Efficient Implementation of Truncated Reweighting Low-rank Matrix Approximation

Jianwei Zheng, Mengjie Qin, Xiaolong Zhou, *Member, IEEE*, Jiafa Mao, and Hongchuan Yu

***Abstract*—The weighted nuclear norm minimization and truncated nuclear norm minimization are two well-known low-rank constraint for visual applications. By integrating their advantages into a unified formulation, we find a better weighting strategy, namely truncated reweighting norm minimization (TRNM), which provides better approximation to the target rank for some specific task. Albeit nonconvex and truncated, we prove that TRNM is equivalent to certain weighted quadratic programming problems, whose global optimum can be accessed by the newly presented reweighting singular value thresholding operator. More importantly, we design a computationally efficient optimization algorithm, namely momentum update and rank propagation (MURP), for the general TRNM regularized problems. The individual advantages of MURP include: (1) reducing iterations through non-monotonic search, and (2) mitigating computational cost by reducing the size of target matrix. Furthermore, the descent property and convergence of MURP are proven. Finally, two practical models, i.e., MCTRNM and SCTRNM, are presented for visual applications. Extensive experimental results show that our methods achieve better performance, both qualitatively and quantitatively, compared with several state-of-the-art algorithms.**

***Index Terms*—Nuclear norm minimization, Singular value thresholding, Accelerated proximal gradient, Matrix completion, Subspace clustering.**

I. INTRODUCTION

THE low-rank property is prevalent in visual applications due to the fact that there often exists a significant correlation between different units of visual data. This can be the case when each column of a matrix $\boldsymbol{X}$ represents 2D image at certain frame of video sequence, since images at nearby frames are strongly correlated [1]. On that basis, finding a low-rank solution to an optimization problem, e.g., matrix completion (MC) or subspace clustering (SC), has attracted a great deal of attention over the last decade. Concrete applications, where low-rank modeling of $\boldsymbol{X}$ is relevant, can be found in scene reconstruction [2], video inpainting [3], background subtraction [4], or video matting [5], among many others.

This work was supported partially by the National Natural Science Foundation of China (No. 61602413, No. 61876168), partially by the Natural Science Foundation of Zhejiang Province (No. LY19F030016, No. LY18F030020), and partially by the EU H2020 project-AniAge (No. 691215). (Corresponding author: Xiaolong Zhou and Jiafa Mao.)

Jianwei Zheng, Mengjie Qin, and Jiafa Mao are with the School of Computer Science and Engineering at Zhejiang University of Technology, China. Email: zjw@zjut.edu.cn

Xiaolong Zhou is with the College of Electrical and Information Engineering, Quzhou University, China. Email: zxl@zjut.edu.cn

Hongchuan Yu is with the Department of National Centre for Computer Animation at Bournemouth University.

Manuscript received January, 2019; revised April, 2019; accepted May, 2019.

Generally speaking, there are two mainstream approaches to find the low-rank structure of data, i.e., factorization based [6], [7] and regularization based [8], [9], [10], [11] methods. Since the former is restricted to problems with known rank, which suffers from finding the global optimal solution due to their non-convex nature, we focus on the latter, whose cost function for a low-rank matrix $\boldsymbol{X} \in \Re^{m\times n}$ can be generally formulated as follows:

$$\min_{\boldsymbol{X}} F(\boldsymbol{X}) = f(\boldsymbol{X}) + \lambda g(\boldsymbol{X}) \tag{1}$$

where λ is a balance parameter, f is the fidelity term, and g is the regularizer. Typically, f is a smooth and convex function (e.g., square and logistic losses), while g is possibly a nonconvex, nonsmooth and non-Lipschitz function. The main problem of Eq. (1) is to deal with nonconvex constraints during minimization. The current research mainly aims to overcome nonconvex optimization barrier, i.e. effectiveness of algorithm. There is a lack of research on time complexity, i.e. efficiency of algorithm.

One of the most typical low-rank regularizers is the nuclear norm (NN), i.e., $\|\boldsymbol{X}\|_* = \sum_{i=1}^{q} \sigma_i(\boldsymbol{X})$, where $q = \min(m, n)$ and $\sigma_i(\boldsymbol{X})$ denotes the ith largest singular values of $\boldsymbol{X}$. According to [12], NN is the tightest convex surrogate function of the intrinsic rank constraint, and it can recover unknown low-rank matrices from finite observations under broad conditions. However, despite strong theoretical foundation, NN-based minimization (NNM) problem may fail to obtain the optimal solution and suffer slow convergence. The main problem is that the nuclear norm is still a suboptimal relaxation to the rank minimization problem. Therefore, lots of attempts for improvements have been investigated [4], [12], [13], [14], [15], [16], [17], [18]. An intuitive scheme is to enforce low-rank property by using nonconvex constraints for closer approximation, e.g., Schatten p [12], Logarithm [12], [15], and Rational [15], etc. However, most of these functions treat all singular values in equivalence, and hence are not effective enough to cope with many practical applications where different rank components bear different contributions. The other alternative is to involve a weight $\boldsymbol{w}$ into NNM (WNNM) [16], [17], i.e., $\|\boldsymbol{X}\|_{\boldsymbol{w}} = \sum_{i=1}^{q} w_i\sigma_i(\boldsymbol{X})$, for holding certain prior knowledge of given singular values. Although WNNM is more flexible compared with NNM by penalizing larger singular values less than the smaller ones such as to preserve the major data components, it does not fully take into consideration a priori rank information for the encountered problem. To address this limitation, the truncated nuclear norm minimization (TNNM) [18], [19], i.e.,$\|\boldsymbol{X}\|_r = \sum_{i=r+1}^{q} \sigma_i(\boldsymbol{X})$, is proposed for achieving better control of the target rank to be a given

parameter r. However, it is also questionable due to the missing consideration of different rank contributions, which leads to suboptimal soft thresholding.

Although effectiveness has been intensively studied, unfortunately the efficiency has seldom mentioned. Most existing approaches resort to the two well-known first-order algorithms for a low-rank solution, i.e., alternating direction method of multipliers (ADMM) [9] and accelerated proximal gradient (APG) [20]. However, both of these two suffer from slow convergence and require great amount of computation cost at each iteration due to the full SVD operation. Additionally, for the TNNM problem, some extra manipulations are further required due to the discrete nature of truncation step. To avoid the directly minimizing of TNNM, a common way is to alternatively optimize it by a two-step scheme [18]. The first approximates TNNM with a difference of two convex surrogates by introducing some auxiliary variables. The second updates the target matrix by certain off-the-shelf algorithms. While some acceleration schemes, e.g., the partial sum of singular values (PSSV) [21] and the extension via weighted residual error (EWRE) [22], have been attempted to minimize the TNNM problem by a one-step scheme, they are still not efficient enough due to the additional matrix multiplication and slow convergence.

To address the issues of both effectiveness and efficiency, a special regularizer by combining WNNM and TRNM constraint as well as a computationally efficient method is proposed in this paper. Concretely, our main contributions can be highlighted as follows:

1) Inspired by WNNM and TNNM, we present a specific low-rank regularizer, namely truncated reweighting norm minimization (TRNM), as well as a reweighting singular value thresholding (RSVT) operator for a better approximation to the original rank minimization problem.
2) An improved APG algorithm with adaptive momentum update criterion is proposed, which incorporates automatic extrapolation and non-monotone search based on an extended cost function. Moreover, we prove that its sufficient descent property can be guaranteed by the well-defined search criterion, and the convergence is also promised.
3) A ranking propagation RSVT scheme is proposed to avoid the full SVD computation. By optimistically predicting the progressive subspace rank with an initial guess of the main matrix action, the RSVT operation can be efficiently approximated from some smaller matrix.
4) We apply the proposed TRNM and the improved APG method to matrix completion and subspace clustering. Extensive experiments on image inpainting, video scene segmentation and gesture segmentation demonstrate that our methods achieve state-of-the-art performance both on effectiveness and efficiency.

The remainder of this paper is organized as follows. In Section II, we describe the TRNM regularizer and present the RSVT operator to analytically solve it. In Section III, the detailed description of momentum update and rank propagation (MURP) is given. TRNM and MURP are applied to matrix completion and subspace clustering in Section IV. The experimental results are presented in Section V, and Section VI concludes the paper.

II. Truncated Reweighting Norm Minimization

In this section, we first present the TRNM constrained low-rank matrix approximation problem, and then provide our RSVT operator under the APG framework for its optimal and closed-form solution.

A. Problem Formulation

The concerned truncated reweighting norm of matrix $\boldsymbol{X} \in \Re^{m\times n}$ is defined as

$$\|\boldsymbol{X}\|_{\boldsymbol{w},r} = \sum_{i=r+1}^{q} w_i \sigma_i(\boldsymbol{X}) \tag{2}$$

where r is the predicted rank, e.g., r=1 for background subtraction and r=3 for photometric stereo; $q = min(m, n)$, and $\boldsymbol{w} = [w_1, \ldots, w_q]$ is a non-negative and non-descending vector for penalizing different rank components. With TRNM, Eq. (1) can be rewritten as

$$\min_{\boldsymbol{X}} F(\boldsymbol{X}) = f(\boldsymbol{X}) + \lambda \|\boldsymbol{X}\|_{\boldsymbol{w},r} \tag{3}$$

Obviously, the convexity property of NNM cannot be preserved in (3) due to involving of truncation operation and weight constraint. In the following subsection, we assume that f in (3) is a $L-$Lipschitz smooth function, i.e., $\|\nabla f(\boldsymbol{X}_1) - \nabla f(\boldsymbol{X}_2)\|_2 \leq L\|\boldsymbol{X}_1 - \boldsymbol{X}_2\|_2$, and present the core updating scheme under the framework of APG.

B. General Solution with APG

ADMM and APG are two most popular first-order approaches for solving problem (3). ADMM separates the objective function via additionally introduced variables, which may results in tedious parameter setting and slow convergence. It has been proved that APG converges to a solution with the primal residual being smaller than ε after $1/\varepsilon^{0.5}$ iterations [24]. We adopt APG for (3), which iteratively update $\boldsymbol{X}$ as

$$\boldsymbol{Y}^k = \boldsymbol{X}^k + \beta_k(\boldsymbol{X}^k - \boldsymbol{X}^{k-1}) \tag{4}$$

$$\boldsymbol{X}^{k+1} = \arg\min_{\boldsymbol{X}} \{\frac{1}{2}||\boldsymbol{X} - \boldsymbol{G}||_F^2 + \frac{\lambda}{\mu}||\boldsymbol{X}||_{\boldsymbol{w},r}\} \tag{5}$$

where $\boldsymbol{G} = \boldsymbol{Y}^k - \frac{1}{\mu}\nabla f(\boldsymbol{Y}^k)$, β_k is the extrapolation parameter for faster convergence, and $\mu > 0$ is a step-size satisfying certain conditions on the Lipschitz constant L.

To minimize Eq. (5), we present the RSVT operator $\Xi_{r,\tau}(\cdot)$. First, from the von Neumann theorem, we know that for any matrices $\boldsymbol{A}, \boldsymbol{B} \in \Re^{m\times n}$, $\mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B})$ is upper bounded by the inner product of $\sigma(\boldsymbol{A})$ and $\sigma(\boldsymbol{B})$. Note that the maximum value of $\mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B})$ can be reached only when $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same singular vector matrices. This fact is crucial to deduce the RSVT operator, which is described in Theorem 1.

Theorem 1 (RSVT). Given scalar $\tau = \lambda/\mu$ and matrix $\boldsymbol{G} \in \Re^{m\times n}$, without loss of generality, we assume $m \geq n$,

and let $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{V}^{\boldsymbol{T}}$ be the SVD of $\boldsymbol{G}$. Then the closed-form solution of (5) can be achieved by the RSVT operator as follows.

$$\Xi_{r,\tau}(\boldsymbol{G}) = \boldsymbol{U}\boldsymbol{S}_{\tilde{\boldsymbol{w}}}(\boldsymbol{\Delta})\boldsymbol{V}^{\mathrm{T}}$$

where $\tilde{\boldsymbol{w}}$ is a truncated vector by setting the first r elements of $\boldsymbol{w}$ to 0, and $\boldsymbol{S}_{\tilde{w}}(\boldsymbol{\Delta})_{ii} = \max(\boldsymbol{\Delta}_{ii} - \tau\tilde{w}_i, 0)$ is the generalized soft-thresholding operator with weight vector $\tilde{\boldsymbol{w}}$ [16], [17].

Proof. Based on the property of Frobenius norm, we get

$$\begin{aligned}
&\min \frac{1}{2}||\boldsymbol{X} - \boldsymbol{G}||_F^2 + \tau||\boldsymbol{X}||_{\boldsymbol{w},r} \\
&\Leftrightarrow \min -\mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{G}) + \frac{1}{2}\mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}) + \tau \sum_{i=r+1}^{q} w_i\sigma_i(\boldsymbol{X}) \\
&\Leftrightarrow \min \sum_{i=1}^{q}(-\sigma_i(\boldsymbol{X})\sigma_i(\boldsymbol{G}) + \frac{1}{2}\sigma_i^2(\boldsymbol{X})) \\
&+ \sum_{i=r+1}^{q} \tau w_i\sigma_i(\boldsymbol{X}) \\
&\Leftrightarrow \min \sum_{i=1}^{r}(-\sigma_i(\boldsymbol{X})\sigma_i(\boldsymbol{G}) + \frac{1}{2}\sigma_i^2(\boldsymbol{X})) \\
&+ \sum_{i=r+1}^{q}(-\sigma_i(\boldsymbol{X})\sigma_i(\boldsymbol{G}) + \frac{1}{2}\sigma_i^2(\boldsymbol{X}) + \tau w_i\sigma_i(\boldsymbol{X}))
\end{aligned} \tag{6}$$

where the second deduction comes from the von Neumann theorem, and the term $||\boldsymbol{G}||_F^2$ is omitted here for simplicity since it is a constant in the minimization with respect to $\boldsymbol{X}$.

Since the final term in (6) consists of simple quadratic equations, it is trivial to derive the optimal solution $\sigma_i^*(\boldsymbol{X})$ separately. With $i \leq r$, $\sigma_i^*(\boldsymbol{X}) = \sigma_i(\boldsymbol{G})$ can be obtained by derivative from the first-order optimality condition; With $i > r$, $\sigma_i^*(\boldsymbol{G}) = \max(\sigma_i(\boldsymbol{G}) - \tau w_i, 0)$ is a global optimal solution from Corollary 1 of [16]. Hence, by substituting $\tilde{\boldsymbol{w}}$ for $\boldsymbol{w}$, we achieve $\boldsymbol{X}^* = \boldsymbol{U}diag(\boldsymbol{\sigma}^*(\boldsymbol{X}))\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left and right singular matrices of $\boldsymbol{G}$.

Note that the RSVT operator exactly degenerates to PSVT [21] or GSVT [16] by letting $\boldsymbol{w} = \boldsymbol{1}$ or $r = 0$, respectively. Compared with PSVT, the introduction of non-descending vector $\boldsymbol{w}$ is significant for most real-world problems in visual applications, since that the larger singular values are always more informative, thus need to be suppressed less than the smaller ones. Moreover, when $\sigma_i(\boldsymbol{G}) \leq \tau w_i$ for $1 \leq i \leq r$, GSVT projects these σ_i to zero leading to deficient rank matrix $\boldsymbol{X}$, i.e., whose rank is lower than the predicted one. Conversely, RSVT implicitly enforces the resulting $\boldsymbol{X}$ to satisfy the predicted rank even when all thresholding values are small, which occasionally happens when parameter τ is suboptimally selected or the number of observations is limited.

III. Momentum Update and Rank Propagation based APG

In Section II, we specialize the typical WNNM and TNNM problems into a unified TRNM formulation. In this section, we focus on the acceleration of APG by virtue of adaptive momentum update and rank propagation scheme, so as to ensure convergence not only with less iterations but also with less computation at each iteration.

A. *Adaptive Momentum Update for APG*

Assuming that f and g are both convex, APG works well if one sets $\mu > L$ and choses β_k as

$$\begin{cases} \beta_k = (t_{k-1} - 1)/t_k \\ t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2, \end{cases} \tag{7}$$

where t_k is a real number and $t_{-1} = t_0 = 1$. However, the convergence by directly applying APG to Eq. (3) is slow due to the nonconvexity of TRNM regularization. Several researches have been conducted to accelerate APG under the nonconvex constraint [24], [25]. One intuitive scheme is to carry out specific extrapolation by means of historical information [25], [26]. Another efficient scheme is to employ certain search criterion for adaptively updating the step-size μ [23]. However, the optimal choice of the momentum parameters $\{\beta_k\}$ and $\{\mu_k\}$ is not clear yet from these two schemes, especially in the nonconvex case. In this section, we present an efficient algorithm that incorporates both extrapolation and step-size search through extending the function F in (1)as

$$\Theta_\delta(\boldsymbol{\Psi}, \boldsymbol{\Gamma}, \mu) = f(\boldsymbol{\Psi}) + \lambda g(\boldsymbol{\Psi}) + \frac{\delta\mu}{4}||\boldsymbol{\Psi} - \boldsymbol{\Gamma}||_F^2, \tag{8}$$

where $\boldsymbol{\Psi}$, $\boldsymbol{\Gamma} \in \Re^{m\times n}$, and $0 \leq \delta < 1$ is a scalar parameter. It can be noted that Eq. (8) adds a third term into Eq. (3) such that it can be more convex.

In terms of Eq. (8), we present the adaptive momentum update based proximal gradient method to minimize it, which is summarized in Algorithm 1. The basic idea is that we first employ the extrapolation step 1.1 and update step 1.2 from APG, then introduce the non-monotone decision step 1.3 and the line search step 1.4 for more flexibility and efficiency.

With respect to flexibility, if $\delta = 0$, we have $\boldsymbol{Y}^k = \boldsymbol{X}^k$ from step 1.1 since that $\beta_k = 0$ always holds from step 1. In this case, our decision step 1.3 degrades into

$$F(\boldsymbol{\Psi}) - \max_{(k-l)_+ \leq i \leq k} F(\boldsymbol{X}^i) \leq -\frac{c}{2}||\boldsymbol{\Psi} - \boldsymbol{X}^k||_F^2.$$

As a result, Algorithm 1 reduces to non-monotone proximal gradient (NPG) [23]. On the other hand, given $\delta \in (0, 1)$ and

$$\mu_k^0 = \mu_{\max} > L \ , \ \beta_k^0 \leq \frac{\sqrt{\delta(\mu_{\max} - L)\mu_{\max}}}{2(\mu_{\max} + L)}, \ \forall\, k \geq 0,$$

then the condition of step 1.3 can be naturally satisfied following from Lemma 1. In this case, both the introduced non-monotone decision and line search can be bypassed and our Algorithm 1 degenerates to the original APG method.

With respect to efficiency, the extended function (8) is sufficient descent (Lemma 1). The momentum search criterion is well-defined (Proposition 1). The overall convergence of Algorithm 1 can also be guaranteed (Theorem 2). Moreover, we propose a rank propagation RSVT operator in Subsection III.B, which achieves further acceleration at each iteration.

Lemma 1. When the conditions $\mu_k > L$ and $\beta_k \leq \frac{1}{2}\sqrt{\delta(\mu_k - L)\mu_{k-1}}\Big/(\mu_k + L)$ hold, then the generated sequences from Algorithm 1, i.e., $\{\boldsymbol{X}^k\}$ and $\{\mu_k\}$, satisfy the following inequality.

$$\begin{aligned}
&\boldsymbol{\Theta}_\delta(\boldsymbol{\Psi}, \boldsymbol{X}^k, \mu_k) - \boldsymbol{\Theta}_\delta(\boldsymbol{X}^k, \boldsymbol{X}^{k-1}, \mu_{k-1}) \\
&\leq \tfrac{L-(1-\delta)\mu_k}{4}||\boldsymbol{\Psi} - \boldsymbol{X}^k||_F^2,
\end{aligned} \tag{9}$$

where $\boldsymbol{\Theta}_\delta$ is the extended function defined in (8).

Proof. By turning step 1.2 of Algorithm 1 back to the Taylor expansion form and based on Remark 1 of [16], we have

$$\begin{aligned}&\left\langle \nabla f(\boldsymbol{Y}^k), \boldsymbol{\Psi}-\boldsymbol{Y}^k\right\rangle + \frac{\mu_k}{2}||\boldsymbol{\Psi}-\boldsymbol{Y}^k||_F^2 + g(\boldsymbol{\Psi})\\ &\le \left\langle \nabla f(\boldsymbol{Y}^k), \boldsymbol{X}^k-\boldsymbol{Y}^k\right\rangle + \frac{\mu_k}{2}||\boldsymbol{X}^k-\boldsymbol{Y}^k||_F^2 + g(\boldsymbol{X}^k)\end{aligned}$$

which implies that

$$\begin{aligned}&g(\boldsymbol{\Psi}) \le g(\boldsymbol{X}^k) + \left\langle \nabla f(\boldsymbol{Y}^k), \boldsymbol{X}^k-\boldsymbol{\Psi}\right\rangle\\ &+\frac{\mu_k}{2}||\boldsymbol{X}^k-\boldsymbol{Y}^k||_F^2 - \frac{\mu_k}{2}||\boldsymbol{\Psi}-\boldsymbol{Y}^k||_F^2\\ &= g(\boldsymbol{X}^k) + \left\langle \nabla f(\boldsymbol{Y}^k), \boldsymbol{X}^k-\boldsymbol{\Psi}\right\rangle - \frac{\mu_k}{2}||\boldsymbol{\Psi}-\boldsymbol{X}^k||_F^2\\ &+\mu_k\left\langle \boldsymbol{X}^k-\boldsymbol{\Psi}, \boldsymbol{X}^k-\boldsymbol{Y}^k\right\rangle\end{aligned} \tag{10}$$

On the other side, it can be seen from [12] that

$$f(\boldsymbol{\Psi}) \le f(\boldsymbol{X}^k) + \left\langle \nabla f(\boldsymbol{X}^k), \boldsymbol{\Delta}_{\boldsymbol{\Psi X}}\right\rangle + \frac{L}{2}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2, \tag{11}$$

where $\boldsymbol{\Delta}_{\boldsymbol{\Psi X}} = \boldsymbol{\Psi}-\boldsymbol{X}^k$. Similarly, by denoting $\boldsymbol{\Delta}_{\boldsymbol{XY}} = \boldsymbol{X}^k-\boldsymbol{Y}^k$ as well as combining (10) and (11), we can get

$$\begin{aligned}&f(\boldsymbol{\Psi}) + g(\boldsymbol{\Psi}) - f(\boldsymbol{X}^k) + g(\boldsymbol{X}^k)\\ &\overset{1}{\le} \frac{L-\mu_k}{2}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2 +\\ &(\mu_k||\boldsymbol{\Delta}_{\boldsymbol{XY}}||_F + ||\nabla f(\boldsymbol{X}^k)-\nabla f(\boldsymbol{Y}^k)||_F)||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F\\ &\overset{2}{\le} \frac{L-\mu_k}{4}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2 + \frac{(\mu_k+L)^2}{\mu_k-L}||\boldsymbol{\Delta}_{\boldsymbol{XY}}||_F^2\\ &\overset{1}{=} \frac{L-\mu_k}{4}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2 + \frac{(\mu_k+L)^2}{\mu_k-L}\beta_k^2||\boldsymbol{X}^k-\boldsymbol{X}^{k-1}||_F^2\\ &\overset{3}{\le} \frac{L-\mu_k}{4}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2 + \frac{\delta\mu_{k-1}}{4}||\boldsymbol{X}^k-\boldsymbol{X}^{k-1}||_F^2\\ &= \frac{L-(1-\delta)\mu_k}{4}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2 - \frac{\delta\mu_k}{4}||\boldsymbol{\Delta}_{\boldsymbol{\Psi X}}||_F^2\\ &+\frac{\delta\mu_{k-1}}{4}||\boldsymbol{X}^k-\boldsymbol{X}^{k-1}||_F^2\end{aligned} \tag{12}$$

where inequality 1 comes from Cauchy-Schwarz theorem, inequality 2 follows form Lipschitz smoothness of f as well as the mathematical relation $4abs \le a^2+4s^2b^2$, equality 1 coincides with step 1.1 of Algorithm 1, and inequality 3 stems from the constraint of β_k. By recalling (8) and rearranging all terms in (12), we complete the proof of Lemma 1.

Note that the descent property established in Lemma 1 is independent of the convexity of f and g, so it is applicable to (3) even with the concavity of TRNM regularization. Literally, it seems that Algorithm 1 requires two loops for optimization, which may yield more overall iterations as similar to the two-step strategy for TNNM [18]. However, with the fact that the sufficient descent property of $\boldsymbol{\Theta}_\delta$ can be satisfied given μ_k large enough and β_k small enough, we present Proposition 1 that the condition in step 1.3 can be easily satisfied given $\theta>1$ and $\eta<1$. Based on the above discussion, we show the convergence of Algorithm 1 in Theorem 2. The stop condition for Algorithm 1 is set as $||\boldsymbol{X}_{k+1}-\boldsymbol{X}_k||_F/||\boldsymbol{X}_k||_F \le \varepsilon$, where ε is a given tolerance.

Proposition 1. For the defined $\{\boldsymbol{X}^k\}$ and $\{\mu_k\}$ in Lemma 1, the condition in step 1.3 of Algorithm 1 can be satisfied within finite iterations of inner loop.

Proof. With the truth that $\mu_{max} \ge (L+2c)/(1-\delta) \ge L$ and $(1-\delta)\mu_{\max}-L \ge 2c$ in Algorithm 1, then from Lemma 1 we can get

$$\begin{aligned}&\boldsymbol{\Theta}_\delta(\boldsymbol{\Psi}, \boldsymbol{X}^k, \mu_{\max}) - \boldsymbol{\Theta}_\delta(\boldsymbol{X}^k, \boldsymbol{X}^{k-1}, \mu_{k-1})\\ &\le \frac{L-(1-\delta)\mu_{\max}}{4}||\boldsymbol{\Psi}-\boldsymbol{X}^k||_F^2 \le \frac{-c}{2}||\boldsymbol{\Psi}-\boldsymbol{X}^k||_F^2,\end{aligned}$$

which together with

$$\boldsymbol{\Theta}_\delta(\boldsymbol{X}^k, \boldsymbol{X}^{k-1}, \mu_{k-1}) \le \max_{(k-l)_+\le i\le k} \boldsymbol{\Theta}_\delta(\boldsymbol{X}^i, \boldsymbol{X}^{i-1}, \mu_{i-1})$$

guarantees the satisfied condition in step 1.3. So this proposition can be proved if $\mu_k = \mu_{\max}$ and $\beta_k \le 0.5(\delta(\mu_k-L)\mu_{k-1})^{0.5}/(\mu_k+L)$ will be satisfied within finite iterations. Due to the similarity of μ_k and β_k, here we only show the proof for μ_k. Note that $\mu_k > \mu_{\max}$ will not happen due to step 1.4. Let n_k denote the iterations required to get $\mu_k = \mu_{\max}$, it is obvious that $n_k = 1$ if $\mu_k^0 = \mu_{\max}$; Otherwise, we have $\mu_{\min}\theta^{n_k-1} \le \mu_k^0\theta^{n_k-1} < \mu_{\max}$ which implies that

$$n_k \le \left\lfloor \frac{\log(\mu_{\max})-\log(\mu_{\min})}{\log\theta} + 1 \right\rfloor,$$

and completes this proof.

Algorithm 1 Adaptive Momentum Update for Proximal Gradient

Input: $\boldsymbol{X}^0$, $\mu_0=1$, $\theta>1$, δ, $\eta\in[0,1)$, $c>0$, $\mu_{\max} \ge \frac{L+2c}{1-\delta} > \mu_{\min} > 0$, $\beta_{\max}\ge 0$, and $l \ge 1$.

Suggested settings:

$\theta=2$, $\delta=0.7$, $\eta=0.8$, $l=3$, $c=1e{-}4$, $\mu_{\min}=1e{-}6$, $\beta_{\max}=5$.

For $k=1,2,\ldots,maxit$

1. Set $\mu_k=\mu_k^0\in[\mu_{\min},\mu_{\max}]$ and $\beta_k=\beta_k^0\in[0,\delta\beta_{\max}]$.

1.1) Extrapolation: $\boldsymbol{Y}^k=\boldsymbol{X}^k+\beta_k(\boldsymbol{X}_k-\boldsymbol{X}_{k-1})$.

1.2) Perform RSVT with $\boldsymbol{G}=\boldsymbol{Y}^k-\frac{\nabla f(\boldsymbol{Y}^k)}{\mu_k}$, and $\tau=\frac{\lambda}{\mu_k}$.

1.3) If

$$\begin{aligned}&\boldsymbol{\Theta}_\delta(\boldsymbol{\Psi}, \boldsymbol{X}^k, \mu_k) - \max_{(k-l)_+\le i\le k} \boldsymbol{\Theta}_\delta(\boldsymbol{X}^i, \boldsymbol{X}^{i-1}, \mu_{i-1})\\ &\le -\frac{c}{2}||\boldsymbol{\Psi}-\boldsymbol{X}^{\boldsymbol{k}}||_F^2,\end{aligned}$$

where $(\cdot)_+$ is a nonnegative operator, then turn to step 2.

1.4) Set $\mu_k=\min\{\theta\mu_k,\mu_{\max}\}$, $\beta_k=\eta\beta_k$, and go to step 1.1.

2. Set $\boldsymbol{X}^{k+1}=\boldsymbol{\Psi}$, $k=k+1$, and turn to step 1.

End for

Output: $\boldsymbol{X}^k$;

Theorem 2. For the defined $\{\boldsymbol{X}^k\}$ and $\{\mu_k\}$ in Lemma 1, the following two statements hold.

i) the sequence $\{\boldsymbol{\Theta}_\delta(\boldsymbol{X}^{\vartheta(k)}, \boldsymbol{X}^{\vartheta(k)-1}, \mu_{\vartheta(k)-1})\}$ is nonincreasing and $\lim_{k\to\infty}\boldsymbol{\Theta}_\delta(\boldsymbol{X}^{\vartheta(k)}, \boldsymbol{X}^{\vartheta(k)-1}, \mu_{\vartheta(k)-1}) = \kappa$, where κ is some constant and $\vartheta(k) = \arg\max_i \boldsymbol{\Theta}_\delta(\boldsymbol{X}^i, \boldsymbol{X}^{i-1}, \mu_{i-1})$, $(k-l)_+ \le i \le k$.

ii) $\lim_{k\to\infty}||\boldsymbol{X}^{k+1}-\boldsymbol{X}^k||_F^2 = 0$.

Proof. Let $\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}_\delta(\boldsymbol{X}^{k+1}, \boldsymbol{X}^k, \mu_k)$ for simplicity. Step

1.3 of Algorithm 1 together with the definition of $\vartheta(k)$ leads to

$$\boldsymbol{\Theta}^{k+1} - \boldsymbol{\Theta}^{\vartheta(k)} \leq \frac{-c}{2}||\boldsymbol{X}^{k+1} - \boldsymbol{X}^{k}||_F^2 \leq 0 \tag{13}$$

Then we have

$$\begin{aligned}\boldsymbol{\Theta}^{\vartheta(k+1)} &= \max_{[k+1-l]_+ \leq i \leq k+1} \boldsymbol{\Theta}^{i} \\ &= \max\{\boldsymbol{\Theta}^{k+1}, \max_{[k+1-l]_+ \leq i \leq k} \boldsymbol{\Theta}^{i}\} \\ &\overset{1}{\leq} \max\{\boldsymbol{\Theta}^{\vartheta(k)}, \max_{[k+1-l]_+ \leq i \leq k} \boldsymbol{\Theta}^{i}\} \\ &\overset{2}{\leq} \max\{\boldsymbol{\Theta}^{\vartheta(k)}, \max_{[k-l]_+ \leq i \leq k} \boldsymbol{\Theta}^{i}\} \\ &= \boldsymbol{\Theta}^{\vartheta(k)},\end{aligned}$$

where inequality 1 and 2 stem from (13) and the definition of $\vartheta(k)$, respectively. This demonstrates that $\{\boldsymbol{\Theta}_\delta(\boldsymbol{X}^{\vartheta(k)}, \boldsymbol{X}^{\vartheta(k)-1}, \mu_{\vartheta(k)-1})\}$ is nonincreasing and its limitation exists (κ is denoted as the limiting point) by the fact that $\boldsymbol{\Theta}_\delta$ is bounded below by zero.

For statement ii), let $\Delta\boldsymbol{X}^k = \boldsymbol{X}^{k+1} - \boldsymbol{X}^k$, we first show (14) and (15) hold by induction.

$$\lim_{k\to\infty} \Delta\boldsymbol{X}^{\vartheta(k)-j} = 0, \tag{14}$$

$$\lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-j}) = \kappa, \ \forall j \geq 1. \tag{15}$$

When $j = 1$, by replacing k with $\vartheta(k) - 1$ in (13), we have

$$\boldsymbol{\Theta}^{\vartheta(k)} - \boldsymbol{\Theta}^{\vartheta(\vartheta(k)-1)} \leq \frac{-c}{2}||\Delta\boldsymbol{X}^{\vartheta(k)-1}||_F^2,$$

which further implies that

$$\lim_{k\to\infty} \Delta\boldsymbol{X}^{\vartheta(k)-1} = 0 \tag{16}$$

due to statement i). Taking both of (8) and (16) into consideration, we obtain

$$\begin{aligned}\kappa &= \lim_{k\to\infty} \boldsymbol{\Theta}_\delta(\boldsymbol{X}^{\vartheta(k)}, \boldsymbol{X}^{\vartheta(k)-1}, \mu_{\vartheta(k)-1}) \\ &= \lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-1} + \Delta\boldsymbol{X}^{\vartheta(k)-1}) + \frac{\delta\mu_{\vartheta(k)-1}}{4}||\Delta\boldsymbol{X}^{\vartheta(k)-1}||_F^2 \\ &= \lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-1})\end{aligned}$$

Hence, the conditions (14) and (15) hold for $j = 1$.

Suppose that (14) holds for $j = J > 1$ and let $\vartheta(k)-J-1 \geq 0$ without loss of generality, then by replacing k with $\vartheta(k)-J-1$ in (13), we have

$$\begin{aligned}&\boldsymbol{\Theta}^{\vartheta(k)-J} - \boldsymbol{\Theta}^{\vartheta(\vartheta(k)-J-1)} \leq \frac{-c}{2}||\Delta\boldsymbol{X}^{\vartheta(k)-J-1}||_F^2 \\ &\overset{1}{\Rightarrow}(\frac{c}{2} + \frac{\delta\mu_{\vartheta(k)-J-1}}{4})||\Delta\boldsymbol{X}^{\vartheta(k)-J-1}||_F^2 \\ &\leq \boldsymbol{\Theta}^{\vartheta(\vartheta(k)-J-1)} - F(\boldsymbol{X}^{\vartheta(k)-J}) \\ &\overset{2}{\Rightarrow} \lim_{k\to\infty} \Delta\boldsymbol{X}^{\vartheta(k)-J-1} = 0\end{aligned}$$

where deduction 1 and 2 follow from the definition of $\boldsymbol{\Theta}_\delta$ and the statement i), respectively. From this, we further obtain

$$\begin{aligned}&\lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-J-1}) \\ &= \lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-J} - \Delta\boldsymbol{X}^{\vartheta(k)-J-1}) \\ &= \lim_{k\to\infty} F(\boldsymbol{X}^{\vartheta(k)-J}) = \kappa\end{aligned}$$

Thus, (15) holds for $j = J + 1$, and the induction is proved.

We now return back for the statement ii). In fact, since that $k - l \leq \vartheta(k) \leq k$ always holds from the definition of $\vartheta(k)$. Thus $k - l - 1 = \vartheta(k) - j_k$ holds for any k given some $j_k \in [1, l + 1]$. Then we can easily conclude that

$$\begin{aligned}&||\Delta\boldsymbol{X}^{k-l-1}|| = ||\Delta\boldsymbol{X}^{\vartheta(k)-j_k}|| \leq \max_{1\leq j\leq l+1} ||\Delta\boldsymbol{X}^{\vartheta(k)-j}|| \\ &\Rightarrow \lim_{k\to\infty} \Delta\boldsymbol{X}^{k} = \lim_{k\to\infty} \Delta\boldsymbol{X}^{k-l-1} = 0\end{aligned}$$

where the deduction follows from (14).

B. Rank Propagation for RSVT Operator

The primary computational burden of Algorithm 1 lies in the iteratively performing of RSVT operator, whose cost for a $m \times n$ matrix is $O(mnq)$. In fact, for the singular values in RSVT operator, only those larger than $\tau\tilde{w}_i$ are referred, which offers the possibility of avoiding the full SVD operation. Following this idea, we present a rank propagation technique to speed up RSVT. Suppose $\boldsymbol{Z} \in \Re^{m\times n}$ has $p' \leq q$ singular values larger than $\tau\tilde{w}_i$ and let $\boldsymbol{U}_{p'}\boldsymbol{\Lambda}_{p'}\boldsymbol{V}_{p'}{}^T$ be the rank p' SVD of $\boldsymbol{Z}$, then Proposition 2 holds by the fact that the major formulation (5) of RSVT comprises of unitary invariant norms [27].

Proposition 2. Let $\Xi_{r,\tau}(\cdot)$ denote the RSVT operator and given orthonormal matrix $\boldsymbol{Q} \in \Re^{m\times p}$ with $p \geq p'$ and $\text{span}(\boldsymbol{U}_{p'}) \subseteq \text{span}(\boldsymbol{Q})$, we have $\Xi_{r,\tau}(\boldsymbol{Z}) = \boldsymbol{Q}\Xi_{r,\tau}(\boldsymbol{B})$, where $\boldsymbol{B} = \boldsymbol{Q}^T\boldsymbol{Z} \in \Re^{p\times n}$.

Proof. By substituting RSVT for the traditional singular value thresholding (SVT), the proof follows a similar derivation as [29, proposition III.1], hence we omit the details here.

Based on Proposition 2, step 1.2 of Algorithm 1 can be avoided by instead conducting partial RSVT on smaller matrix $\boldsymbol{B}$ following two main routines: 1) Generating an orthogonal matrix $\boldsymbol{Q}$, and 2) Performing RSVT on $\boldsymbol{B}$. This strategy has already been intensively investigated in some pioneer works. The randomized SVD [28] sequentially perform these two routines with a fixed p. However, the unknown parameter p is difficult to estimate in advance. To handle this issue, several variants of randomized SVD [28], [29] have recourse to the incremental implementation through achieving the complete SVD block by block until certain given precision is satisfied. Nevertheless, the incremental scheme may be vulnerable to round-off deviation, thus some extra orthogonalization operations should be imposed for acceptable accuracy, which inevitably leads to more time complexity.

In practice, two facts should be noted for the rank evolving process of Algorithm 1. First, the amount of the singular values those greater than $\tau\tilde{w}_i$ shall progressively approaches to the real rank of the target matrix. Second, the precision in the initial steps is often poorer due to a loosely approximation at that stage. Inconsideration of these two, a natural question rises: Whether performing the full RSVT operator is indispensable when the target variable deviates far from the final solution? These facts and the question motivate us to employ the rank propagation strategy of implementing RSVT from inexactness to exactness as presented in Algorithm 2.

Using the rank propagation strategy, Algorithm 1 begins with an loose guess of subspace rank p^0, and then iteratively

performs randomized SVD for the RSVT operator. With the obtained rank b in the kth iteration, the estimated rank p^{k+1} shall be computed as

$$p^{k+1} = \begin{cases} b + a & if\ b \leq p^k \\ \lfloor p^k + \rho q \rfloor & if\ b > p^k \end{cases} \tag{17}$$

where a is an integer earning more rank information, $\rho \in (0, 1)$ denotes a scale factor, and $\lfloor \cdot \rfloor$ represents the floor operation. In the case of $b \leq p^k$, it demonstrates that p^k is over estimated resulting in excessively large block size and undue computation cost, thus (17) reassigns p^{k+1} to a smaller value with slightly over-sampling rate a. In the other case that $b > p^k$, it indicates p^k is smaller than that required to approximate the authentic rank, so (17) reassigns p^{k+1} to a larger value for better capturing the whole energy. Empirically, Algorithm 2 works well when we set a=2 and $\rho = 0.05$. Compared to the original RSVT operator, although Algorithm 2 also involves a full SVD operation, it performs on the smaller matrix $\boldsymbol{B} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Z}$ and costs only $O(np^{k2})$ time complexity, which is clearly more efficient due to $p^k \ll q$. Therefore, our scheme of optimistically propagating rank shall bring better computational efficiency without any predicted rank information or given precision.

Algorithm 2 RSVT with Rank Propagation

Input: $\boldsymbol{Z}^k = \Re^{m\times n}$, $r > 0$, $\tau = \lambda/\mu_k$, $a > 0$, $\rho \in (0, 1)$, $p^k \in (r, q]$.
Output: $\boldsymbol{Z}^{k+1}$ and p^{k+1}.

```
Generate Gaussian matrix Ω ∈ ℜ^{n×p^k};
Let A = ZΩ;
Q = qr(A); //for othogonalization
Q = powermethod(A, Q); //optional, suggested by [28];
Perform SVD operation as: [U, Λ, V] = SVD(Q^T Z);
Let b be the amount of elements Λ_ii greater than τ w̃_i, i = 1, ..., q;
If b ≤ r
Z^{k+1} = QU(:, 1 : r)Λ_{1:r} V(:, 1 : r)^T;
else if
Z^{k+1} = QU S_{w̃}(Λ) V^T;
End if
Compute p^{k+1} as (17).
```

IV. Applying TRNM to Matrix Completion and Subspace Clustering

To validate the effectiveness of the proposed TRNM constraint and MURP scheme, we apply them to two typical data mining applications: matrix completion and subspace clustering. For matrix completion, similar to TNNM [18], [21], [22], TRNM is applied directly to the input matrix for structure recovery. For subspace clustering, like the TSCLR [30], we propose a TRNM constrained SC method to recover the "low-rank + temporal" structure of the data matrix. Although TRNM can also be applied in image denoising following the similar steps of [17], we move the detailed discussions to the supplemental material due to space limit.

A. TRNM in Matrix Completion

Given an incomplete data matrix $\boldsymbol{M}$, let $\boldsymbol{P}_\Omega : \Re^{m\times n} \rightarrow \Re^{m\times n}$ denote an orthogonal projection onto the subspace of matrices that have nonzero entries corresponding to the observed components in a domain Ω, i.e.,

$$[\boldsymbol{P}_\Omega(\boldsymbol{M})]_{ij} = \begin{cases} \boldsymbol{M}_{ij}, & if\ (i,j) \in \Omega \\ 0, & otherwise \end{cases} \tag{18}$$

Then, the matrix completion problem via TRNM (MCTRNM) can be formulated as follows:

$$\min_{\boldsymbol{X}} \frac{1}{2}||\boldsymbol{P}_\Omega(\boldsymbol{X} - \boldsymbol{M})||_F^2 + \lambda||\boldsymbol{X}||_{\boldsymbol{w},r} \tag{19}$$

It has been observed that (19) becomes easier by recovering rows with more known entries preferentially in each step [22]. For this purpose, we assign the rows of the residual $\boldsymbol{E} = \boldsymbol{X} - \boldsymbol{M}$ with different weights. Accordingly, problem (19) is modified as

$$\min_{\boldsymbol{X}} \overbrace{\frac{1}{2}||\boldsymbol{P}_\Omega(\boldsymbol{T}(\boldsymbol{X} - \boldsymbol{M}))||_F^2}^{f(\boldsymbol{X})} + \lambda||\boldsymbol{X}||_{\boldsymbol{w},r} \tag{20}$$

where $\boldsymbol{T} \in \Re^{m\times n}$ is a diagonal matrix. Suppose $\boldsymbol{s}_{ori} = (s_1, s_2, \ldots, s_m)$ is a vector with its element s_i being the number of observed entries in the ith row of input matrix $\boldsymbol{M}$. By reordering $\boldsymbol{s}_{ori}$ into a non-ascending vector $\boldsymbol{s}_{sort} = (s_{i_1}, s_{i_2}, \ldots, s_{i_m})$, where $i_j, j \in \{1, 2, \ldots, m\}$ is the index of s_{i_j} in s_{ori}, then the corresponding weight $\boldsymbol{T}_{sort} = diag(t_1, \ldots, t_m)$ can be given by

$$t_{i_j} = \begin{cases} 1, & 1 \leq j \leq \beta \\ \frac{\theta - 1}{m - \beta} + t_{i_{j-1}}, & \beta < j \leq m \end{cases} \tag{21}$$

where $\theta > 1$ and $0 < \beta < m$. Eq. (21) ensures that the rows with more known entries are given smaller weights than others, which further leads to easier recovery than others. We can now apply Algorithm 1 for solving problem (20), for which the gradient of $f(\boldsymbol{X})$ is

$$\nabla f(\boldsymbol{X}) = \boldsymbol{P}_\Omega(\boldsymbol{T}^2(\boldsymbol{X} - \boldsymbol{M})). \tag{22}$$

Since that $||\nabla f(\boldsymbol{X}_1) - \nabla f(\boldsymbol{X}_2)||_2 = ||\boldsymbol{P}_\Omega(\boldsymbol{T}^2(\boldsymbol{X}_1 - \boldsymbol{X}_2))||_2 \leq ||\boldsymbol{X}_1 - \boldsymbol{X}_2||_2$, where the inequality comes from the fact that $t_i \leq 1$ for $i \in \{1, 2, \ldots, m\}$, then the Lipschitz constant for (20) is L=1.

B. TRNM in Subspace Clustering

Subspace clustering methods differ in the constraints enforced on the coefficient matrix. Recently, some sequential constraint based approaches, e.g., ordered subspace clustering (OSC) [31] and temporal subspace clustering [30], have been proposed to cluster data drawn from a temporally or spatially ordered union of subspaces, and obtain overwhelming advantage against the traditional methods. However, all these methods neglect the low-rank property that possibly resides in data. By applying TRNM to the Laplacian regularized TSC (TSCLR) model [30] and following the self-expressive

property [10], [11], our subspace clustering model via TRNM (SCTRNM) is formulated as follows:

$$\min_{\boldsymbol{X}} \overbrace{\frac{1}{2}||\boldsymbol{M}-\boldsymbol{M}\boldsymbol{X}||_F^2+\frac{\lambda_1}{2}\mathrm{tr}(\boldsymbol{X}\boldsymbol{L}_T\boldsymbol{X}^{\mathrm{T}})}^{f(\boldsymbol{X})}+\lambda_2||\boldsymbol{X}||_{\boldsymbol{w},r} \tag{23}$$

where λ_1 and λ_2 are two balance parameters, $\boldsymbol{M}$ is the input matrix with multiple subspaces, and $\boldsymbol{L}_T \in \Re^{m\times n}$ is the temporal Laplacian regularization [30].

The gradient of $f(\boldsymbol{X})$ in (23) can be computed as

$$\nabla f(\boldsymbol{X}) = \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{X}-\boldsymbol{M}^T\boldsymbol{M}+\lambda_1\boldsymbol{X}\boldsymbol{L}_T, \tag{24}$$

from which $||\nabla f(\boldsymbol{X}_1)-\nabla f(\boldsymbol{X}_2)||_2 \le ||\boldsymbol{M}^T\boldsymbol{M}||_2||\boldsymbol{X}_1-\boldsymbol{X}_2||_2+\lambda_1||\boldsymbol{X}_1-\boldsymbol{X}_2||_2||\boldsymbol{L}_T||_2$ holds, and we can further get the Lipschitz constant $L=||\boldsymbol{M}^T\boldsymbol{M}||_2+\lambda_1||\boldsymbol{L}_T||_2$.

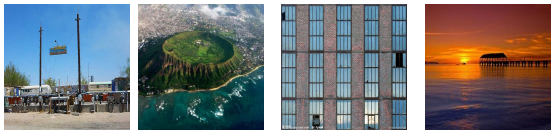
V. Experimental Results

We evaluate the performance of MCTRNM on image inpainting, which is a typical matrix completion problem. Some improved low-rank constraints based on NNM, including IRNN [12], WNNM [16], and TNNM [18], [19], are selected as the competing methods. For TNNM, we further compare its variants, such as TNN-ADMM [18], TNN-APG [18], PSSV [21], and EWRE [22], to validate the efficiency of different optimization schemes. For MCTRNM, we adopt $w_i = 1/(\sigma_i^p(\boldsymbol{X})+\epsilon)$ with p=0.8 as the default weight constraint following [4] and [16]. The main parameters r and λ are traversed in $\{1,2,\dots 9\}$ and $\{1e2,\dots,7e2\}$, respectively, to report the best results. Unless otherwise stated, the parameters θ and β for weight matrix $\boldsymbol{T}$ are set as 1.1 and 100 consistently. For MURP, we simply choose $\{\beta_k^0\}$ by Eq. (7) with β_k^0 in place of β_k. The other parameters are permanently set as the suggested values shown in Algorithm 1. The maximum iterative number and the convergence tolerance are set as *maxit*=1000 and ε =$1e{-}5$ for all the competing methods to ensure a fair comparison.

We also compare another proposed approach, SCTRNM, with the state-of-the-art SC algorithms, including FSCNN [8], IRIALM [9], NSGLRR [10], IBDLR [11], TSCLR [30], and OSC [31], for well-known clustering problems. For SCTRNM, the parameters r and p are set as c and 1 in default, where c is the target clusters in specific experiments. The balance parameters λ_1 and λ_2 are all traversed in $\{1e{-}3, 1e{-}2,\dots,1e3\}$ for the best results. The maximum iterative number and the convergence tolerance are set as *maxit*=200 and $\varepsilon = 1e{-}4$, respectively. Other parameters of all competitors are tuned as suggested in original papers to achieve the best performance. The program platform is with Intel Core i7-5500U 2.40GHz and 8GRAM.[1]

A. *Image Inpainting*

The samples in different resolutions listed in Fig. 1, including 5 color images and 2 range data, are used in this subsection. We adopt the Peak Signal-to-Noise Ratio (PSNR) [17], which is a commonly used criterion in image inpainting, to evaluate the quality of recovery results. The execution time is also reported by recovering an incomplete image with 10 runs and generating the average results in seconds.



(1) 341 × 600 (2) 300 × 300 (3) 278 × 410 (4) 210 × 350

(5) 1300 × 975 (6) 496 × 718 (7) 375 × 450

Fig. 1: The used color images (1-5) and range data (6-7).

1. Text Removal

Text removal is a real world problem in most visual applications. By regarding the corrupted pixels as missing elements, the text removal mission can be directly considered as a matrix completion issue. For all the samples listed in Fig. 1, we randomly pave them some texts and then perform image recovery using all the competing approaches. Notice that the natural images 1-5 contain three color channels in RGB setting; we recover them independently and then unify the results on average as the final performance. Part of the visual results, the runtime, and the PSNR values can be found in Fig. 2, Fig. 3, and Table I. Notice that in Fig. 3, we scale the result from sample 5 to 1/20 for balanced visualization, since it requires much too more execution time than others.

The first observation from Fig. 2 is that all the methods can recover the true data from text corruption. It makes sense for these outcomes since that all the selected constraints have been verified with appealing performance theoretically and empirically. However, with more careful observations, our method still achieves better visual quality with less ghost shadow of text remnants, especially than the result from WNNM, whose image is full of abnormal points. From Table I, it can be seen that the PSNR results of WNNM are inferior to others in most scenarios. IRNN and TNNM based methods share similar recovery performance in this experiment. However, their PSNR is lower than ours in almost all cases. On average of samples 1-7, MCTRNM achieves 0.24dB-2.6dB improvement over other competing methods. Such an improvement is notable since all of these methods have been proven to be superior to many typical image inpainting approaches.

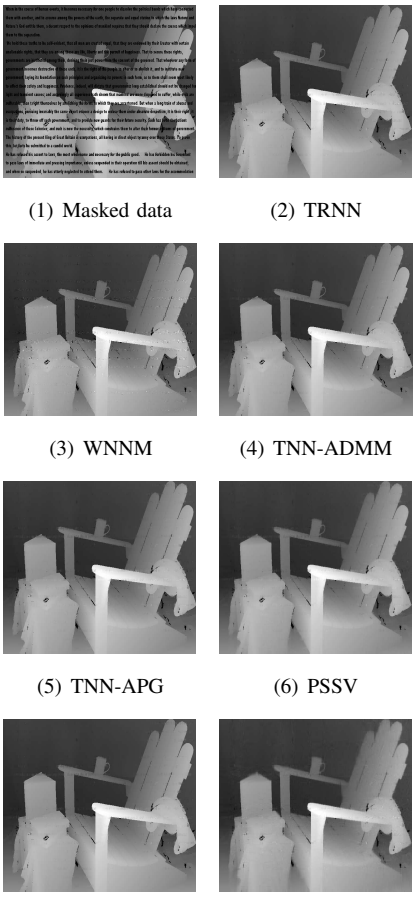
Moreover, from Fig. 3 and Table I (the values in brackets), our method runs much more efficient than the competitors, which not only requires least overall runtime but also converges with less iterations. WNNM behaves the slowest among all the seven methods; it fails in convergence until the maximum iterations reached in most cases. IRNN, though better than WNNM, still requires much more iterations than others. For the TNNM based methods, while PSSV runs faster than ADMM and APG by adopting a partial SVT operator for

[1]More experimental results and the source code are provided as the supplemental material and will be released on: http://www.escience.cn/people/zhengjianwei/index.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2916986, IEEE Transactions on Industrial Informatics

TABLE I: The PSNR results and required iterations of all the seven methods on samples 1-7 under text corruption.

Sample	IRNN	WNNM	TNN-ADMM	TNN-APG	PSSV	EWRE	MCTRNM
1	23.38 (1071)	22.28 (3000)	23.49 (836)	23.46 (435)	23.18 (280)	23.48 (196)	**23.60** (102)
2	22.04 (984)	18.71 (3000)	22.10 (1148)	**22.11** (420)	21.98 (231)	22.09 (240)	22.06 (80)
3	25.54 (867)	20.96 (1088)	25.81 (484)	25.83 (452)	25.89 (234)	25.83 (159)	**26.00** (75)
4	34.61 (799)	34.56 (2571)	34.57 (1116)	34.43 (354)	27.18 (316)	34.62 (201)	**34.71** (192)
5	24.58 (2663)	23.98 (3000)	24.35 (726)	24.35 (766)	24.21 (465)	24.16 (183)	**25.03** (177)
6	27.33 (502)	23.70 (1000)	27.64 (295)	27.64 (161)	27.62 (135)	27.56 (68)	**27.92** (27)
7	23.00 (439)	20.31 (1000)	23.19 (527)	23.21 (264)	22.93 (121)	23.19 (72)	**23.37** (26)



(1) Masked data (2) TRNN

(3) WNNM (4) TNN-ADMM

(5) TNN-APG (6) PSSV

(7) EWRE (8) MCTRNM

Fig. 2: Visual comparison on range data 6. (1) Masked data. (2)-(8) Recovered images of IRNN, WNNM, TNN-ADMM, TNN-APG, PSSV, EWRE, and MCTRNM, respectively.

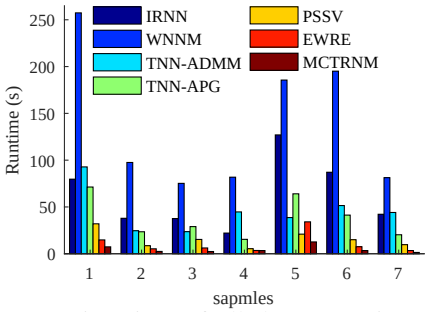


Fig. 3: The execution time of all the competing methods on samples 1-7 with text noise.

avoiding the two-step scheme [21], it runs slower than EWRE. The advantage of EWRE hinges on recovering different rows (or columns) under scheduled order. However, this scheme also causes additional matrix multiplications, whose cost is $O(mnq)$ for any two matrices $\boldsymbol{A}, \boldsymbol{B} \in \Re^{m\times n}$.

2. Random Pixels Missing

Due to coding or transmission issues, partial pixels of the images may be occasionally missing. In this subsection, we randomly mask part pixels of the input images, and then evaluate the effectiveness and efficiency of all the compared methods. Table II shows the quantitative results, i.e., PSNR and runtime (in brackets), for all the tested samples with 60% missing ratio. An overall impression observed from Table II is that MCTRNM achieves the highest PSNR in all cases. WNNM again is inferior to others both in PSNR result and in execution time. IRNN, though outperforming WNNM, performs poorer than the remaining five in most cases. The PSNR results of the TNNM based methods are roughly equal to each other on average, so the one with better efficiency will be more favorable for practicability. TNN-APG converges with less iterations than TNN-ADMM. Nevertheless, it requires a little more runtime in each iteration obtaining the objective value [18]. Thus, we can deem that TNN-ADMM and TNN-APG share similar efficiency level. PSSV converges much faster than TNN-ADMM and TNN-APG by performing the PSVT operator in one-step strategy, but it runs slower than EWRE in most cases. However, EWRE still ranks behind our method. The appealing efficiency of MCTRNM stems from the momentum adaption and rank propagation scheme, where the former results in a rapid convergence and the latter reduces per-iteration computational burden.

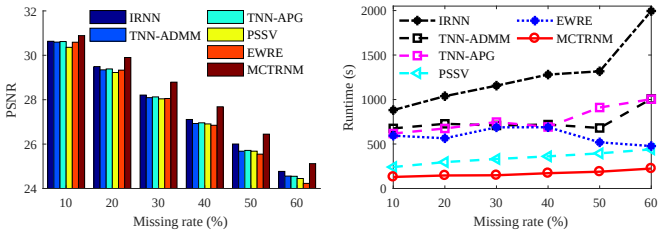
Fig. 4 further shows the PSNR and runtime results of six methods, excluding WNNM that performs the poorest in previous experiments, on image 5 under 10%-50% mask ratios. For all ratios of missing entries in Fig. 4 (1), the PSNR results get lower with the increasing of mask level. It is reasonable because more pixels are available when there are smaller percent of outliers. Again, we can observe that our method achieves the highest PSNR in all tests. Overall, the gains of MCTRNM over the second one are 0.26dB, 0.41dB, 0.58dB, 0.57dB, 0.45dB, and 0.35dB along with the increasing of mask ratio. In Fig. 4 (2), a phenomenon can be observed that all the competitors run slower along with the increasing of the missing ratio. It is also reasonable since that an image with more pixels corrupted always holds less meaningful information. Interestingly, we observe that EWRE behaves reversely in this experiment. This can be attributed to the addition of several extra variables generating better recovery effectiveness, which requires to be finely tuned and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2916986, IEEE Transactions on Industrial Informatics

TABLE II: Quantitative results on samples 1-7 with 60% random missing pixels .

Sample	IRNN	WNNM	TNN-ADMM	TNN-APG	PSSV	EWRE	MCTRNM
1	21.30 (108.8)	19.03 (171.5)	21.32 (145.6)	21.34 (54.5)	21.33 (27.2)	21.31 (18.9)	**21.45** (13.2)
2	20.75 (37.7)	18.09 (48.1)	21.14 (41.3)	21.20 (23.6)	21.37 (9.1)	21.27 (7.8)	**21.37** (3.5)
3	23.92 (34.8)	21.20 (66.2)	24.27 (36.6)	24.35 (37.1)	24.54 (16.4)	24.37 (14.4)	**24.63** (4.9)
4	32.18 (28.1)	30.92 (51.7)	32.13 (33.3)	32.14 (10.9)	31.02 (4.9)	31.92 (4.8)	**32.59** (4.0)
5	24.77 (1995)	22.46 (3562)	24.56 (1007)	24.55 (1004)	24.45 (440)	24.23 (477)	**25.12** (224)
6	27.11 (62.9)	24.50 (165.9)	27.31 (80.7)	27.35 (46.4)	27.39 (12.8)	27.33 (8.3)	**27.49** (4.5)
7	22.92 (40.4)	20.85 (52.4)	23.12 (31.9)	23.15 (26.3)	23.16 (13.0)	23.13 (11.3)	**23.21** (5.7)

may cause side-effect to the overall convergence. MCTRNM runs most efficiently again in all tests, and is also more stable under different missing rates.



(1) PSNR (2) Runtime

Fig. 4: PSNR and runtime of six competing methods on color image 5 under different missing rates.

B. *Data Clustering*

In this subsection, we investigate the performance of SCTRNM by conducting experiments on video scene segmentation, gesture segmentation, and regular image clustering. Three metrics, i.e., accuracy (AC) [10], normalized mutual information (NMI) [10], and execution time are adopted to evaluate the clustering performance.

1. Video Scene Segmentation

The purpose of this experiment is to cluster different scenes in video sequences. Following OSC, the data are drawn from an animation freely available in the Internet Archive [31]. The videos are separated into sequences containing three scenes, where each sequence consists of approximately 80-300 frames. There are overall 19 sequences from the video. The scenes to be segmented involve remarkable translation and morphing of objects within the scene and occasionally camera or perspective variations.

Since the frame size of original video is extremely high dimensional, we down-sample all the frames to a resolution of 129×96 for computational tractability. Each frame in the sequence is vectorized to $\boldsymbol{x}_i \in \Re^{12384}$ and concatenated with consecutive columns to form the input matrix $\boldsymbol{X}$. All the sequences are then corrupted by Gaussian noise with magnitude at 30% to generate clustering results that can be found in Table III. From this table, Our first observation is that SCTRNM generally outperforms the state-of-the-art methods both in AC and NMI. Without consideration of temporal constraint, FSCNN, IRIALM, NSGLRR, and IBDLR all underperform our method with notable gap. Although TSSCLR and OSC achieve closer performance to ours, they require more than twice as much runtime compared with SCTRNM. Besides, the standard deviation of our proposed method also outperforms other competitors in most cases, which further demonstrate that SCTRNM is more stable and behaves more predictably. Note that FSCNN runs the fastest in this experiment, yet it lags behind SCTRNM noticeably with respect to AC and NMI, i.e., 9.33% and 9.69%, respectively.

TABLE III: Clustering accuracies, standard deviation, and runtime of all methods on video sequences.

Method	AC (%)		NMI (%)		Times
	Mean	Std	Mean	Std	
FSCNN	85.96	16.46	80.30	19.97	45.52
IRIALM	77.63	18.30	64.70	28.45	420.4
NSGLRR	57.50	12.62	40.48	23.98	558.8
IBDLR	86.77	15.54	77.56	23.91	70.51
TSSCLR	94.66	8.79	88.10	14.59	155.6
OSC	94.60	9.81	88.33	**14.02**	188.24
SCMATN	**95.29**	**7.45**	**89.99**	14.53	70.38

2. Gesture Segmentation

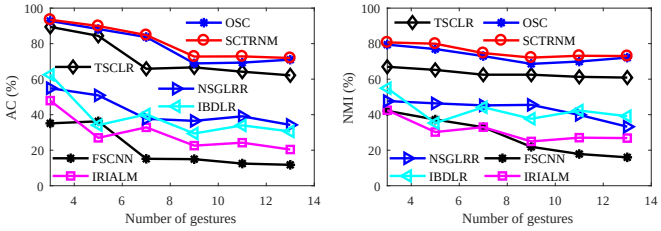
In this subsection, we use the Keck gesture data [31] that consists of 14 different gestures for evaluation. For each gesture, there are three sequences performed by different subjects. In each sequence, the same gesture is repeatedly performed with three times. The original resolution of each frame is 480×640. Following [31],we resize each frame to the resolution of 80×106 to speed up the computation. Different gesture sequences of each subject are concatenated into a single long video sequence, which is further used as the input matrix $\boldsymbol{X}$.

Fig. 5 plots the clustering performance of all seven methods versus different number of gestures. All results are averaged from 10 runs of randomly selected gesture sequences. Benefiting from the temporal information, TSCLR, OSC, and SCTRNM consistently and significantly outperform the other methods. Among these three, the performance of our proposed method again ranks the highest with regard to both AC and NMI. To further demonstrate the advantage of SCTRNM, Fig. 6 shows the clustering visualization containing all the 14 gestures, by rendering clusters as different colors. The numerical results are also given in the subcaption for clearer comparison. We can see that the sequential subspace structure of TSCLR is more confused than OSC and SCTRNM, which leads to the poorer numerical results. OSC roughly generates the correct ordered clusters across columns. However, the span of certain columns distinctly deviates from the ground truth. Finally, SCTRNM keeps well-ordered the balanced structures and generates the least chaos among all the visualization results, which confirms the superiority of TRNM constraint.

In addition, an interesting phenomenon is noticeable from Fig. 6. It shows that each cluster is led by a short extra block caused by the preliminary motion of the performer. This is a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2916986, IEEE Transactions on Industrial Informatics

positive result since it demonstrates that our approach is capable of discovering implicit clusters, which brings fine property for some high-level applications such as video understanding. In combining with Fig. 5 and Fig. 6, OSC achieves close performance to our method with a difference less than 2% numerically. However, the running time of SCTRNM is about 100 times faster than OSC, which ensures the more remarkable application feasibility.



(1) AC (2) NMI

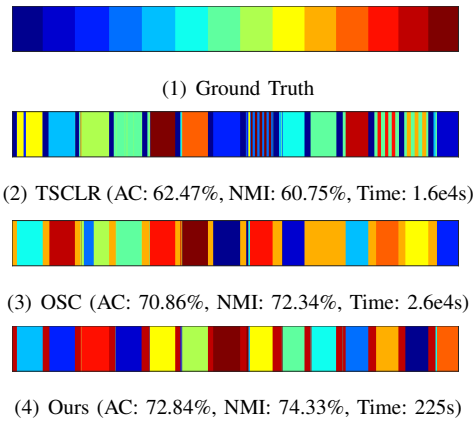
Fig. 5: Clustering performance (%) on the Keck dataset with respect to different number of gestures.

3. Regular Image Clustering

The aim of this experiment is to cluster unique objects from a set of captured images. We draw the dataset from the COIL database that contains 20 objects. The images of each object are sequentially taken 5 degree apart as the object is rotated on a turntable and each object has 72 images. The size of each image is 40×40. All the images are kept contiguous, i.e., unique object do not mix, so that we can exploit the spatial information from the final input matrix $\boldsymbol{X} \in \Re^{1600\times 1440}$.

Fig. 7 shows the quantitative results of AC and NMI for this experiment. Similar to the previous experiments, SCTRNM gives the best results outperforming other competitors. On average of AC and NMI, the performance improvement of SCTRNM is 25.76%, 44.92%, 24.60%, 22.97%, 9.64%, and 11.00%, over FSCNN, IRIALM, NSGLRR, IBDLR, TSCLR, and OSC, respectively.

In Fig. 8, we further exhibit the normalized objective values versus runtime of all methods under their optimally tuned parameters. As can be seen from the figure, the proposed approach is computationally more efficient compared to other



(1) Ground Truth

(2) TSCLR (AC: 62.47%, NMI: 60.75%, Time: 1.6e4s)

(3) OSC (AC: 70.86%, NMI: 72.34%, Time: 2.6e4s)

(4) Ours (AC: 72.84%, NMI: 74.33%, Time: 225s)

Fig. 6: Clustering visualization and the numerical results on Keck dataset.

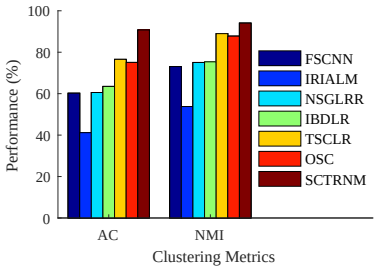


Fig. 7: Clustering performance (%) on COIL20 dataset.

iterative based clustering methods. Numerically, the runtime required for convergence of SCTRNM is about 1/100 of that by TSCLR and OSC. The objective values of IBDLR in Fig. 8 decrease more smoothly than our method. However, the computation of Eigen decomposition costs $O(n^3)$ iteratively, which restricts IBDLR from convergence in short time. For SCTRNM, the bumps shown in the figure come from the non-monotone nature of Algorithm 1 and the rank approximation operation. Especially, at the initial stage of optimization, the predicted rank deviates far away from the authentic one, which leads to a reverse effect on the cost function. As the iteration continues, the real rank of matrix $\boldsymbol{X}$ is gradually revealed, which impels Algorithm 1 to converge in an efficient manner.

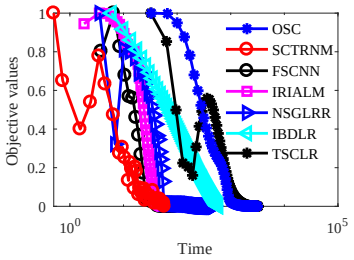


Fig. 8: The normalized objective values versus elapsed time on COIL20 dataset.

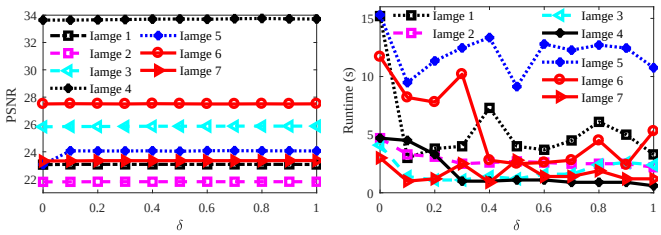
C. *Component Analysis*

The efficiency of our Algorithm 1 mainly stems from two factors: (i) The momentum adaptation ruled by the extended function (8); and (ii) The rank propagation RSVT operator on a smaller matrix. Their respective contributions are evaluated on image inpainting under text corruption and shown in Fig. 9 and Table IV, respectively.

Notice that when given $\delta = 0$, Algorithm 1 degenerates to the NPG algorithm. In Fig. 9 (2), though the curves on different samples fluctuates strongly with the changes of δ, the execution time of Algorithm 1 with $\delta > 0$ is stably less than that when $\delta = 0$. This result together with Fig. 9 (1) verifies that a fine-tuned δ prompts faster convergence while holding close performance. Analogously, in Table IV, the substitution of full RSVT operator with our rank propagation scheme has trivial impact on recovery results, but facilitates better efficiency, especially in the case of high resolution, e.g., samples 5 and 6. Basically, the rank propagation RSVT scheme ensures less per-iteration runtime with sacrifice of a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2916986, IEEE Transactions on Industrial Informatics

little more iterations. In practice, the overall speed-up always holds due to the fact that the intrinsic rank of common visual data is much smaller than the spatial dimension or sample size.



(1) PSNR (2) Runtime

Fig. 9: PSNR and runtime of MCTRNM on text corrupted samples 1-7 under various δ.

TABLE IV: PSNR, runtime, and consumed iterations (iter) of Algorithm 1 on image inpainting from text corruption with and without rank propagation scheme.

Sample	With rank propagation			Without rank propagation		
	PSNR	time	iter	PSNR	time	iter
1	23.60	7.3	111	23.60	7.8	102
2	22.06	3.3	89	22.04	4.0	86
3	26.00	2.5	93	25.97	3.1	75
4	34.71	3.8	129	34.72	5.9	112
5	25.04	127.2	248	25.03	261.1	236
6	27.92	3.0	40	27.95	6.5	31
7	23.37	2.1	31	23.34	3.0	26

On the other hand, the effectiveness of our model mainly relies on the proposed TRNM regularization, which is further controlled by the parameters r and p. Fig. 10 plots the accuracy of TRNM versus these two parameters in the subspace clustering application on COIL20 dataset. It can be observed that the variations coming from both r and p have a regular and unimodal trend when one of them is fixed. For p, the best value lies in the interval [0.4, 0.8] in most cases; while for r, a wide span centered at the cluster number always leads to a desirable result. These properties make the two parameters easy to be determined, which further promotes feasibility for practical problems.

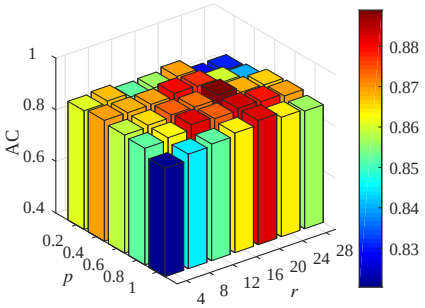


Fig. 10: Accuracy versus p and r on COIL20 database.

VI. Conclusions

In this paper, the TRNM constraint is specified for low-rank matrix approximation. On one hand, TRNM is effective to fit into real-world problems by imposing different treatment for sequential singular values; On the other hand, the target rank of the matrix can be better warranted by the truncation parameter r. The RSVT operator is further provided to facilitate the solving of TRNM constrained problems. Moreover, based on an extended function, we propose a more efficient APG method, namely MURP, for the TRNM related applications. There are several merits in the proposed MURP scheme. First, the extended cost function is guaranteed with a sufficient descent property iteratively. Second, the momentum parameters can be adaptively updated within given span, which avoids the tedious searching process needed for traditional APG model. Finally, the rank propagation RSVT operator can optimistically approximate the main matrix action without requiring any prior information. We validate the performance of the proposed method on practical MC and SC problems such as image inpainting, video scene segmentation, gesture segmentation, and image clustering. Experimental results demonstrate the superiority of TRNM over up-to-date methods both in effectiveness and efficiency.

References

[1] J. Bigot, C. Deledalle, and D. Feral, "Generalized SURE for optimal shrinkage of singular values in low-rank matrix denoising," *Journal of Machine Learning Research*, vol. 18, no. 11, pp. 1-50, 2017.

[2] J. Wen, N. Han, X. Z. Fang, L. K. Fei, K. Yan, and S. H. Zhan, "Low-rank preserving projection via graph regularized reconstruction," *IEEE Trans. Cybernetics*, vol. 49, no. 4, pp. 1279-1291, 2019.

[3] Y. Liu, Z. Long, H. Huang, and C. Zhu, "Low CP rank and Tucker rank tensor completion for estimating missing components in image data," *IEEE Trans. Circuits and Systems for Video Technology*, DOI: 10.1109/TCSVT.2019.2901311, 2019.

[4] Y. Xie, S. H. Gu, Y. Liu, W. M. Zuo, W. S. Zhang, and L. Zhang, "Weighted Schatten p-norm minimization for image denoising and background subtraction," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4842-4857, 2016.

[5] D. Q. Zhou, X. W. Chen, G. Y. Cao, and X. G. Wang, "Unsupervised video matting via sparse and low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, https://DOI: 10.1109/TPAMI.2019.2895331.

[6] C. Xu, Z. C. Lin, and H. B. Zha, "A unified convex surrogate for the Schatten-p norm," in *Proc. 31th AAAI Conf. Artificial Intelligence*, 2017, pp. 926-932.

[7] B. L. Yi, X. Shen, H. Liu, Z. L. Zhang, W. Zhang, S. Liu, and N. X. Xiong, "Deep matrix factorization with implicit feedback embedding for recommendation system," *IEEE Trans. on Industrial Informatics*, 2019, https://DOI: 10.1109/TII.2019.2893714.

[8] C. Peng, Z. Kang, M. Yang, and Q. Cheng, "Feature selection embedded subspace clustering," *IEEE Signal Process. Lett.*, vol. 23, no. 7, pp. 1018-1022, 2016.

[9] J. W. Zheng, C. Lu, H. C. Yu, S. Y. Chen, and W. L. Wang, "Iterative re-constrained low-rank representation via weighted nonconvex regularizer," *IEEE Access*, vol. 6, no. 1, pp. 51693-51707, 2018.

[10] M. Yin, J. B. Gao, and Z. C. Lin, "Laplacian regularized low-rank representation and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 504-514, 2016.

[11] X. Y. Xie, X. L. Guo, G. C. Liu, and J. Wang, "Implicit block diagonal low-rank representation," *IEEE Trans. on Image Process.*, vol. 27, no. 1, pp. 477-489, 2018.

[12] C. Y. Lu, J. H. Tang, S. C. Yan, and Z. C. Lin, "Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm," *IEEE Trans. on Image Process.*, vol. 25, no. 2, 829-839, 2016.

[13] I. Selesnick, "Sparse regularization via convex analysis," *IEEE Trans. Signal Process.*, vol. 65, no. 17, pp. 4481-4494, 2017.

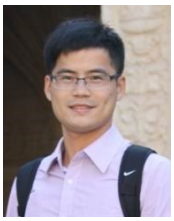
[14] I. Selesnick, M. Farshchian, "Sparse signal approximation via non-separable regularization," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2561-2575, 2017.

[15] A. Lanza, S. Morigi, I. Selesnick, and F. Sgallari, "Nonconvex non-smooth optimization via convex-nonconvex majorization-minimization," *Numerische Mathematik*, vol. 136, no. 2, pp. 343-381, 2017.

[16] S. Gu, Q. Xie, D. Meng, W. M. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vis.*, vol. 121, no. 2, pp. 183-208, 2017.
[17] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014.
[18] T. Y. Geng, G. L. Sun, Y. Xu, and J. F. He, "Truncated nuclear norm regularization based group sparse representation for image restoration," *SIAM J. Imageing Sciences*, vol. 11, no. 3, pp. 1878-1897, 2018.
[19] F. Cao, J. Chen, H. Ye, J. Zhao, and Z. Zhou, "Recovering low-rank and sparse matrix based on the truncated nuclear norm," *Neural Netw.*, vol. 85, pp. 10-20, 2017.
[20] Q. Lin, and L. Xiao, "An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization," *Computational Optimization and Applications*, vol. 60, no. 3, pp. 633-674, 2015.
[21] T. H. Oh, Y. W. Tai, J. C. Bazin, H. W. Kim, and I. S. Kweon, "Partial sum minimization of singular values in robust pca: algorithm and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 744-758, 2016.
[22] Q. Liu, Z. H. Lai, Z. W. Zhou, F. J. Kuang, and Z. Jin, "A truncated nuclear norm regularization method based on weighted residual error for matrix completion," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 316-330, 2016.
[23] T. Liu, T. Pong, and A. Takeda, "A successive difference-of-convex approximation method for a class of nonconvex nonsmooth optimization problems," *arXiv:1710.05778*, 2017.
[24] Q. Li, Y. Zhou, Y. Liang, and P. Varshney, "Convergence analysis of proximal gradient with momentum for nonconvex optimization," in *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, PMLR 70, 2017.
[25] B. Wen, X. Chen, and T. Pong, "Linear convergence of proximal gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 124-145, 2017.
[26] B. Wen, X. Chen, and T. Pong, "A proximal difference-of-convex algorithm with extrapolation," *Comput. Optim. Appl.*, vol. 69, no. 2, pp. 297-324, 2017.
[27] T. Oh, Y. Matsushita, Y. Tai, and I. Kweon, "Fast randomized singular value thresholding for low-rank optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 376-391, 2018
[28] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217-288, 2011.
[29] Q. Yao, J. Kwok, and W. Zhong, "Fast low-rank matrix learning with nonconvex regularization," in *Proc. Int. Conf. Data Mining*, pp. 539-548, 2015.
[30] L. Clopton, E. Mavroudi, M. Tsakiris, H. Ali, and R. Vidal, "Temporal subspace clustering for unsupervised action segmentation," *CSMR REU*, pp. 1-7, 2017.
[31] S. Li, K. Li, and Y. Fu, "Temporal subspace clustering for human motion segmentation," in *Proc. IEEE Conf. Comput. Vis.*, Satiago, Chile, 2015, pp. 4453-4461.

Jianwei Zheng (PhD(2010), BSc(2005)) is an associate professor in School of Computer Science and Engineering, Zhejiang University of Technology. He received his PhD in Control Theory and Control Engineering in 2010 from Zhejiang University of Technology, China. He has published more than 40 academic articles in reputable journals and conferences, including IEEE TIP, IEEE TNNLS, Neurocomputing, Visual Computer, Applied Intelligence, PCM, CGI, etc.

Mengjie Qin received the M.S. degree at School of Computer Science and Engineering, Zhejiang University of Technology, China in 2019. She is currently pursuing the Ph.D. degree in Zhejiang University of Technology. Her research interests include image and video enhancement, pattern recognition and machine learning.



Xiaolong Zhou (M15) received the Ph.D. degree in mechanical and biomedical engineering from the City University of Hong Kong, Hong Kong, in 2013. He is currently an associate professor with the College of Electrical and Information Engineering, Quzhou University, Quzhou, China. He was a post-doctoral research fellow with the School of Computing of University of Portsmouth, Portsmouth, UK, from 2015 to 2016. His research interests include visual tracking, gaze estimation, 3D reconstruction, and their applications in various fields. He has authored over 70 papers in peer-reviewed journals and conferences.

Jiafa Mao received the Ph.D. degree in pattern recognition from East China University of Science and Technology, China, in 2009. Since then, he has worked as a Post-doc Researcher at Beijing University of Posts and Telecommunications, China. In July 2011, he joined Zhejiang University of Technology, China, where he is currently a Professor in the School of Computer Science & Technology. His research interests include computer vision,pattern recognition, and information hiding.

Hongchuan Yu (Ph.D. (2000), M.Sc. (1996), B.Sc. (1990)) is a Senior Lecturer of computer graphics in National Centre for Computer Animation, Bournemouth University. He received his Ph.D. in Computer Vision, Inst. of Intelligent Machine, Chinese Academy of Sciences, in 2000. After that, he worked as research fellow at Tsinghua University; Nanyang Technological University (Singapore) and The University of Western Australia (Perth). He has published more than 70 academic articles in reputable journals and conferences, and regularly served as PC members/referees for international journals and conferences, including IEEE TPAMI, IEEE TIP, IEEE TVCG, IVC, PR, CVIU, PRL, CAD, CGI, etc.