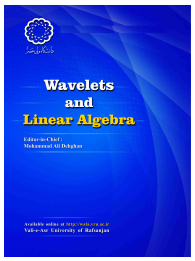
Wavelets and Linear Algebra 11(2) (2024) 32-51

Vali-e-Asr University of Rafsanjan

Wavelets and Linear Algebra

http://wala.vru.ac.ir



Accelerated algorithms for SiLRTC algorithm by fast tri-factorization method and total variation regularization

Rasool Ebrahimi[a], Ali Tavakoli*[a],

[a]*Department of Applied Mathematics, University of Mazandaran, Babolsar, Iran..*

Article Info

Article history:
Received 27 November 2023
Accepted 25 January 2024
Available online 16 November 2024

Keywords:
Image processing
Low rank matrix completion
Low rank tensor completion
Fast tri-factorization method

2000 MSC:
15A18, 15A69, 74B99

Abstract

Tensor completion is one of the efficient methods for restoring data such that minimizing the rank of the tensor leads to an appropriate solution. However, it gives a non-convex objective function, which generates an NP-hard problem. To overcome this problem, instead of using the rank function, the trace norm is applied. To solve this problem, Simple Low Rank Tensor Completion (SiLRTC) can be used. In the methods based on trace norm, the Singular Value Decomposition (SVD) is used, which increases computational complexity of these methods with increasing dimensions. In order to reduce the computational complexity of SVD, the approximate SVD can be utilized. In this paper, to accelerate the convergence speed of SiLRTC Algorithm, the new combined method FTF-SiLRTC is presented. On the other hand, the images recovered using the mentioned algorithms are generally accompanied by horizontal and vertical noise lines and have low accuracy. To solve this difficulty, the total variation (TV) regularization is added to the problem and the FTF-SiLRTC-TV Algorithm is introduced to solve it with higher accuracy.

https://doi.org/10.22072/wala.2024.2016583.1439

1. Notation and Preliminaries

In this section, we introduce the notations required for the subsequent sections. We follow [12] to define all notations. We use lowercase letters to denote scalars, e.g., x, capital letters to denote matrices, e.g., $X \in \mathfrak{R}^{I_1 \times I_2}$, and boldface Euler script letters to denote higher-order tensors, e.g., $\mathbb{X} \in \mathfrak{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The Frobenius norm of a tensor $\mathbb{X}$ is defined as $\|\mathbb{X}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_N} x_{i_1,i_2,\cdots,i_n}^2}$. The nuclear norm (NN) of a matrix X is defined as the sum of all singular values of X, denoted by $\|X\|_* = \sum_i \sigma_i(X)$, where $\sigma_i(X)$ is the i-th largest singular value of X. The unfold operation along the $k-mode$ on a tensor $\mathbb{X}$ is defined as $unfold_k(\mathbb{X}) = X_{(k)} \in \mathfrak{R}^{I_k \times (I_1 \ldots I_{k-1} I_{k+1} \ldots I_n)}$. The opposite operation fold is defined as $fold_k(X_{(k)}) = \mathbb{X}$. Tucker rank or multilinear rank which is introduced by Tucker is defined as $rank_{tc}(\mathbb{X}) = (rank(X_{(1)}), rank(X_{(2)}), \ldots, rank(X_{(n)}))$.

Definition 1.1. Let $U\Sigma V^T$ be the SVD of A. The thresholding operator D_τ is defined as:

$$D_\tau(A) = UD_\tau(\Sigma)V^T, \qquad D_\tau(\Sigma) = diag(max(\{\sigma_i - \tau\}, 0)).$$

Definition 1.2. The $shrinkage_\alpha()$ operator is the elementwise shrinkage-thresholding operator of a matrix, defined by

$$[shrinkage_\alpha(A)]_{i,j} = [A]_{i,j} - \min(\alpha, |[A]_{i,j}|).\frac{[A]_{i,j}}{|[A]_{i,j}|}.$$

Definition 1.3. Let Ω be the index set of the tensor $\mathbb{A}$ due to the observed data. The operator P_Ω is defined as below:

$$P_\Omega(.) : \mathfrak{R}^{I_1 \times I_2 \times \cdots \times I_n} \longrightarrow \mathfrak{R}^{I_1 \times I_2 \times \cdots \times I_n}$$

$$P_\Omega(\mathbb{X}_{i_1 i_2 \ldots i_n}) = \begin{cases} \mathbb{X}_{i_1 i_2 \ldots i_n} & (i_1, i_2, \ldots, i_n) \in \Omega \\ 0 & otherwise \end{cases}$$

Let $X = QR$ where Q and R are given by QR factorization. In this paper, $QR(X) = Q$. Most of notations are given in Table 1.

2. Introduction

It is well known that every day a huge amount of data is generated in a variety of fields such as data mining, pattern detection, machine learning and image processing. Therefore, handling big data is an interesting field for many researchers. However, working with big data usually presents some difficulties such as incomplete or lost data. Reconstructing this information is usually very costly and sometimes impossible, making the method of recovery very important. In the meantime, matrix and tensor completion methods are particularly important [1, 3, 5]. Meanwhile, the rank of a matrix (tensor) is a powerful tool to finding the general information of a matrix (tensor) when some of its entries are missing [6, 15]. For this reason, we examine the low-rank matrix (tensor) completion problem. In the sequel, we will briefly explain the work that has been done on low rank matrix and tensor completion and express the corresponding algorithms.

*Corresponding author
Email addresses: r.ebrahimi101@umail.umz.ac.ir (Rasool Ebrahimi), a.tavakoli@umz.ac.ir (Ali Tavakoli*)

Table 1: Notation

Notation	Type
$\mathbb{X}$	A tensor in the space $\Re^{I_1 \times I_2 \times \cdots \times I_n}$
X	A matrix in the space $\Re^{m \times n}$
X^T	Transpose of the matrix X
X_{ij}	The entry (i, j) of the matrix X
$\Sigma(X)$	The diagonal matrix given by SVD of X
$\sigma_i(X)$	The $i-th$ largest singular value of X
$\|X\|_*$	The trace norm of the matrix X
$\|X\|_F$	The Frobenius norm of the matrix X
$shrinkage_\alpha$	The shrinkage-thresholding operator
D_τ	The thresholding operator
Ω	The set of observed data
$QR(X)$	The orthogonal matrix Q of QR factorization

2.1. Matrix completion

We begin with the well-known optimization problem for matrix completion (MC):

$$\begin{aligned} \min_X \quad & rank(X), \\ \text{s.t.} \quad & P_\Omega(X) = P_\Omega(A), \end{aligned} \tag{2.1}$$

where $X, A \in \Re^{m \times n}$, and the elements of A in the set Ω are given while the remaining elements are missing. We aim to use a low-rank matrix X to approximate the missing elements. The optimization Problem (2.1) is non-convex, since the objective function $rank(X)$ is non-convex. One common approach is to replace $rank(.)$ by the trace norm $\|.\|_*$ [7]. A main advantage of the trace norm is that it is the convex hull for the rank of matrices. This leads to the following convex optimization problem for the MC [2, 13, 14]:

$$\begin{aligned} \min_X \quad & \|X\|_*, \\ \text{s.t.} \quad & P_\Omega(X) = P_\Omega(A). \end{aligned} \tag{2.2}$$

Therefore, to find the answer for the MC problem, we can solve Problem (2.2) instead of Problem (2.1). The solution to this problem using the iterative truncated SVD is given in Algorithm 1 [20].

In Algorithm 1, $\tau > 0$ is the threshold for the singular value, δ is the step length, and Y^0 is the initial value. Due to the use of SVD decomposition, this algorithm is associated with challenges such as being time-consuming or the inability of SVD decomposition for large matrices [20]. To overcome this problem, Liu et al. used the approximate SVD for MC to increase the speed of SVD and make it easier to compute the SVD of large matrices in [13], as presented in Algorithm 2. In this method, they have used the factorization $X = LMR$, in which:

- $M \in \Re^{r \times r}$,

Algorithm 1: Algorithm for matrix completion (MC)

Input : An incomplete data matrix $Y^0 \in \mathfrak{R}^{m\times n}$, the observation index set Ω, step length δ and thresholding value τ

Output: Completed data matrix X^{k+1}

while *not convergence* **do**

  $X^{k+1} = D_\tau(Y^k)$

  $Y^{k+1} = Y^k + \delta P_\Omega(A - X^{k+1})$

end

- $L \in \mathfrak{R}^{m\times r}, L^TL = I,$
- $R \in \mathfrak{R}^{r\times n}, RR^T = I,$
- $r \ll \min\{m, n\}.$

In addition, we have [13]:

$$\|X\|_* = \|LMR\|_* = \|M\|_*.$$

Therefore, by applying $X = LMR$, the Problem (2.2) can be expressed as follows:

$$\begin{aligned} \min_{L,M,R} \quad & \|M\|_*, \\ \text{s.t.} \quad & P_\Omega(LMR - A) = 0, \\ & L^TL = I, \\ & RR^T = I. \end{aligned} \tag{2.3}$$

Problem (2.3) can be solved by using the Alternating Direction Method of Multipliers (ADMM). To do this, an auxiliary variable can be added, which allows the Problem (2.3) to be equivalently rewritten as follows:

$$\begin{aligned} \min_{L,M,R,Z} \quad & \|M\|_*, \\ \text{s.t.} \quad & P_\Omega(Z - A) = 0, \\ & Z = LMR, \\ & L^TL = I, \\ & RR^T = I. \end{aligned} \tag{2.4}$$

Then, the partial augmented Lagrangian function of Problem (2.4) is given by:

$$\begin{aligned} \mathfrak{L}_\mu(L, M, R, Z, Y, \mu) = \min_{L,M,R,Z,Y,\mu} \quad & \|M\|_* + \tfrac{\mu}{2}\|Z - LMR + \frac{\Lambda^k}{\mu}\|_F^2, \\ \text{s.t.} \quad & P_\Omega(Z) = P_\Omega(A), \\ & L^TL = I, \\ & RR^T = I. \end{aligned} \tag{2.5}$$

where Λ is the Lagrange coefficient and μ is the penalty parameter, the iterative method to solve the Problem (2.5) can be given by [13]:

Algorithm 2: Fast Tri-Factorization Matrix Completion (FTF-MC)

Input : An incomplete data matrix $Z^0 \in \Re^{m\times n}$, the observation index set Ω, the size r of matrix $M \in \Re^{r\times r}$ and thresholding value μ

Output: Completed data matrix Z^{k+1}

while *not convergence* **do**

$M^{k+1} = D_{\mu^{-1}}\Big((L^k)^T(Z^k + Y^k/\mu)(R^k)^T\Big)$

$Z^{k+1} = L^k M^{k+1} R^k - Y^k/\mu + P_\Omega(A - L^k M^k R^k + Y^k/\mu)$

$L^{k+1} = QR\Big((Z^k + Y^k/\mu)(R^k)^T\Big)$

$(R^{k+1})^T = QR\Big((Z^k + Y^k/\mu)^T L^{k+1}\Big)$

$\Lambda^{k+1} = P_\Omega\Big(\Lambda^k + \mu(Z^{k+1} - L^{k+1}M^{k+1}R^{k+1})\Big)$

$Y^{k+1} = Y^k + \delta P_\Omega(Z^0 - Z^{k+1})$

end

2.2. Tensor completion

The tensor generalizes the concept of the matrix. We extend the completion algorithm for the matrix mode to higher-order tensors by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbb{X}} \quad & rank_{tc}(\mathbb{X}), \\ \text{s.t.} \quad & P_\Omega(\mathbb{X}) = P_\Omega(\mathbb{T}). \end{aligned} \tag{2.6}$$

According to the definition of Tucker rank and using the tensor matricization along each of its states, the above problem can be equivalently rewritten as:

$$\begin{aligned} \min_{X_{(i)}} \quad & rank(X_{(i)}), \\ \text{s.t.} \quad & P_\Omega(X_{(i)}) = P_\Omega(T_{(i)}), \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.7}$$

One can replace the non-convex function $rank(.)$ in (2.7) with a convex objective function $\|.\|_*$ as follows:

$$\begin{aligned} \min_{X_{(i)}} \quad & \|X_{(i)}\|_*, \\ \text{s.t.} \quad & P_\Omega(X_{(i)}) = P_\Omega(T_{(i)}), \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.8}$$

To solve Problem (2.8), Liu et al.[12] proposed Simple Low-Rank Tensor Completion (SiLRTC) Algorithm.

In Alg. 3, $\tau > 0$ is the threshold of the singular value and $\mathbb{Y}_i^0$ is the initial value.

In this paper, by using the approximate SVD of matrices for tensor surface, we present a method to accelerate the SiLRTC Algorithm. Additionally, to enhance the accuracy of the mentioned algorithm, TV regularization was incorporated into the problem, leading to the introduction of a new algorithm.

Algorithm 3: SiLRTC Algorithm

Input : An incomplete data tensor $\mathbb{X}^0 \in \Re^{I_1 \times I_2 \times \cdots \times I_n}$, the observation index set Ω and thresholding value τ

Output: Completed data tensor $\mathbb{X}^{k+1}$

while *not convergence* **do**

$X_{(i)}^{k+1} = D_\tau(unfold_i(\mathbb{X}^k))$

$\mathbb{X}^{k+1} = \frac{1}{n}\sum_{i=1}^{n} fold_i(X_{(i)}^{k+1})$

$\mathbb{X}_\Omega^{k+1} = \mathbb{X}_\Omega^0$

end

3. Main Results

It is well known that the unfolded matrices along each tensor mode are generally large scale and the SVD decomposition for these matrices is time-consuming or sometimes unstable. In order to accelerate the convergence speed of the SiLRTC algorithm, we use the approximate SVD. In fact, we combine the SiLRTC Algorithm with the approximate SVD algorithm to derive the new algorithm that converges faster than the SiLRTC Algorithm. In the following, the computational complexity of the stated algorithms is examined in each iteration. At the end, we evaluate the convergence of the stated algorithms.

3.1. Fast tri-factorization simple low rank tensor completion

To increase the speed of convergence in Algorithm 3, we use the approximate SVD. The Problem (2.6) can be rewritten as follows:

$$\begin{aligned} \min_{L_{(i)},M_{(i)},R_{(i)},Z_{(i)}} \quad & \|M_{(i)}\|_*, \\ \text{s.t.} \quad & P_\Omega(Z_{(i)} - T_{(i)}) = 0, \\ & Z_{(i)} = L_{(i)}M_{(i)}R_{(i)}, \\ & L_{(i)}^T L_{(i)} = I, \\ & R_{(i)}R_{(i)}^T = I, \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{3.1}$$

Then, the partial augmented Lagrangian function of Problem (3.1) is given as follows:

$$\begin{aligned} \mathfrak{L}_\mu(L_{(i)}, M_{(i)}, R_{(i)}, Z_{(i)}, \Lambda_{(i)}) \quad & = \|M_{(i)}\|_* + \tfrac{\mu}{2}\|Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\|_F^2, \\ \text{s.t.} \quad & P_\Omega(Z_{(i)} - T_{(i)}) = 0, \\ & L^T L = I, \\ & RR^T = I, \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{3.2}$$

Since simultaneously minimizing all terms of $\mathfrak{L}_\mu(L_{(i)}, M_{(i)}, R_{(i)}, Z_{(i)}, \Lambda_{(i)})$ is hard, we apply an alternating direction method of multipliers (ADMM) to solve Problem (3.2). This is computed by successively minimizing the Lagrange function $\mathfrak{L}_\mu(L_{(i)}, M_{(i)}, R_{(i)}, Z_{(i)}, \Lambda_{(i)})$ one at a time while fixing the others at their latest values.

Updating $\{M_{(i)}\}_{i=1}^{N}$: In the SiLRTC Algorithm, to find $M_{(i)}$, the optimization problem can be formulated as follows:

$$\begin{aligned} M_{(i)} &= \underset{M_{(i)}}{argmin} \quad \mathfrak{L}_{\mu}(L_{(i)}, M_{(i)}, R_{(i)}, Z_{(i)}, \Lambda_{(i)}), \\ &= \underset{M_{(i)}}{argmin} \quad \|M_{(i)}\|_{*} + \tfrac{\mu}{2}\|Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\|_{F}^{2}. \end{aligned} \tag{3.3}$$

where μ is the penalty parameter. Then, based on SiLRTC Algorithm the optimal solution $M_{(i)}$ of Problem (3.3) is [13]:

$$M_{(i)} = D_{\mu^{-1}}\Big((L_{(i)})^{T}(Z_{(i)} + \frac{\Lambda_{(i)}}{\mu})(R_{(i)})^{T}\Big). \tag{3.4}$$

Updating $\{Z_{(i)}\}_{i=1}^{N}$: The optimal solution $Z_{(i)}$ of Problem (3.3) can be computed as follows:

$$\begin{aligned} Z_{(i)} &= \underset{Z_{(i)}}{argmin} \quad \mathfrak{L}_{\mu}(L_{(i)}, M_{(i)}, R_{(i)}, Z_{(i)}, \Lambda_{(i)}), \qquad P_{\Omega}(Z_{(i)} - X_{(i)}) = 0, \\ &= \underset{Z_{(i)}}{argmin} \quad \tfrac{\mu}{2}\|Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\|_{F}^{2}, \quad P_{\Omega}(Z_{(i)} - X_{(i)}) = 0. \end{aligned} \tag{3.5}$$

In addition, by deriving the KKT conditions for Problem (3.5) simply, the optimal solution $Z_{(i)}$ satisfies the following equations:

$$P_{\Omega}(Z_{(i)} - X_{(i)}) = 0, \quad P_{\Omega^{c}}(Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}) = 0, \tag{3.6}$$

where Ω^{C} is the complement of Ω, i.e., the set of indeces of the unobserved entries, according to (3.6), the solution $Z_{(i)}$ of Problem (3.5) can be explicitly computed as follows:

$$Z_{(i)} = L_{(i)}M_{(i)}R_{(i)} - \frac{\Lambda_{(i)}}{\mu} + P_{\Omega}\Big(X_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\Big). \tag{3.7}$$

Updating $\{L_{(i)}\}_{i=1}^{N}$ **and** $\{R_{(i)}\}_{i=1}^{N}$: To find $L_{(i)}$ and $R_{(i)}$, the following subproblems are to be considered:

$$\begin{cases} L_{(i)} = \underset{L_{(i)}}{argmin} \quad \tfrac{\mu}{2}\|Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\|_{F}^{2}, \quad s.t. \quad L_{(i)}^{T}L_{(i)} = I, \\ R_{(i)} = \underset{R_{(i)}}{argmin} \quad \tfrac{\mu}{2}\|Z_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\mu}\|_{F}^{2}, \quad s.t. \quad R_{(i)}R_{(i)}^{T} = I. \end{cases} \tag{3.8}$$

Both subproblems in (3.8) are least-squares problems with orthogonality constraints on columns and rows, respectively. Following [13], $L_{(i)}$ and $R_{(i)}$ are updated as follows:

$$\begin{cases} L_{(i)} = QR\Big((Z_{(i)} + \frac{\Lambda_{(i)}}{\mu})R_{(i)}^{T}\Big), \\ R_{(i)} = QR\Big(L_{(i)}^{T}(Z_{(i)} + \frac{\Lambda_{(i)}}{\mu})\Big). \end{cases} \tag{3.9}$$

Updating $\mathbb{X}$: Finally, updating the variable $\mathbb{X}$ is done as follows;

$$\mathbb{X} = \frac{1}{n}\sum_{i=1}^{n} fold_i(Z_{(i)}), \tag{3.10}$$

such that

$$P_\Omega(\mathbb{X}) = P_\Omega(\mathbb{T}). \tag{3.11}$$

Consequently, by using the ADMM method and the preceding updates for solving Problem (2.3), we propose a new method that is explained in Algorithm 4 (FTF-SiLRTC).

Algorithm 4: FTF-SiLRTC Algorithm

Input : An incomplete data tensor $\mathbb{X}^0 \in \Re^{I_1\times I_2\times\cdots\times I_n}$, the observation index set Ω, the size r of the matrix $M \in \Re^{r\times r}$ and thresholding value μ

Output: Completed data tensor $\mathbb{X}^{k+1}$

while *not convergence* **do**

  for $i = 1, 2, \ldots, n$ **do**

    $M_{(i)}^{k+1} = D_{\mu^{-1}}\Big((L_{(i)}^k)^T(unfold_i(\mathbb{X}^k) + \frac{\Lambda_{(i)}^k}{\mu})(R_{(i)}^k)^T\Big)$

    $Z_{(i)}^{k+1} = L_{(i)}^k M_{(i)}^{k+1} R_{(i)}^k - \frac{\Lambda_{(i)}^k}{\mu} + P_\Omega(\mathbb{X}_{(i)}^0 - L_{(i)}^k M_{(i)}^k R_{(i)}^k + \frac{\Lambda_{(i)}^k}{\mu})$

    $L_{(i)}^{k+1} = QR\Big((Z_{(i)}^k + \frac{\Lambda_{(i)}^k}{\mu})(R_{(i)}^k)^T\Big)$

    $(R_{(i)}^{k+1})^T = QR\Big((L_{(i)}^{k+1})^T(Z_{(i)}^{k+1} + \frac{\Lambda_{(i)}^k}{\mu})\Big)$

    $\Lambda_{(i)}^{k+1} = P_\Omega\Big(\Lambda_{(i)}^k + \mu(Z_{(i)}^{k+1} - L_{(i)}^{k+1} M_{(i)}^{k+1} R_{(i)}^{k+1})\Big)$

  end

  $\mathbb{X}^{k+1} = \frac{1}{n}\sum_{i=1}^{n} fold_i(Z_{(i)}^{k+1})$

  $\mathbb{X}_\Omega^{k+1} = \mathbb{X}_\Omega^0$

end

3.2. Fast tri factorization simple low rank tensor completion via TV regularization

Although the FTF-SiLRTC Algorithm is fast and convergent, it has low accuracy and the recovered images are accompanied by vertical or horizontal lines. In order to solve this problem, we add the total variation regularization to it. Then, problem (3.1) reformulated as follows:

$$\begin{aligned} \min_{\{Z_{(i)}\}_{i=1}^N} \quad & \lambda\sum_{i=1}^{N}\beta_i\left|F_{(i)}Z_{(i)}\right| + \tfrac{1}{N}\sum_{i=1}^{N}\|Z_{(i)}\|_*, \\ \text{s.t.} \quad & P_\Omega(Z_{(i)}) = P_\Omega(T_{(i)}), \end{aligned} \tag{3.12}$$

by adding the auxiliary variables $\mathbb{X}, \mathbb{Y}$ and $\{Q_{(i)}\}_{i=1}^{N}$, we have:

$$\begin{aligned} \min_{\{Q_{(i)},X_{(i)},Y_{(i)},L_{(i)},M_{(i)},R_{(i)}\}_{i=1}^{N}} \quad & \lambda \sum_{i=1}^{N} \beta_i \left|Q_{(i)}\right| + \tfrac{1}{N} \sum_{i=1}^{N} \|M_{(i)}\|_*, \\ \text{s.t.} \quad & Q_{(i)} = F_{(i)}X_{(i)}, X_{(i)} = Z_{(i)}, Y_{(i)} = Z_{(i)}, \\ & Y_{(i)} = L_{(i)}M_{(i)}R_{(i)}, L_{(i)}^T L_{(i)} = I, R_{(i)}R_{(i)}^T = I, \\ & P_\Omega(Z_{(i)}) = P_\Omega(T_{(i)}), \end{aligned} \tag{3.13}$$

where $F_{(i)}$ is a (J_{n-1})-by-J_n matrix, such that $[F_{(i)}]_{j,j} = 1$, $[F_{(i)}]_{j,j+1} = -1$ and the other entries are zeros. Moreover, we define the operator $|.|$ by $|A| = \sum_i \sum_j \left|[A]_{i,j}\right|$ for developing TV constraint. λ is a tunable parameter; $\beta_1, \cdots, \beta_N$ is 0 or 1, which indicates whether we have a smooth and piecewise prior on the $n-$th mode of recovered tensor. The values of $\beta_1, \beta_2, \cdots, \beta_N$ depend on the domain. The augmented Lagrangian formulation of (3.13) can be given by:

$$\begin{aligned} \mathfrak{L} \quad = & \sum_{i=1}^{N} \beta_i \Big(\lambda \left|Q_{(i)}\right| + \frac{\rho_1}{2}\|Q_{(i)} - F_{(i)}X_{(i)} + \frac{\Lambda_{(i)}}{\rho_1}\|_F^2\Big), \\ & + \sum_{i=1}^{N} \beta_i \Big(\frac{\rho_2}{2}\|Z_{(i)} - X_{(i)} + \frac{\Gamma_{(i)}}{\rho_2}\|_F^2\Big), \\ & + \sum_{i=1}^{N} \Big(\frac{1}{N}\|M_{(i)}\|_* + \frac{\rho_3}{2}\|Y_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Phi_{(i)}}{\rho_3}\|_F^2\Big), \\ & + \sum_{i=1}^{N} \Big(\frac{\rho_4}{2}\|Z_{(i)} - Y_{(i)} + \frac{\Psi_{(i)}}{\rho_4}\|_F^2\Big) \\ \text{s.t.} \quad & P_\Omega(Z_{(i)} - T_{(i)}) = 0, \\ & L_{(i)}^T L_{(i)} = I, \\ & R_{(i)}R_{(i)}^T = I. \end{aligned} \tag{3.14}$$

where $\Lambda_{(i)}, \Gamma_{(i)}$ and $\Psi_{(i)}$ for $i = 1, \cdots, N$ are the Lagrange coefficients and also ρ_1, ρ_2, ρ_3 and ρ_4 are the penalty parameters. Since simultaneously minimizing all terms of $\mathfrak{L}$ is hard, we apply the ADMM to solve Problem (3.2). This is computed by successively minimizing the Lagrange function $\mathfrak{L}$ one at a time while fixing the others at their latest values.

Updating $\{Q_{(i)}\}_{i=1}^{N}$: We update $\{Q_{(i)}\}_{i=1}^{N}$ with fixing the other variables based on the following optimization problem:

$$\min_{\{Q_{(i)}\}_{i=1}^{N}} \sum_{i=1}^{N} \beta_n \Big(\lambda \left|Q_{(i)}\right| + \frac{\rho_1}{2}\|Q_{(i)} - F_{(i)}X_{(i)} + \frac{\Lambda_{(i)}}{\rho_1}\|_F^2\Big). \tag{3.15}$$

Because of the independence of $Q_1, Q_2, \cdots, Q_N$ in the optimization problem, they are updated as follows:

$$Q_{(i)} = \beta_i shrinkage_{\frac{\lambda}{\rho_1}}\Big(F_{(i)}X_{(i)} - \frac{\Lambda_{(i)}}{\rho_1}\Big). \tag{3.16}$$

Updating $\{X_{(i)}\}_{i=1}^N$: We update $\{X_{(i)}\}_{i=1}^N$ with fixing the other variables based on the following optimization problem:

$$\begin{aligned}\min_{\{X_{(i)}\}_{i=1}^N} \quad & \sum_{i=1}^N \beta_i \frac{\rho_1}{2}\|Q_{(i)} - F_{(i)}X_{(i)} + \frac{\Lambda_{(i)}}{\rho_1}\|_F^2 \\ & + \sum_{i=1}^N \beta_i\Big(\frac{\rho_2}{2}\|Z_{(i)} - X_{(i)} + \frac{\Gamma_{(i)}}{\rho_2}\|_F^2\Big).\end{aligned} \tag{3.17}$$

The optimizing solution of $X_{(i)}$ is given as follows:

$$X_{(i)} = \beta_i(\rho_1 F_{(i)}F_{(i)}^T + \rho_2 I)^{-1}(F_{(i)}^T\Lambda_{(i)} + \rho_1 F_{(i)}^T Q_{(i)} + \rho_2 Z_{(i)} - \Gamma_{(i)}), \tag{3.18}$$

where I stands for the identify matrix.

Updating $\{M_{(i)}\}_{i=1}^N$, $\{Y_{(i)}\}_{i=1}^N$, $\{L_{(i)}\}_{i=1}^N$ **and** $\{R_{(i)}\}_{i=1}^N$: Based on FTF-SiLRTC Algorithm updating $M_{(i)}, Y_{(i)}, L_{(i)}$ and $R_{(i)}$ are as follow:

$$\begin{aligned} M_{(i)} &= D_\tau\Big((L_{(i)})^T(Y_{(i)} + \frac{\Lambda_{(i)}}{\rho_3})(R_{(i)})^T\Big), \\ Z_{(i)} &= L_{(i)}M_{(i)}R_{(i)} - \frac{\Lambda_{(i)}}{\rho_3} + P_\Omega(T_{(i)} - L_{(i)}M_{(i)}R_{(i)} + \frac{\Lambda_{(i)}}{\rho_3}), \\ L_{(i)} &= QR\Big((Z_{(i)} + \frac{\Lambda_{(i)}}{\rho_3})(R_{(i)})^T\Big), \\ (R_{(i)})^T &= QR\Big((L_{(i)})^T(Z_{(i)} + \frac{\Lambda_{(i)}}{\rho_3})\Big). \end{aligned}$$

Updating $\{\Lambda_{(i)}\}_{i=1}^N$, $\{\Gamma_{(i)}\}_{i=1}^N$, $\{\Phi_{(i)}\}_{i=1}^N$ **and** $\{\Psi_{(i)}\}_{i=1}^N$: By following [11], updating $\{\Lambda_{(i)}\}_{i=1}^N$, $\{\Gamma_{(i)}\}_{i=1}^N$ and $\{\Psi_{(i)}\}_{i=1}^N$ are presented as follow:

$$\begin{aligned} \Lambda_{(i)} &= \Lambda_{(i)} + \rho_1(Q_{(i)} - F_{(i)}X_{(i)}), \\ \Gamma_{(i)} &= \Gamma_{(i)} + \rho_2(Z_{(i)} - X_{(i)}), \\ \Phi_{(i)} &= P_\Omega\Big(\Phi_{(i)} + \mu(Y_{(i)} - L_{(i)}M_{(i)}R_{(i)})\Big) \\ \Psi_{(i)} &= \Psi_{(i)} + \rho_4(Z_{(i)} - Y_{(i)}). \end{aligned}$$

Updating $\mathbb{Z}$: Finally, $\mathbb{Z}$ can be updated based on the following optimization problem:

$$\min_{\{Z_{(i)}\}_{i=1}^N} \sum_{i=1}^N \beta_i\Big(\frac{\rho_2}{2}\|X_{(i)} - Z_{(i)} + \frac{\Gamma_{(i)}}{\rho_2}\|_F^2\Big) + \sum_{i=1}^N \Big(\frac{\rho_4}{2}\|Y_{(i)} - Z_{(i)} + \frac{\Phi_{(i)}}{\rho_4}\|_F^2\Big). \tag{3.19}$$

Then, the update formulae of $\mathbb{Z}$ are computed as:

$$[\mathbb{Z}]_{\bar{\Omega}} = [\frac{\sum_{i=1}^N \Big(fold_i(\Gamma_{(i)} + \rho_2 X_{(i)}) + fold_i(\Psi_{(i)} + \rho_4 Y_{(i)})\Big)}{N\rho_2 + \sum_{i=1}^N \beta_i\rho_4}]_{\bar{\Omega}}, \tag{3.20}$$

such that

$$[\mathbb{Z}]_\Omega = [\mathbb{T}]_\Omega.$$

By these update formulas, we summarize the solution of Problem (3.12), named FTF-SiLRTC-TV, in Algorithm 5. It is an iterative algorithm under ADMM framework [11]. In lines 3 − 8 and

13, the auxiliary matrices $\{Q_{(i)}\}_{i=1}^{N}$, $\{X_{(i)}\}_{i=1}^{N}$, $\{M_{(i)}\}_{i=1}^{N}$, $\{Y_{(i)}\}_{i=1}^{N}$, $\{L_{(i)}\}_{i=1}^{N}$, $\{R_{(i)}\}_{i=1}^{N}$ and target output variable $\mathbb{Z}$ are updated according to our derivations. In lines 9 − 11, we renew the Lagrange multipliers $\{\Lambda_{(i)}\}_{i=1}^{N}$, $\{\Gamma_{(i)}\}_{i=1}^{N}$ and $\{\Psi_{(i)}\}_{i=1}^{N}$ as the standard ADMM framework.

Algorithm 5: FTF-SiLRTC-TV Algorithm

Input : An incomplete data tensor $\mathbb{Z}^0 \in \mathfrak{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the observation index set Ω, the size r of the matrix $M \in \mathfrak{R}^{r \times r}$, thresholding value ρ and tunable parameter λ

Output: Completed data tensor $\mathbb{Z}^{k+1}$

1 **while** *not convergence* **do**

2 **for** $i = 1, 2, \ldots, N$ **do**

3 $Q_{(i)}^{k+1} = \beta_i shrinkage_{\frac{\lambda}{\rho_1}}\Big(F_{(i)}X_{(i)}^{k} - \dfrac{\Lambda_{(i)}^{k}}{\rho_1}\Big)$

4 $X_{(i)}^{k+1} = \beta_i(\rho_1 F_{(i)}F_{(i)}^{T} + \rho_2 I)^{-1}(F_{(i)}^{T}\Lambda_{(i)}^{k} + \rho_1 F_{(i)}^{T}Q_{(i)}^{k+1} + \rho_2 Z_{(i)}^{k} - \Gamma_{(i)}^{k})$

5 $M_{(i)}^{k+1} = D_{\frac{1}{N\rho_3}}((L_{(i)}^{k})^{T} unfold_i(\mathbb{Z}^{k})(R_{(i)}^{k})^{T})$

6 $Y_{(i)}^{k+1} = L_{(i)}^{k}M_{(i)}^{k+1}R_{(i)}^{k} + P_{\Omega}\Big(Z_{(i)}^{0} - L_{(i)}^{k}M_{(i)}^{k}R_{(i)}^{k}\Big)$

7 $L_{(i)}^{k+1} = QR\Big((Y_{(i)}^{k})(R_{(i)}^{k})^{T}\Big)$

8 $(R_{(i)}^{k+1})^{T} = QR\Big((Y_{(i)}^{k})^{T}L_{(i)}^{k}\Big)$

9 $\Lambda_{(i)}^{k+1} = \Lambda_{(i)}^{k} + \rho_1(Q_{(i)}^{k+1} - F_{(i)}X_{(i)}^{k+1})$

10 $\Gamma_{(i)}^{k+1} = \Gamma_{(i)}^{k} + \rho_2(Z_{(i)}^{k} - X_{(i)}^{k+1})$

11 $\Phi_{(i)}^{k+1} = P_{\Omega}\Big(\Phi_{(i)}^{k} + \mu(Y_{(i)}^{k+1} - L_{(i)}^{k+1}M_{(i)}^{k+1}R_{(i)}^{k+1})\Big)$

12 $\Psi_{(i)}^{k+1} = \Psi_{(i)}^{k} + \rho_4(Z_{(i)}^{k} - Y_{(i)}^{k+1})$

13 **end**

14 $\mathbb{Z}^{k+1} = \dfrac{\sum_{i=1}^{N}(fold_i(\Gamma_{(i)}^{k+1} + \rho_2 X_{(i)}^{k+1}) + fold_i(\Psi_{(i)}^{k+1} + \rho_4 Y_{(i)}^{k+1}))}{N\rho_2 + \sum_{i=1}^{N}\beta_i\rho_4}$

15 **end**

3.3. Computational complexity

In this part, we discuss the time complexity of mentioned algorithms, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms.

The computational complexity of each operator for matrix $X \in \mathfrak{R}^{m \times n}$ is presented in Table 2 [8].

The computational complexity of SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms in each iteration for solving the tensor completion problem is presented in Table 3:

where $K_j = \prod_{\substack{i=1 \\ i \neq j}}^{n} I_i$.

Particularly, for image tensor $\mathbb{T} \in \mathfrak{R}^{m \times n \times 3}$, the computational complexity of the three algorithms is given in Table 4.

Table 2: Computational complexity of operators

Operation	Computational complexity	
$SVD(X)$	$4m^2n + 8mn^2 + 9n^3$	if $m \geq n$
$SVD(X)$	$4m^2n + 22n^3$	if $m \gg n$
$QR(X)$	$2mn^2 - \frac{2}{3}n^3$	
$D_\tau(\Sigma)$	$2n$	
$UD_\tau(\Sigma)V^T$	$2m^2n + n^2 - mn$	
$A \pm X, \quad A \in \mathfrak{R}^{m\times n}$	mn	
$c.X, \qquad c \in \mathfrak{R}$	mn	
$P_\Omega(X)$	mn	

Table 3: Computational complexity of algorithms

Algorithm	Computational complexity
SiLRTC	$\sum_{j=1}^{N}\Big((6K_j + 8I_j - 1)K_jI_j + I_j^2(9I_j + 1)\Big)$
FTF-SiLRTC	$\sum_{j=1}^{N}\Big(\frac{59}{3}r^3 + (6I_j + 4K_j - 1)r^2 + (6I_jK_j - 2I_j - K_j)r + 9I_jK_j\Big)$
FTF-SiLRTC-TV	$\sum_{j=1}^{N}\Big(\frac{59}{3}r^3 + (6I_j + 4K_j - 1)r^2 + (6I_jK_j - 2I_j - K_j)r$ $+4I_j^3 + 8I_j^2K_j + 6I_j^2 + 3I_jK_j + 2I_j - 10K_j\Big)$

In the following, we show that the following relation always holds for $r \leq \min\{m, n\}$:

$$n(\text{FTF} - \text{SiLRTC}) \leq \text{n(SiLRTC)}, \tag{3.21}$$

that means

$$\begin{aligned} &59r^3 + (4mn + 18(m + n) + 15)r^2 + (53mn - 5(m + n) - 6)r - 81mn \\ &\leq 18m^2n^2 + 9(m^3 + n^3) + 78(m^2n + mn^2) + m^2 + n^2 + 6mn + 594, \end{aligned}$$

Let $r = n$, then we have:

$$(68 + 4m)n^3 + 9n^2 - 6n \leq 18m^2n^2 + 9m^3 + 7m^2n + 78mn^2 + m^2 + 92mn + 594,$$

As a result, assuming that $m = n$

$$14m^4 + 26m^3 + 84m^2 + 6m + 594 \geq 0,$$

It is clear that the above relations are always established for $m \geq 1$.

Remark 3.1. By a simple calculation, it can be seen that for $r \leq \min\{m, n\}$, the inequalities (3.21) hold true.

Table 4: Computational complexity of algorithms

Algorithm	Computational complexity
SiLRTC	$18m^2n^2 + 9(m^3 + n^3) + 78(m^2n + mn^2) + m^2 + n^2 + 6mn + 594$
FTF-SiLRTC	$59r^3 + (4mn + 18(m + n) + 15)r^2 + (53mn - 5(m + n) - 6)r - 81mn$
FTF-SiLRTC-TV	$59r^3 + (4mn + 18(m + n) + 15)r^2 + (55mn - 5(m + n) - 6)r + 168$ $+4(m^3 + n^3) + 24(m^2n + mn^2) + 6(m^2 + n^2) + 89mn - 28(m + n)$

4. Numerical Results

In this section, we present some examples to show the efficiency of our proposed FTF-SiLRTC and FTF-SiLRTC-TV Algorithms. Moreover, a comparison between them is presented. In all of the examples, for corrupted images, 90 percent of the original images are randomly lost.
All codes have been run by MATLAB R2018a software on the computer 64-bit Intel(R) Core(TM) i5-5200U CPU.
Figure 1 shows a set of coloured images in which Figure 1(a) is the original image, Figure 1(b) is the corrupted image, Figure 1(c) is the reconstructed image by SiLRTC Algorithm and Figure 1(d) through Figure 1(i) are the reconstructed image by FTF-SiLRTC Algorithm with $r = 5, 10, \cdots, 35$, respectively. Also, the stop criterion is thought of as $\|\mathbb{X}^{k+1} - \mathbb{X}^k\|_F \leq .2$ and the PSNR and RMSE formulas used in the examples are defined by

$$\begin{aligned} PSNR(\mathbb{X}) &= 10\log_{10}\Big(\frac{255^2}{MSE(\mathbb{X})}\Big), \\ RMSE(\mathbb{X}) &= \sqrt{MSE(\mathbb{X})}, \end{aligned}$$

where $MSE(\mathbb{X}) = \frac{1}{3mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{3}(t_{ijk} - x_{ijk})^2$.

Tables 5, 6, and 7 show the CPU time, PSNR and RMSE for FTF-SiLRTC Algorithm with $r = 5, 10, \cdots, 35$, respectively.

Table 5: The CPU time of FTF-SiLRTC Algorithm with r =5, 10, 15, 20, 25, 30 and 35 on the images of Figure 1.

Image	Size	FTF-SiLRTC						
		$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	$194 \times 259 \times 3$	8	13	27	36	75	86	178
Dog	$213 \times 238 \times 3$	9	11	21	26	62	67	145
House	$256 \times 256 \times 3$	17	21	22	31	52	53	64
Parrot	$256 \times 384 \times 3$	5	6	15	18	27	33	37
F16	$512 \times 512 \times 3$	15	17	37	52	53	82	119
Squirrel	$553 \times 800 \times 3$	67	132	199	580	1251	1810	1531

Table 6: The PSNR of FTF-SiLRTC Algorithm with $r = 5, 10, 15, 20, 25, 30$ and 35 on the images of Figure 1.

Image	FTF-SiLRTC						
	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	18.70	21.71	23.78	26.11	26.17	26.64	28.21
Dog	24.04	29.06	30.76	32.16	32.66	33.95	34.45
House	22.20	24.48	27.24	28.61	30.65	30.88	32.25
Parrot	22.33	26.36	26.79	28.48	29.44	29.75	30.28
F16	21.69	23.12	25.10	25.44	26.90	27.04	27.28
Squirrel	21.63	23.05	23.74	24.31	24.19	24.79	24.68

Table 7: The RMSE of FTF-SiLRTC Algorithm with $r = 5, 10, 15, 20, 25, 30$ and 35 on the images of Figure 1.

Image	FTF-SiLRTC						
	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	29.60	20.95	16.49	12.61	12.53	11.87	9.91
Dog	16.02	8.98	7.39	6.29	5.93	5.12	4.83
House	19.80	15.22	11.08	9.46	7.49	7.28	6.23
Parrot	19.51	12.26	11.67	9.61	8.61	8.30	7.81
F16	20.99	17.81	14.19	13.64	11.52	11.33	11.04
Squirrel	21.12	17.95	16.57	15.52	14.70	15.75	14.92

As is seen in Tables 5, 6 and 7, choosing the appropriate r (the size of matrix M) has a significant effect on the result of the FTF-SiLRTC Algorithm. The CPU time and RMSE of FTF-SiLRTC Algorithm is significantly decreased and the PSNR of the images are almost increased by increasing the value of r.

Figure 2 shows a set of coloured images in which Figure 2(a) is the original image, Figure 2(b) is the corrupted image and Figure 2(c) through Figure 2(f) are the reconstructed image by FTF-SiLRTC-TV Algorithm with $r = 5, 15, 25$ and 35, respectively.

Tables 8, 9, and 10 show the CPU time, PSNR and RMSE for FTF-SiLRTC-TV Algorithm with $r = 5, 10, \cdots, 35$, respectively.

As is seen in Tables 8, 9 and 10, changing the r value does not have much effect on the recovery result and the FTF-SiLRTC-TV Algorithm works almost independently of the r value.

Figure 3 shows a set of coloured images in which Figure 3(a) is the original image, Figure 3(b) is the corrupted image and Figure 3(c) through Figure 3(f) are the reconstructed image by TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$, respectively.

Tables 11, 12, and 13 show the CPU time, PSNR and RMSE for TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$, respectively.

Table 8: The CPU time of FTF-SiLRTC-TV Algorithm with r =5, 10, 15, 20, 25, 30 and 35 on the images of Figure 2.

Image	FTF-SiLRTC-TV						
	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	9.24	8.56	8.90	8.68	6.63	8.49	8.52
Dog	7.51	10.83	10.78	7.62	9.00	8.37	7.32
House	9.70	9.41	8.77	8.61	9.06	9.18	9.63
Parrot	8.36	8.64	7.64	8.01	8.89	8.45	7.96
F16	6.28	6.16	6.01	6.44	6.33	6.29	6.29
Squirrel	170	180	185	187	195	192	199

Table 9: The PSNR of FTF-SiLRTC-TV Algorithm with r = 5, 10, 15, 20, 25, 30 and 35 on the images of Figure 2.

Image	FTF-SiLRTC-TV						
	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	27.09	28.54	27.56	27.71	29.88	28.54	29.60
Dog	32.64	30.31	30.81	32.35	32.62	32.28	33.76
House	28.43	30.14	30.25	30.23	31.34	31.81	31.92
Parrot	30.15	30.52	29.99	30.93	29.62	29.89	31.16
F16	26.54	25.67	26.81	26.37	26.03	26.86	26.51
Squirrel	26.23	26.99	26.56	26.52	25.85	26.27	26.43

As is seen in Tables 11, 12 and 13, the best results are related to the FTF-SiLRTC-TV Algorithm. FTF-SiLRTC-TV Algorithm has higher speed and accuracy compared to other algorithms.

5. Conclusion

In this paper, we presented the algorithms called FTF-SiLRTC and FTF-SiLRTC-TV to accelerate the SiLRTC Algorithm. By comparison of SiLRTC Algorithm (Alg. 3 and our proposed algorithms, it is shown that for image tensor in $\Re^{m\times n\times 3}$ the CPU time of SiLRTC Algorithm can be reduced by FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with choosing appropriate r. Also, we can see that in general, for FTF-SiLRTC Algorithm the bigger r, the more PSNR and the less RMSE. Therefore, it is concluded that increasing the value of r will increase the accuracy of FTF-SiLRTC Algorithm. On the other hand, FTF-SiLRTC-TV algorithm provides higher accuracy in less time than other algorithms. The efficiency of our proposed method will be increased if the number of dimensions and/or the size of dimensions grow up.

Table 10: The RMSE of FTF-SiLRTC-TV Algorithm with $r = 5$, 10, 15, 20, 25, 30 and 35 on the images of Figure 2.

Image	FTF-SiLRTC-TV						
	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
Peppers	11.27	9.54	10.68	10.50	8.18	9.54	8.45
Dog	5.95	7.78	7.34	6.15	5.97	6.20	5.23
House	9.67	7.93	7.83	7.85	6.91	6.55	6.46
Parrot	9.55	7.59	8.07	7.24	8.41	8.16	7.05
F16	12.01	13.28	11.65	12.26	12.73	11.58	12.06
Squirrel	12.45	11.41	11.98	12.04	13.01	12.39	12.18

Table 11: The CPU time of TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$ on the images of Figure 3.

Image	TRLRF	HaLRTC	SiLRTC	FTF-SiLRTC	FTF-SiLRTC-TV
Peppers	150	43	45	35	**9**
Dog	168	46	35	25	**11**
House	164	50	34	22	**9**
Parrot	157	46	33	15	**8**
F16	157	63	40	37	**6**
Squirrel	1817	1655	504	**199**	**186**

References

[1] A. J. Bengua, H. N. Phien, H. D. Tuan and M. N. Do. Efficient tensor completion for color image and video recovery: Low-rank tensor train. *IEEE Transactions on Image Processing*, 26 (2017), 2466-2479.

[2] J. F. Cai, E. J. Cands and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20 (2010), 1956-1982.

[3] E. J. Candes and B. Recht. Exact low-rank matrix completion via convex optimization. Annual Allerton Conference on Communication, Control, and Computing, IEEE, (2008) 806-812.

[4] A. Cichocki, R. Zdunek, A. H. Phan and S. I. Amari. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley & Sons, 2009.

[5] J. Fan and T. W. Chow. Matrix completion by least-square, low-rank, and sparse self-representations. *Pattern Recognition*, 71 (2017), 290-305.

[6] M. Fazel. Matrix rank minimization with applications. PhD thesis., Stanford University, 2002.

[7] M. Fazel, H. Hindi and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. *In Proceedings of the 2001 American Control Conference*.IEEE, 6 (2001), 4734-4739.

[8] G. H. Golub, and C. F. Van Loan. Matrix computations. JHU press, 2013.

[9] M. Kurucz, A. A. Benczr and K. Csalogny. Methods for large scale SVD with missing values. *In Proceedings of KDD cup and workshop*, 12 (2007), 31-38.

[10] A.S. Lewis, H.S. Sendov. Nonsmooth analysis of singular values. *Part i: theory, Set-Valued Anal*, 13 (2005), 213241.

[11] X. Li, Y. Ye and X. Xu. "Low-rank tensor completion with total variation for visual data inpainting." *In Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1. (2017).

[12] J. Liu, P. Musialski, P. Wonka and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence* 35 (2012), 208-220.

Table 12: The PSNR of TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$ on the images of Figure 3.

Image	TRLRF	HaLRTC	SiLRTC	FTF-SiLRTC	FTF-SiLRTC-TV
Peppers	13.34	26.14	23.11	23.78	**27.56**
Dog	19.60	**30.84**	**31.77**	**30.76**	**30.81**
House	20.08	27.83	**30.54**	27.24	**30.25**
Parrot	18.26	26.51	27.78	26.79	**29.99**
F16	17.58	23.73	**26.39**	25.10	**26.81**
Squirrel	15.27	26.30	**27.01**	24.31	**26.56**

Table 13: The RMSE of TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$ on the images of Figure 3.

Image	TRLRF	HaLRTC	SiLRTC	FTF-SiLRTC	FTF-SiLRTC-TV
Peppers	54.90	12.57	17.83	16.49	**10.68**
Dog	26.70	**7.32**	**6.58**	**7.39**	**7.34**
House	25.26	10.36	**7.58**	11.08	**7.83**
Parrot	31.15	12.05	10.41	11.67	**8.07**
F16	32.67	16.60	12.22	14.19	**11.65**
Squirrel	43.98	12.35	**11.38**	15.52	**11.98**

[13] Y. Liu, L. C. Jiao and F. Shang. A fast tri-factorization method for low-rank matrix recovery and completion. *Pattern Recognition* 46 (2013), 163-173.

[14] S. Ma, D. Goldfarb and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming* 128 (2011), 321-353.

[15] B. Recht, M Fazel and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review* 52 (2010), 471-501.

[16] J. D. Rennie and N Srebro. Fast maximum margin matrix factorization for collaborative prediction. *In Proceedings of the 22nd international conference on Machine learning* (2005), 713-719.

[17] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software* 11 (1999), 625-653.

[18] L. Tang, X. Cao, W. Chen and C. Ye. An Efficient Tensor Completion Method Combining Matrix Factorization and Smoothness. *Wireless Communications and Mobile Computing* 2021 (2021).

[19] K. C. Toh, M. J. Todd and R. H. Ttnc. SDPT3a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software* 11 (1999), 545-581.

[20] W. Yahya. Image reconstruction from a limited number of samples: a matrix-completion-based approach. *Ph.D. thesis, McGill University Libraries*, (2012).

[21] X. Zhang. A nonconvex relaxation approach to low-rank tensor completion. *IEEE transactions on neural networks and learning systems* 30 (2018), 1659-1671.

[22] P. Zhou, C Lu, Z Lin and C. Zhang. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing* 27 (2017), 1152-1163.

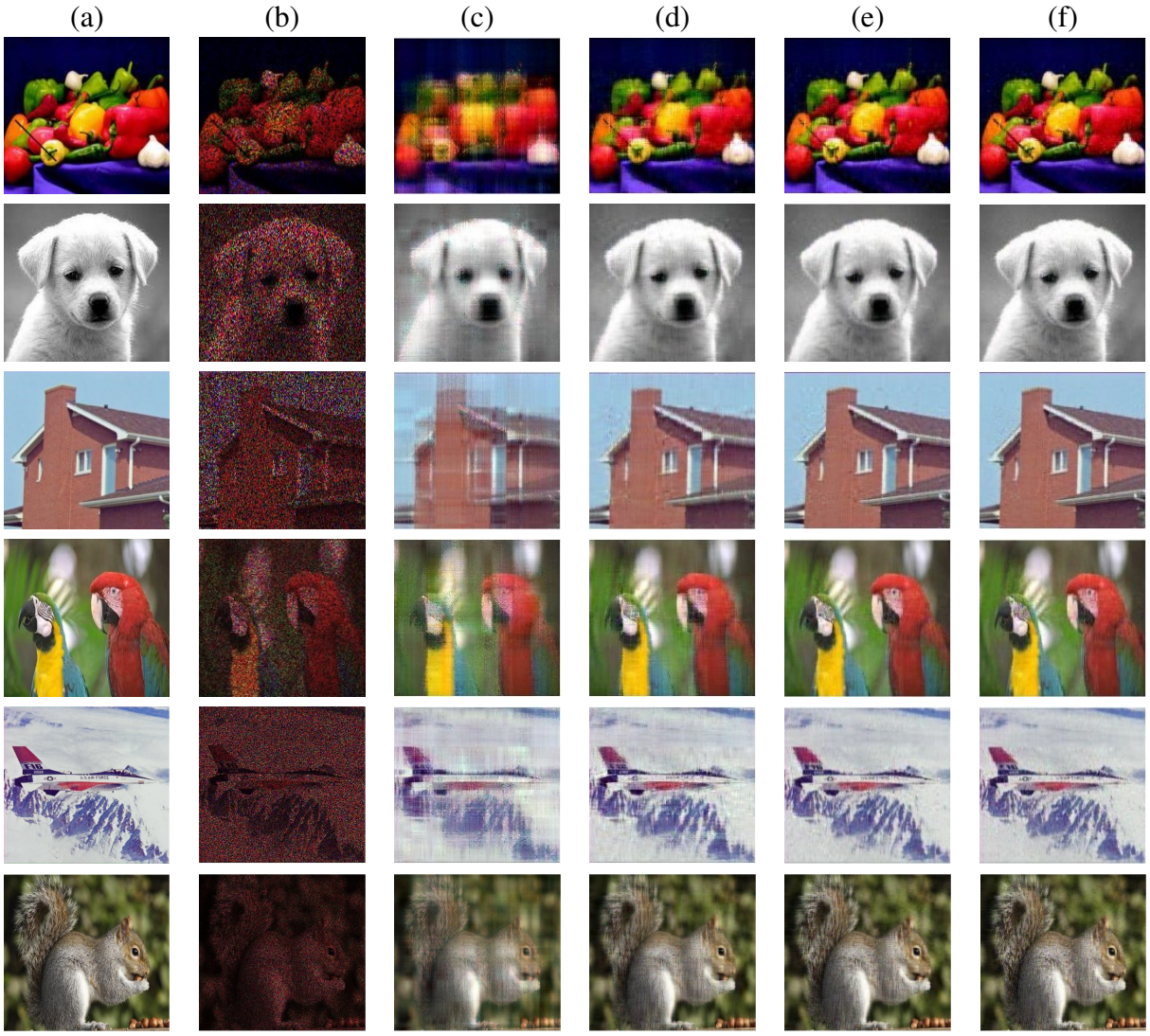


Figure 1: A dataset of images. The columns show (a) original image, (b) corrupted images and (c) to (f) reconstructed by FTF-SiLRTC algorithm with $r = 5, 15, 25$ and 35, respectively.

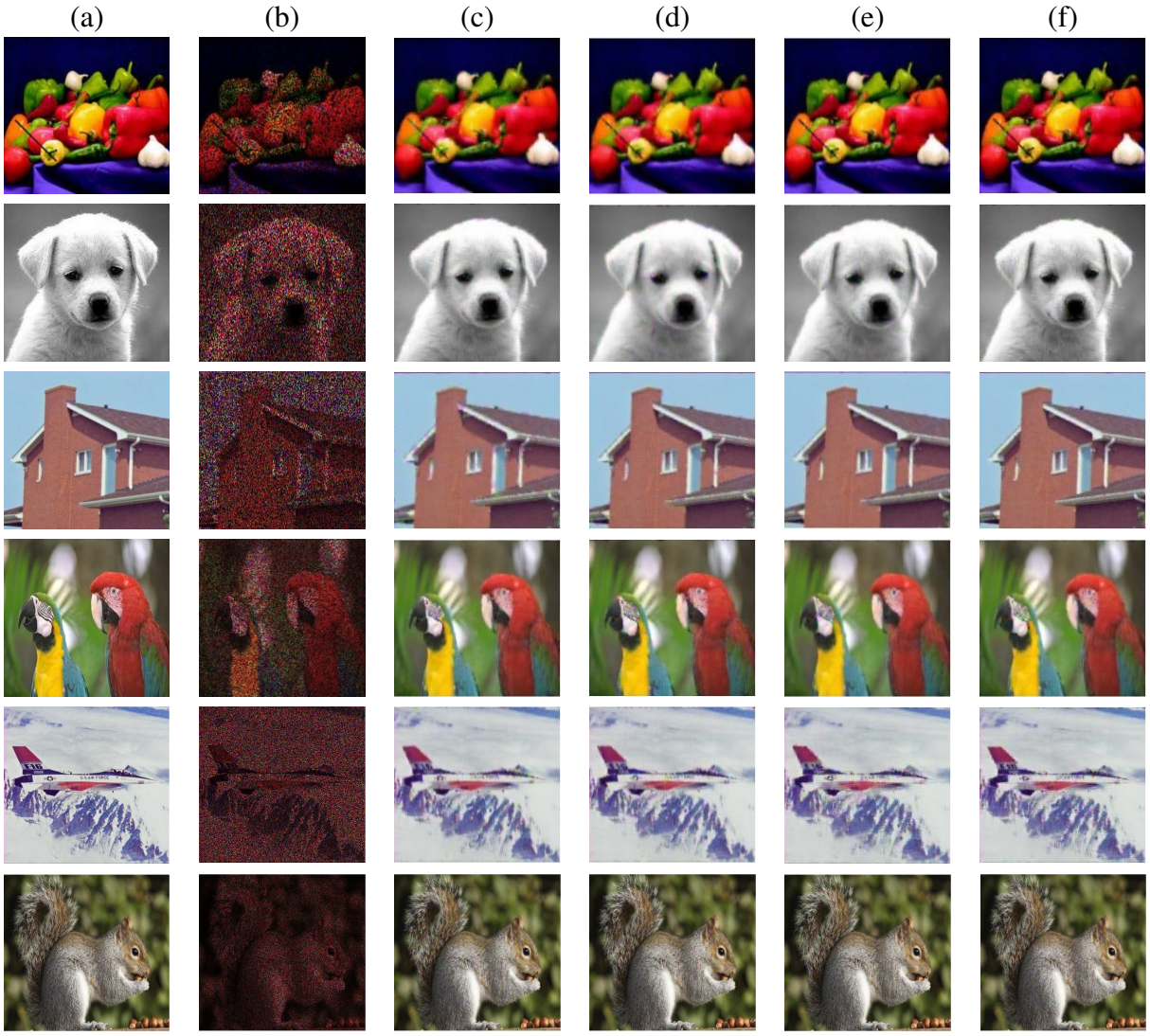


Figure 2: A dataset of images. The columns show (a) original image, (b) corrupted images and (c) to (f) reconstructed by FTF-SiLRTC algorithm with $r = 5, 15, 25$ and 35, respectively.

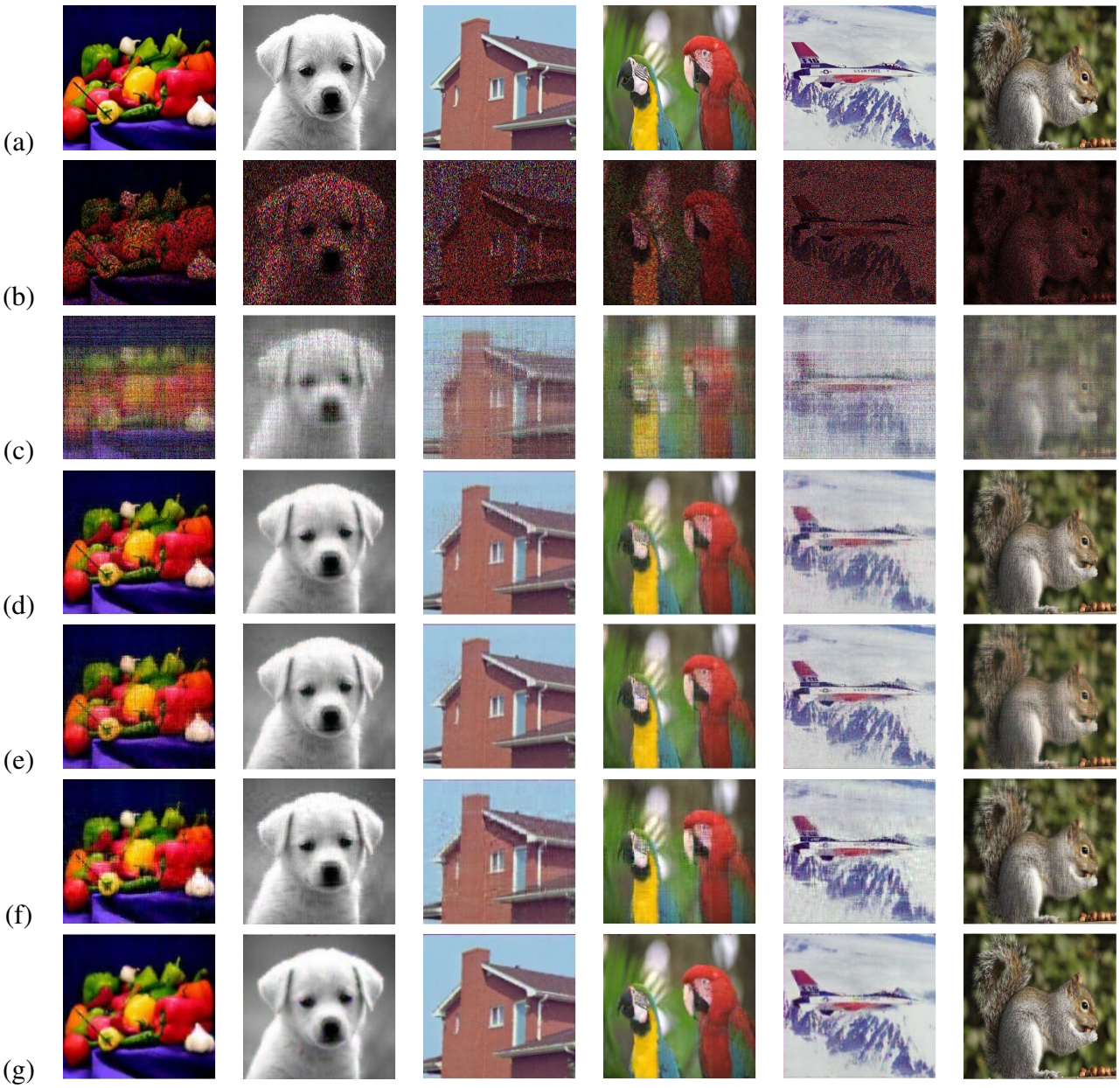


Figure 3: A dataset of images. The rows show (a) original image, (b) corrupted images (c) to (g) reconstructed by TRLRF, HaLRTC, SiLRTC, FTF-SiLRTC and FTF-SiLRTC-TV Algorithms with $r = 15$, respectively.