2012b). For fair comparison, we set the common parameters in all compared methods the same. All the codes are in MATLAB and the experiments are run on a workstation with an Intel Xeon E5540@2.53GHz CPU and 48 GB memory.

6.1 Numerical Study on the Inner Loop Only

In this section, we use synthetic data to compare the efficiency of the original ADM based algorithm, our LADMAP based algorithm, and LADMAP with warm start for computing SVD (LADMAP+SVDWS) on the inner loop only. As this time we only focus on the inner loop, the effectiveness of variable warm start, which requires outer loops, cannot be shown. We generate $D \circ \tau$ and Jacobian J randomly and use them as the input for ADM, LADMAP, and LADMAP+SVDWS. For fair comparison, we tune the extra

Table 1 Comparison of the efficiency of algorithms on the inner loop of TILT only

Size of $D \circ \tau$	ADM			LADMAP			LADMAP+SVDWS		
	Time (s)	#Iteration	Objective function	Time (s)	#Iteration	Objective function	Time (s)	#Iteration	Objective function
10×10	1.98E-05	49.2	6.4836	**7.49E-06**	20.4	6.4814	8.76E-06	20.9	6.4815
50×50	0.00112	51.5	76.6508	**0.00054**	21.2	76.6468	0.00056	22.2	76.6585
100×100	0.00538	51.1	217.001	0.00252	22.2	216.985	**0.00225**	22.9	216.987
300×300	0.1859	50	1123.66	0.10501	21.9	1123.67	**0.08792**	22.9	1123.70
500×500	1.3802	50	2420.41	0.6574	22	2419.63	**0.5343**	21	2420.28
1000×1000	53.0936	50	6844.82	25.8139	23	6844.14	**18.5953**	23	6845.32

The time costs, numbers of iterations, and optimal objective function values are averaged over ten trials on random data

parameters ε_2 and ρ_0 in LADMAP and LADMAP+SVDWS and ε_{svd} in LADMAP+SVDWS so that the three methods stop with almost the same objective function values. All methods are initialized as $A_0^0 = D \circ \tau^0$ and $E_0^0 = 0$. $\Delta\tau_0$ in the ADM method is initialized as $\Delta\tau_0 = 0$. Under these conditions, we compare the three methods on their computation time, the number of iterations needed to converge, and the objective function value when the iterations stop. The comparison is done under different sizes of $D \circ \tau$. All the tests are repeated for ten times and the average quantities are reported.

The comparative results are shown in Table 1. We can observe that all the three methods arrive at roughly the same objective function values. However, LADMAP uses less than half of the number of iterations than ADM does and the SVD warm start only changes the number of iterations very slightly. Consequently, LADMAP is much faster than ADM, while SVDWS further speeds up the computation of LADMAP when the size of $D \circ \tau$ is not too small (e.g., ≥ 100). The acceleration rate also increases when the size of $D \circ \tau$ grows[4]. When the size of $D \circ \tau$ is very small (e.g., < 100), SVDWS does not seems to speed up the computation. This may due to the ultra-short computing time and hence other processes on the workstation supporting the computing environment can influence the total computing time. Anyway, the slowdown is rather insignificant. As the speed of solving large sized TILT is more time demanding in real applications, adopting SVDWS is still advantageous.

6.2 Comparisons on the Complete TILT Problem

In this subsection, using real image data we compare the original ADM method, LADMAP, LADMAP with variable warm start (LADMAP+VWS), and LADMAP with both variable warm start and SVD warm start (LADMAP+VWS+SVDWS) on solving the whole TILT problem, in order to show the effectiveness of LADMAP and the two warm start techniques. As we have pointed out before that the original ADM may not converge to the optimal solution of the inner loop, we first compare the convergence performance of ADM and LADMAP. Then we test the robustness of ADM and LADMAP when there are corruptions. We also present some examples on which ADM fails but our LADMAP works well. Finally, we compare the computation time of various methods on both synthetically and naturally deformed images. The synthetically deformed images are generated by deforming the low-rank textures with predefined transforms. The naturally deformed images are from our images data set which contains over 100 images downloaded from the web.

6.2.1 Range of Convergence

Since LADMAP for the inner loop is proven to converge to the optimal solution, we expect that it will outperform ADM in recovering the correct deformation when solving the whole TILT problem[5]. To show that LADMAP can recover broader range of deformation than ADM does, we test them with a standard checker-board pattern.

Following the same setting in Zhang et al. (2012b), we deform a checker-board pattern by an affine transform: $y = Ax + b$, where $x, y \in \mathbb{R}^2$, and test if the two algorithms can recover the correct transform. The matrix A is parameterized as $A(\theta, t) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$, where θ is the rotation angle and t is the skew value. We change θ within the range $[0, \pi/6]$ with a step size $\pi/60$, and $t \in [0, 1]$ with a step size 0.05. We can observe from Fig. 2 that the range of convergence of our LADMAP completely encloses that of ADM. So the working range of LADMAP is larger than that of ADM.

We further test with real images taken from natural scenes and manually prescribe the regions to be rectified. We have

[4] We will see much higher acceleration rates when using SVDWS on real data (see Sect. 6.2.3).

[5] Note that as the whole TILT problem (2) is *not* a convex program, LADMAP cannot achieve the global optimum either. So LADMAP can also fail to recover the correct deformation.

Table 2 Comparison of speed on images with artificial affine deformations

	Size	ADM		LADMAP		LADMAP+VWS		LADMAP+VWS+SVDWS	
		Time (s)	Relative errors	Time (s)	Relative errors	Time (s)	Relative errors	Time (s)	Relative errors
$\theta = 30°$									
1	50×43	6.6318	0.1563	2.2941	0.1561	1.1857	0.1561	0.4034	0.1560
2	31×36	0.2503	0.0086	0.1834	0.0087	0.0913	0.0085	0.0063	0.0085
3	38×37	0.1594	0.00087	0.1057	0.00087	0.0459	0.00088	0.0338	0.00088
4	28×26	0.1091	0.0040	0.0634	0.0040	0.0408	0.0039	0.0353	0.0039
5	38×45	3.139	0.4226	1.977	0.4219	0.420	0.4193	0.199	0.4191
$t = 0.2$									
1	27×27	0.8926	0.0252	0.3068	0.0245	0.1605	0.0243	0.0995	0.0247
2	47×50	3.2381	0.2066	2.1583	0.2082	0.4551	0.1944	0.1988	0.1949
3	47×45	0.6271	0.0065	0.5043	0.0065	0.3914	0.0067	0.3124	0.0066
4	30×26	2.4916	0.1896	1.3644	0.1881	0.9039	0.1859	0.6126	0.1849
5	26×28	0.3681	0.0538	0.2254	0.0536	0.1622	0.0536	0.1274	0.0539

The time costs are counted in seconds. The relative errors are between the computed τ and the ground truth

Table 3 Comparison of computation speed on real images

Case	Size	ADM	LADMAP		LADMAP+VWS		LADMAP+VWS+SVDWS	
		Time (s)	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	82×82	6.2728	4.8299	1.29	1.8119	3.46	1.1386	5.50
2	84×78	7.6238	2.8338	2.69	1.6067	4.75	1.3056	5.84
3	80×87	5.4040	2.4863	2.17	1.3238	4.08	0.9587	5.64
4	68×72	4.7575	2.8580	1.66	0.9953	4.78	0.8237	5.78
5	51×60	3.1128	2.7048	1.15	0.8821	3.53	0.4906	6.35
6	89×90	3.3004	2.2573	1.46	0.8274	3.99	0.5174	6.38
7	96×107	6.7319	2.2305	3.02	1.6650	4.04	0.8212	8.20
8	65×60	2.5367	1.4115	1.79	0.5022	5.05	0.4172	6.08
9	72×71	2.5501	0.9307	2.73	0.6234	4.09	0.3967	6.43
10	83×90	4.3284	3.6923	1.17	1.1078	3.91	0.7141	6.06

The time costs are counted in second. The speedup rates are w.r.t. the baseline ADM method

found many examples that LADMAP works better than ADM. However, we have not encountered any example that ADM works better. Part of the examples are shown in Fig. 3. They are chosen according to the challenging cases listed in Zhang et al. (2012b) for TILT to rectify. For example, the first example is lacking regularity in the printed texts or the prescribing rectangular contains too much background; the second example has very large deformation; and the third example may has a boundary effect. On these examples, LADMAP works much better than ADM in rectifying the prescribed regions.

6.2.2 Robustness under Corruption

In this subsection, we compare the robustness of ADM and LADMAP when there is corruption in images. We test with some low-rank textures shown in Fig. 4a. Following the same setting in Zhang et al. (2012b), we introduce a small deformation, say rotation by 10°, and examine whether the two methods can recover the correct transform under different levels of random corruption. We randomly select a fraction (from 0 to 100 %) of the pixels and assign all their RGB values as random values uniformly distributed in (0, 255). We run the two methods on such corrupted images and record at each level of corruption how many images are correctly rectified.

The comparative results are shown in Fig. 4b. We can see that when the percentage of corruption is larger than 15 %, our LADMAP always outperforms ADM. For example, when 50 % of the pixels are corrupted, LADMAP can still obtain the correct solution on more than half of the test images, while ADM can only deal with about 30 % of the images. The

above comparison demonstrates that our LADMAP method is more robust than the original ADM method when there is corruption in images.

6.2.3 Speed of the Algorithms

In this subsection, we report the computational cost of four different algorithms: ADM, LADMAP, LADMAP+VWS, and LADMAP+VWS+ SVDWS for solving the whole TILT problem, in order to show the computational improvement from each component.

The comparisons are done on two types of images. The first type of images are obtained by applying affine transforms to real images. The second one are images of natural scenes that can undergo either affine or projective transform, in which the rectangular regions to be rectified are manually prescribed.

For the first type of images, part of the comparison results, namely the images are rotated with $\theta = 30°$ only or skewed with $t = 0.2$ only, are presented in Table 2. We tuned the four algorithms so that all of them produced nearly the same relative error, which is defined as $\frac{\|\tau^* - \tau_G\|}{\|\tau_G\|}$ where τ^* is the optimal solution produced by the corresponding algorithm and τ_G is the ground-truth deformation matrix. We can see that LADMAP is much faster than ADM, the variable warm start speeds up the convergence, and SVD warm start further cuts the computation cost.

For the second type of images, comparison of the speed on ten of the images is shown in Table 3. We can see that LADMAP can be at least 20 % faster than ADM, with the variable warm start LADMAP can be at least 2.5 times faster than ADM, and with the SVD warm start further added, LADMAP can be at least 4.5 times faster than ADM. So the speed advantage of our new algorithm is apparent.

7 Conclusions

In this paper we propose an efficient and robust algorithm for solving the recently popular TILT problem. We reformulate the inner loop of TILT and apply a recently proposed LADMAP to solve the subproblem of the inner loop. For further speed up, we also propose variable warm start for initialization and introduce an SVD warm start technique to cut the computational cost of computing SVDs. Extensive experiments have testified to the better convergence property, higher robustness to corruption, and faster speed of LADMAP, and the effectiveness of our warm start techniques.

Acknowledgments Z. Lin is supported by the National Natural Science Foundation of China (Grant Nos. 61272341, 61231002, and 61121002).

Appendix A: Details of Computing SVD with Warm Start

In this section, for brevity we simply write $U(0)$, $\Sigma(0)$, and $V(0)$ as U, Σ, and V, respectively.

Gradients

The gradients are:

$$G_U = \frac{\partial F}{\partial U} = U\Sigma^2 - MV\Sigma, \tag{42}$$

$$G_V = \frac{\partial F}{\partial V} = V\Sigma^2 - M^T U\Sigma, \tag{43}$$

$$G_\Sigma = \frac{\partial F}{\partial \Sigma} = \Sigma - U^T MV. \tag{44}$$

The projected gradients are:

$$P_U = G_U U^T - UG_U^T = (USV^T)M^T - M(USV^T)^T, \tag{45}$$

$$P_V = G_V V^T - VG_V^T = (USV^T)^T M - M^T(USV^T), \tag{46}$$

$$P_\Sigma = \mathrm{diag}(\Sigma - U^T MV). \tag{47}$$

Note that USV^T is just the SVD of the matrix in last iteration. So we do not need to recompute their products.

Derivatives of $f(t)$

First-Order Derivative

We utilize the chain rule to calculate the derivative, let

$$H = h(t) = M - U(t) \cdot \Sigma(t) \cdot V(t)^T$$

and $\ell(H) = \frac{1}{2}\|H\|_F^2$, then

$$\begin{aligned} f'(t) = \frac{\mathrm{d}\ell(H)}{\mathrm{d}t} &= \mathrm{tr}\Big[\Big(\frac{\mathrm{d}\ell(H)}{\mathrm{d}H}\Big)^T \cdot \frac{\mathrm{d}h(t)}{\mathrm{d}t}\Big] \\ &= \mathrm{tr}\Big[h(t)^T \cdot h'(t)\Big], \end{aligned}$$

where

$$\begin{aligned} h'(t) = &-U'(t)\Sigma(t)V(t)^T - U(t)\Sigma'(t)V(t)^T \\ &-U(t)\Sigma(t)V'(t)^T, \end{aligned}$$

and

$$\begin{aligned} U'(t) = &-\Big(I + \frac{1}{2}tP_U\Big)^{-1}\frac{1}{2}P_U\Big(I + \frac{1}{2}tP_U\Big)^{-1} \\ &\Big(I - \frac{1}{2}tP_U\Big)U - \Big(I + \frac{1}{2}tP_U\Big)^{-1}\frac{1}{2}P_U U, \\ \Sigma'(t) = &-P_\Sigma, \end{aligned}$$

$$V'(t)^T = V^T \frac{1}{2} P_V \left(I - \frac{1}{2} t P_V\right)^{-1} + V^T \left(I + \frac{1}{2} t P_V\right)\left(I - \frac{1}{2} t P_V\right)^{-1} \frac{1}{2} P_V \left(I - \frac{1}{2} t P_V\right)^{-1}.$$

Second-Order Derivative

Since $\mathrm{d}\big(\mathrm{tr}(Q)\big) = \mathrm{tr}(\mathrm{d}Q)$ and $\mathrm{d}Q^T = (\mathrm{d}Q)^T$, we have

$$f''(t) = \frac{\mathrm{d}f'(t)}{\mathrm{d}t} = \mathrm{tr}\Big[h'(t)^T \cdot h'(t) + h(t)^T \cdot h''(t)\Big],$$

where

$$\begin{aligned} h''(t) &= \frac{\mathrm{d}h'(t)}{\mathrm{d}t} \\ &= -U''(t)\Sigma(t)V(t)^T - U'(t)\Sigma'(t)V(t)^T \\ &\quad -U'(t)\Sigma(t)V'(t)^T - U'(t)\Sigma'(t)V(t)^T \\ &\quad -U(t)\Sigma''(t)V(t)^T - U(t)\Sigma'(t)V'(t)^T \\ &\quad -U'(t)\Sigma(t)V'(t)^T - U(t)\Sigma'(t)V'(t)^T \\ &\quad -U(t)\Sigma(t)V''(t)^T \end{aligned}$$

and

$$\begin{aligned} U''(t) &= \frac{\mathrm{d}U'(t)}{\mathrm{d}t} \\ &= \frac{1}{2}\left(I + \frac{1}{2} t P_U\right)^{-1} P_U \left(I + \frac{1}{2} t P_U\right)^{-1} \\ &\quad P_U \left(I + \frac{1}{2} t P_U\right)^{-1} \left(I - \frac{1}{2} t P_U\right) U \\ &\quad + \frac{1}{2}\left(I + \frac{1}{2} t P_U\right)^{-1} P_U \left(I + \frac{1}{2} t P_U\right)^{-1} P_U U, \\ \Sigma''(t) &= 0, \\ V''(t)^T &= \frac{\mathrm{d}V'(t)}{\mathrm{d}t} \\ &= \frac{1}{2} V^T \left(I + \frac{1}{2} t P_V\right)\left(I - \frac{1}{2} t P_V\right)^{-1} \\ &\quad P_V \left(I - \frac{1}{2} t P_V\right)^{-1} P_V \left(I - \frac{1}{2} t P_V\right)^{-1} \\ &\quad + \frac{1}{2} V^T P_V \left(I - \frac{1}{2} t P_V\right)^{-1} P_V \left(I - \frac{1}{2} t P_V\right)^{-1}. \end{aligned}$$

Taylor Expansion

We want to do the second-order Taylor expansion of $f(t)$ at $t = 0$:

$$f(t) \approx f(0) + f'(0) \cdot t + \frac{1}{2} f''(0) \cdot t^2.$$

First, the function value at 0 is:

$$f(0) = \frac{1}{2}\|M - U\Sigma V^T\|_F^2.$$

However, $f(0)$ need not be computed as we are only interested in $f'(0)$ and $f''(0)$ for obtaining $\tilde{t}^*$.

Second, since $f'(t) = \mathrm{tr}\Big[h(t)^T \cdot h'(t)\Big]$ we have

$$f'(0) = \mathrm{tr}\Big[h(0)^T \cdot h'(0)\Big], \tag{48}$$

where

$$\begin{aligned} h(0) &= M - U(0)\Sigma(0)V(0)^T, \\ h'(0) &= -U'(0)\Sigma(0)V(0)^T - U(0)\Sigma'(0)V(0)^T \\ &\quad -U(0)\Sigma(0)V'(0)^T. \end{aligned}$$

Given $U(0) = U, \Sigma(0) = \Sigma, V(0) = V, U'(0) = -P_U U, \Sigma'(0) = -P_\Sigma, V'(0) = -P_V V$, we have

$$h'(0) = P_U(U\Sigma V^T) + U P_\Sigma V^T - (U\Sigma V^T) P_V.$$

Finally, we come to the second-order derivative:

$$f''(0) = \mathrm{tr}\Big[h'(0)^T \cdot h'(0) + h(0)^T \cdot h''(0)\Big], \tag{49}$$

where

$$\begin{aligned} h''(0) &= -U''(0)\Sigma(0)V(0)^T - U'(0)\Sigma'(0)V(0)^T \\ &\quad -U'(0)\Sigma(0)V'(0)^T - U'(0)\Sigma'(0)V(0)^T \\ &\quad -U(0)\Sigma''(0)V(0)^T - U(0)\Sigma'(0)V'(0)^T \\ &\quad -U'(0)\Sigma(0)V'(0)^T - U(0)\Sigma'(0)V'(0)^T \\ &\quad -U(0)\Sigma(0)V''(0)^T, \end{aligned}$$

with $U''(0) = P_U^2 U$, $\Sigma''(0) = 0$ and $V''(0) = P_V^2 V$.

Thus we have

$$\begin{aligned} h''(0) &= -P_U^2 U\Sigma V^T - 2 \cdot P_U U P_\Sigma V^T + 2 \cdot P_U U\Sigma V^T P_V \\ &\quad +2 \cdot U P_\Sigma V^T P_V - U\Sigma V^T P_V^2 \\ &= -P_U Z + Z P_V, \end{aligned}$$

where $Z = h'(0) + U P_\Sigma V^T$.

References

Bertsekas, D., & Tsitsiklis, J. (1997). *Parallel and distributed computation: Numerical methods*. Cambridge: Athena Scientific.

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

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

Chen, G., & Teboulle, M. (1994). A proximal-based decomposition method for convex minimization problems. *Mathematical Programming*, *64*(1), 81–101.

Deng, W., & Yin, W. (2012). *On the global and linear convergence of the generalized alternating direction method of multipliers*. Rice University Technical Report. Houston: Rice University.

Donoho, D. (1995). De-noising by soft-thresholding. *IEEE Transaction on Information Theory*, *41*(3), 613–627.

Donoho, D. (2004). For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, *59*(6), 797–829.

Edelman, A., Arias, T., & Smith, S. (1999). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, *20*(2), 303–353.

He, B., Tao, M., & Yuan, X. (2012). Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, *22*(2), 313–340.

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

Lin, Z., Liu, R., & Su, Z. (2011). Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in Neural Information Processing System*. Granada: NIPS.

Liu, G., Lin, Z., & Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*. Haifa: ICML.

Mobahi, H., Zhou, Z., Yang, A., & Ma, Y. (2011). Holistic 3D reconstruction of urban structures from low-rank textures. In *IEEE Workshop on 3D Representation and Recognition (3dRR-11)*. Barcellona: IEEE.

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 *IEEE Conference on Computer Vision and Pattern Recognition*. San Francisco: IEEE.

Pierra, G. (1984). Decomposition through formalization in a product space. *Mathematical Programming*, *28*, 96–115.

Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., & Dellaert, F. (2008). Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage: IEEE.

Toh, K., & Yun, S. (2009). An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, *6*, 615–640.

Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, accepted.

Yang, J., & Zhang, Y. (2011). Alternating direction algorithms for l_1 problems in compressive sensing. *SIAM Journal on Scientific Computing*, *33*(1), 250–278.

Zhang, Z., Liang, X., & Ma, Y. (2011a). Unwrapping low-rank textures on generalized cylindrical surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs: IEEE.

Zhang, Z., Matsushita, Y., & Ma, Y. (2011b). Camera calibration with lens distortion from low-rank textures. In *IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs: IEEE.

Zhang, X., Lin, Z., Sun, F., & Ma, Y. (2012a). Rectification of Chinese characters as transform invariant low-rank textures. *Pattern Recognition* (submitted to).

Zhang, Z., Ganesh, A., Liang, X., & Ma, Y. (2012b). TILT: Transform-invariant low-rank textures. *International Journal of Computer Vision (IJCV)*, *99*(1), 1–24.

Zhao, P., & Quan, L. (2011). Translation symmetry detection in a fronto-parallel view. In *IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs: IEEE.