LOW RANK MODELS FOR MULTI-DIMENSIONAL DATA RECOVERY AND IMAGE SUPER-RESOLUTION

By

Mohammed Al-Qizwini

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering -Doctor of Philosophy

2017

ABSTRACT

LOW RANK MODELS FOR MULTI-DIMENSIONAL DATA RECOVERY AND IMAGE SUPER-RESOLUTION

By

Mohammed Al-Qizwini

In the past decade tremendous research efforts focused on signals with specific features, especially sparse and low rank signals. Researchers showed that these signals can be recovered from much smaller number of samples than the Nyquist rate. These efforts were promising for several applications in which the nature of the data is known to be sparse or low rank, but the available samples are much fewer than what is required by the traditional signal processing algorithms to grant an exact recovery.

Our objective in the first part of this thesis is to develop new algorithms for low rank data recovery from few observed samples and for robust low rank and sparse data separation using the Robust Principal Component Analysis (RPCA). Most current approaches in this class of algorithms are based on using the computationally expensive Singular Value Decomposition (SVD) in each iteration to minimize the nuclear norm.

In particular, we first develop new algorithms for low rank matrix completion that are more robust to noise and converge faster than the previous algorithms. Furthermore, we generalize our recovery function to the multi-dimensional tensor domain to target the applications that deal with multi-dimensional data. Based on this generalized function, we propose a new tensor completion algorithm to recover multi-dimensional tensors from few observed samples. We also used the same generalized functions for robust tensor recovery to reconstruct the sparse and low rank tensors from the tensor that is formed by the superposition of those parts. The experimental results for this application showed that our algorithms provide comparable performance, or even outperforms, state-of-the-art matrix completion, tensor completion and robust tensor recovery algorithms; but at the same time our algorithms converge faster.

The main objective of the second part of the thesis develops new algorithms for example based single image super-resolution. In this type of applications, we observe a low-resolution image and using some external “example” high-resolution – low-resolution images pairs, we recover the underlying high-resolution image. The previous efforts in this field either assumed that there is a one-to-one mapping between low-resolution and high-resolution image patches or they assumed that the high-resolution patches span the lower dimensional space. In this thesis, we propose a new algorithm that parts away from these assumptions. Our algorithm uses a subspace similarity measure to find the closes high-resolution patch to each low-resolution patch. The experimental results showed that DMCSS achieves clear visual improvements and an average of 1dB improvement in PSNR over state-of-the-art algorithms in this field.

Under this thesis, we are currently pursuing other low rank and image super-resolution applications to improve the performance of our current algorithms and to find other algorithms that can run faster and perform even better.

To my family…

ACKNOWLEDGMENTS

First of all, I wish to express my great gratitude to my advisor, Dr. Hayder Radha, for his guidance and support throughout all stages of this research. His advice and encouragement have provided me with the skills needed to develop and refine this research. Special Thanks for Dr. Selin Aviyente, Dr. Xiaobo Tan and Dr. Xiaoming Liu for agreeing to serve on my committee.

I would like to acknowledge and thank Michigan State University (MSU) for allowing me to conduct my study and research here. I also wish to thank all my friends and colleagues in the Electrical and Computer Engineering departments at MSU. A huge thank to all my friends who have assisted and supported me in so many ways during my study.

Lastly, I would like to express my great gratitude to my mother, father, brothers and sisters for all their support and encouragement for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Matrix Completion

According to the Nyquist-Shannon sampling theorem, a signal can be completely determined by its samples if the sampling rate is twice the maximum frequency of that signal. On the other hand, there are numerous applications where sampling at Nyquist frequency is either impossible, very expensive or time consuming [1]. In addition to that the tremendous number of images and videos that are being generated, transmitted and stored on the internet, researchers suggested that instead of the usual data acquisition method of collecting as much information as possible and then through away the redundant information, why not collecting only the information necessary to recover the data. Researchers proved that if the underlying signal has specific properties, then it can be recovered using much fewer samples than the Nyquist rate, for instance, if the signal is sparse or it can be transformed into a sparse domain, then only few number of samples are required to recover that signal using the Compressive Sampling framework [2]–[4]. From a similar concept to Compressed Sensing, Matrix Completion also emerged during the past decade as one of these theories and several researches are being performed in this topic to this day. The main concept behind matrix completion is recovering the underlying low rank data from only few observed samples of the original signal.

There is a large number of applications in which we observe an incomplete matrix of measurements and we wish to fill in the unobserved matrix entries using the observed measurements [5], [6]. For instance, in collaborative filtering, the famous Netflix problem, in which the vendor has a large number of users and a set of movies, each user represents a row and each column represents a movie, and the challenge was to predict each user's ratings for the movies that he/she didn't watch based on the ratings provided by the user for the movies that he/she watched in order to recommend other movies to that user. Because there is a small number of factors that can affect users' taste of preferences about movies, these matrices are considered to be low rank matrices [7], [8]. Other applications include recovering missing pixels in images for computer vision, system identification in control, multi-class learning in data analysis, global positioning of sensors in a network, partial distance information, remote sensing applications in signal processing and many statistical problems involving succinct factor models [7]. Here we stop to ask a question, is it possible to recover the unknown entries of a matrix given a subset of its known entries? A first thought to the problem will give an answer no to the question, however if the matrix is low rank, then it is possible to recover the original matrix with high probability using nuclear norm minimization [5], [8], [9].

1.2 Robust Principal Component Analysis

Principal Component Analysis (PCA) is one of the most used statistical techniques in dimension reduction. However, the performance of PCA-based approaches degrades significantly for non-Gaussian noise, and especially for grossly corrupted observations or in the present of outliers [9]–[11]. To overcome these shortcomings of traditional PCA and make it more robust, Candes *et al.* [10] suggested invoking the assumption that the observed data consists of two components, a low rank component L and a sparse component S and they introduced a novel algorithm based on tractable convex optimization to solve the problem within a polynomial time and a strong performance guarantee. Consequently, the problem can be reduced to the recovery of L from highly corrupted measurements $D = L + S$, in which S can have arbitrary large elements, but must be a sparse matrix. Furthermore, to make such modeling approach viable and meaningful, the low rank component L is assumed to be not sparse [9], [10].

For some applications, the sparse matrix S could carry useful information. For instance, on latent semantic indexing, the input matrix D contains entries that encode the relevance of a word to a document. If D can be decomposed into two components, low rank and sparse, then L could represent common words, and S captures some key words that could be used to distinguish each document from others [9], [12]. On the other hand, some applications consider S as merely noise or outliers and they are only interested in recovering$L$. For example, in the face recognition problem S represents the noise and outliers such as facial expression changes or glasses or other objects. While L represents the original facial expressions to be recovered [13].

Recovering the low rank and sparse components of an observed multidimensional signal (tensor) that is formed by the superposition of both components is a very interesting problem and it has a wide range of applicability in computer vision, bioinformatics and graph analysis [10],[14]. Also, since most applications of interest, especially those related to imaging and computer vision, have to handle multidimensional data that are inherently 3D or 4D [15], [16]. Applying the recovery algorithms to each two dimensional plane separately ignores the internal correlation in the signal.

1.3 Image Super-resolution

Image super-resolution is the ill-posed problem of recovering the underlying High-resolution (HR) image from an observed Low-resolution (LR) image(s). Image super-resolution is important in several computer vision applications such as license plates recognition [17], object and face recognition [18] and restoration of historical images i.e. image inpainting [19] as it is promising to overcome the shortcoming of low cost camera sensors and allows a better utilization to the high-resolution screens. Many scenarios have been suggested for image super-resolution based on the availability of additional information to help with the recovery. In some scenarios additional LR images for the same object are available but under different conditions which are referred to as multi-frame image super-resolution [20]–[22]. In other scenarios no training images are used, instead researchers base their approaches to recover the HR image on information from the available LR image itself [23]–[25]. Other approaches assume only one LR image is available and examples of training HR and LR images are used to recover the target HR image, which is called example based image super-resolution [26]–[29]. This thesis falls in the example based single image super-resolution category since we will be using two external dictionaries, one for LR example images and the other for HR images.

1.4 Notations

Throughout this thesis, matrices are denoted using capital letters, e.g. M, and tensors by cursive capital letters, e.g. M. Also, M^T is the transpose of M. $M_{i_1 i_2}$ is an entry of the matrix $M \in \mathbb{R}^{m \times n}$ at row i_1 and column i_2; and $\mathrm{M}_{i_1 \dots i_N}$ is an entry of the tensor M $\in \mathbb{R}^{m_1 \times m_2 \dots m_N}$

at indices i_1 through i_N. For a matrix $M \in \mathbb{R}^{m\times n}$, $P_\Omega(M)$ is the sampling operator which is defined as:

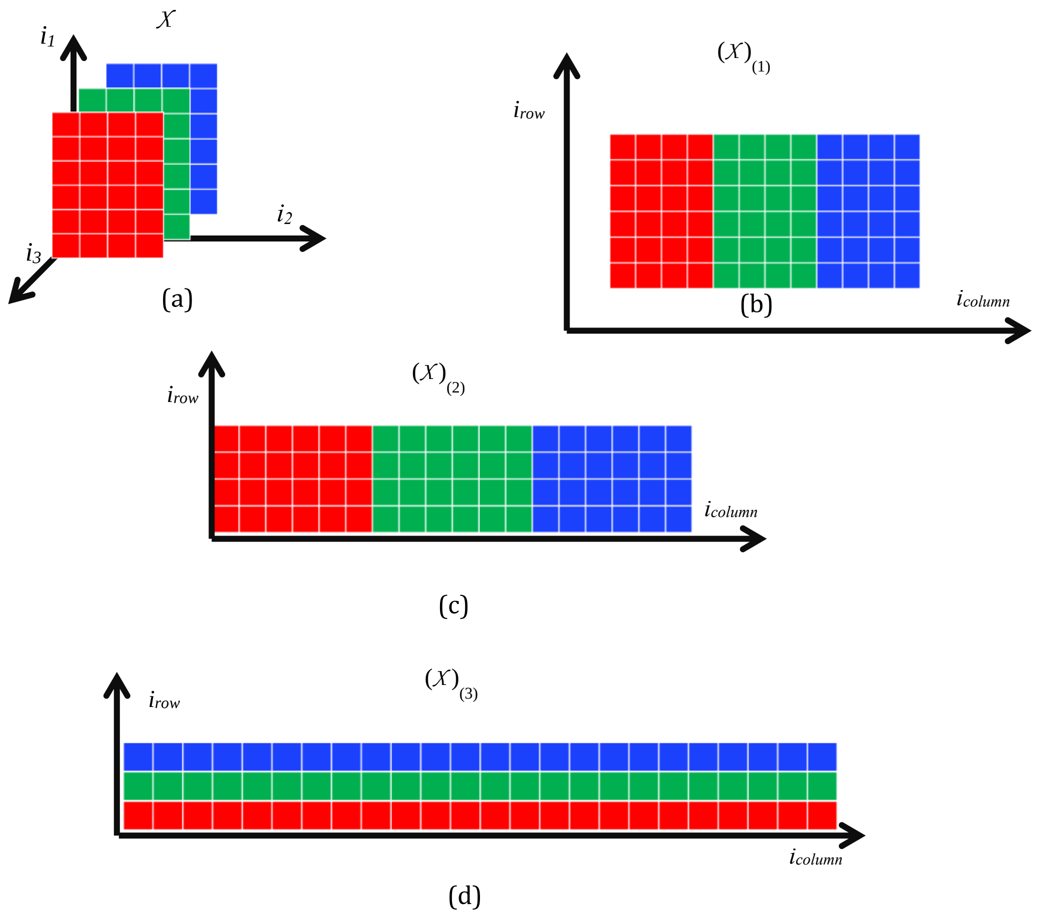


Figure 1.1: Visual explanation of the $(.)_{(.)}$ function in tensor domain. (a) the tensor in 3D domain, (b) the unfolded tensor along i_1, (c) the unfolded tensor along i_2 and (d) the unfolded tensor along i_3.

$$P_\Omega(M) = \begin{cases} M_{i_1 i_2} & if\ i_1, i_2 \in \Omega \\ 0 & elsewhere \end{cases} \tag{1.1}$$

where Ω represents the set of observed entries.

For M $\in \mathbb{R}^{m_1 \times m_2 \dots m_N}$, the $unfold(.)_{(i)}$ operation over dimension i unfolds the N-dimensional tensor into a matrix, $fold(.)_{(i)}$ is the inverse of $unfold(.)_{(i)}$ and it can be defined as:

$$\text{M}=fold((\text{M})_{(i)})_{(i)} \tag{1.2}$$

A visual example of the tensor unfold operation is shown in Figure 1.1.

The inner product between two matrices is defined by:

$$\langle C, E \rangle = Tr(CE) \tag{1.3}$$

where $Tr(.)$ is the trace function.

If $M = U\Sigma V^T$ is the SVD decomposition of M, then the soft thresholding operator over M is defined as [30], [31]:

$$D_{Th}(M) = U\Sigma_{Th}V^T \tag{1.4}$$

Where $\Sigma_{Th} = max(\sigma_i - Th, 0)$ is the diagonal soft thresholded singular values matrix.

1.5 Literature Survey

1.5.1. Matrix Completion

For the past decade, many algorithms have been proposed mainly for matrix completion and several variations of the problem has been suggested. In this section, we are going to talk about some of the important work that has been done for matrix completion that is related to our contributions in this field.

E. J. Candès and B. Recht, proposed the first convex programming algorithm to solve the exact matrix completion algorithm in [5], they showed that low rank matrices can be recovered with very high probability if the number of observed samples (m) obey:

$$m \geq Cn^{1.2} r \log n \tag{1.5}$$

For some positive constant C, most $n \times n$ matrices with rank r can be recovered perfectly using convex optimization. They also showed that by replacing the 1.2 exponent with 1.25 the recovery is guaranteed for all rank values.

E. J. Candès and Y. plan proposed a relaxation to the equality constrains of the matrix completion problem in [7] to reduce the effect of noise on the signal reconstruction performance. They performed numerical and quantitative analysis and showed that nuclear norm minimization can perfectly fit large low rank matrices from only few observed noisy samples.

A simple Singular Value Thresholding (SVT) algorithm for matrix completion was proposed in [32], the algorithm performs matrix completion by soft thresholding the singular values of the SVD decomposition of the estimated matrix. The authors provided convergence analysis of the algorithm and the experimental results showed that 1000×1000 matrices can be recovered within one minute on an average desktop computer.

A Fixed Point Continuation Algorithm with approximate SVD decomposition (FPCA) for nuclear norm minimization were proposed in [31]. The authors compared the algorithm to other semidefinite programming algorithms and showed that FPCA converges much faster than the others and at the same time it provided much better performance. Their experimental results showed that on 1000 ×1000 matrix with rank 50, they achieved 10^{-5} error and the algorithm converged within 3 minutes.

In [33], the authors used the nuclear norm regularized least square instead of minimizing the nuclear norm. The regularized least square is the sum of a convex smooth function with Lipschitz continuous gradient and a convex function on a set of matrices. An accelerated proximal gradient algorithm is proposed to solve the objective function. The experimental results showed that the algorithm is robust and efficient to solve large scale matrices of dimensions up to 10^5 in less than 10 minutes on an average personal computer.

In [34], the authors proposed a novel method by only minimizing the smaller singular values from the nuclear norm and leave the significant ones unchanged instead of all the singular values. The authors build an algorithm using the alternative direction method of multiplier (ADM) based on the Truncated Nuclear Norm Regularization (TNNR) by only minimizing the $N - r$ singular values where N is the number of singular values and r is the rank of the matrix. The authors compared the performance of the algorithm against state of the art algorithms on grayscale and colored images and showed that the proposed algorithm achieve significant improvement in performance.

A new set of Iterative Reweighted Least Squares (IRLS-p) where $0 \geq p \leq 1$ algorithms was proposed by K. Mohan and M. Fazel in [35]. The algorithms are used as a computationally efficient variation of the nuclear norm minimization. The authors also present smoothed version of the algorithms sIRLS-p which shown very fast convergence results and at the same time improved the recovery performance when compared to the conventional nuclear norm minimization algorithms.

Also recently, the notion of matrix completion over samples collected from different times has been proposed in [36]; the author modified and proposed several algorithms that performs adaptive matrix completion when the samples are collected from a process that is being changed over time and the goal is to recover all the underlying low rank matrices. The proposed algorithm that performed the best in their experiments is a variation to the SVT algorithm in [32] and they referred to it as the Adaptive Singular Value Thresholding (ASVT). ASVT performance is evaluated against state-of-the-art matrix completion algorithms and it showed that ASVT outperforms all the other algorithms using the relative error measure.

1.5.2. Tensor Completion

Although the matrix completion problem has been studied thoroughly during the past years, less research was done in tensor completion due to the complexity of high dimensional mathematical analysis and the lack of definitions of simple matrix norms when we move to the higher dimensional space. Despite this difficulty, some researchers overcome these problems and proposed different algorithms to solve it.

Ji Liu et.al. proposed three algorithms for Low Rank Tensor Completion (LRTC) based on nuclear norm minimization over each fold of the tensor in [30], [37]. The SiLRTC is a simple algorithm and it implement a relaxation technique to separate the dependent components and the Block Coordinate Descent (BCD) to solve for each component separately and achieve the global optimization, the FaLRTC uses a smooth scheme to change the objective function to a smooth function and solved the general nuclear norm minimization problem. Finally, HaLRTC uses the Alternative Direction Methods of Multiplier (ADM) to minimize the tensor nuclear norm. The experimental results showed that all algorithms are applicable for image and video signals and the algorithms are more robust and faster than the available tensor completion algorithms. They also showed that FaLRTC and HaLRTC are more efficient that SiLRTC while FaLRTC is the most efficient algorithm.

In [38], the authors suggests minimizing the nuclear norm of the tensor without metricizing the tensor which was shown theoretically to perform better than minimizing the nuclear norm. The authors also develop a series of algebraic and probabilistic techniques such as characterization of sub-differential for tensor nuclear norm and concentration inequalities for tensor martingales, which are useful in other tensor related problems.

Based on the work in [34], L T Huang et.al. proposed an algorithm for tensor completion using the truncated nuclear norm in the tensor domain [39]. The used the Alternative Direction Method of Multiplier (ADM) to solve the problem, and they reduced the two-step solution in [34] to one-step solution. They also ran extensive experimental results and showed that their algorithm performs significantly better than the state-of-art algorithms.

In [40], the authors proposed an algorithm for tensor completion with high ratio of missing samples, in which situations the usual low rank and smoothness assumptions do not work. The authors address this issue by applying a novel PARAFAC decomposition. The decomposition is represented as a sum of the outer product of functional smooth component vectors, which are represented by linear combinations of smooth basis functions. Based on that, they also developed an algorithm with greedy deflation and rank-one tensor decomposition. The experimental results showed that this algorithm performs better than state-of-the-art algorithms for tensor completion.

In [41] the authors proposed an alternative method for tensor completion using Tensor Train (TT) rank which is capable of capturing hidden information from tensors. Based on that, a new optimization function is proposed for tensor completion and two algorithms were proposed for tensor completion; the first one is SiLRTC-TT which uses the usual nuclear norm minimization based on TT rank and the other algorithm uses a multilinear factorization model to approximate the TT rank of a tensor, and is called tensor completion by parallel matrix factorization via TT (TMac-TT). The experimental results showed significant performance improvement of the proposed algorithms compared to the existing algorithms.

1.5.3. Robust Tensor Recovery

Similar to Matrix and Tensor Completion, despite the applicability of robust tensor recovery to several real-life applications, the area of Tensor recovery witnessed less research interest than robust matrix completion. In this section, we will talk about the key research articles that tackles the problem of robust matrix and tensor recovery.

E. J. Candès et.al. in [10] proved that it is possible to separate a matrix that is generated by superposing a low rank and sparse matrices into its low rank and sparse components by solving convex problem called principal component pursuit. The algorithm they proposed works by minimizing both the nuclear norm of the predicted low rank matrix and the ℓ_1 norm of the sparse component. The authors also present several practical applications for this method such as video surveillance where the algorithm was able to successfully detect objects in cluttered environment, face detection where the algorithm is used to remove shadow and secularities.

In [42] and similarly [43] the authors proposed a method to recover low rank matrices from corrupted observation by using robust matrix recovery. In this method, the noise matrix is considered the sparse portion and the recovered low rank matrix represents the predicted "clean" matrix. The authors provided a proof that most matrices can be recovered with very high probability using this approach. The algorithm was applied to several computer vision problems and it shows promising performance.

For the tensor domain, Goldfarb and Qin apply convex optimization to recover the low rank tensor using high order tensor PCA [14]. They propose an optimization algorithm based on the ADM and the Accelerated Proximal Gradient (APG) algorithms with convergence guarantees for solving both the constrained and the Lagrangian forms of the problem. They also propose a non-convex model that improves the results of the recovery from the convex model. They apply their algorithm to several real computer vision applications and showed the practical use of the proposed method.

In [44] the authors used a new model for tensor Singular Value Decomposition (tSVD) that was proposed in [45] for third order tensors only. The proposed method uses ADM algorithm to separate the low rank and sparse components of the observed tensor. The algorithm was applied for colored image denoising and it showed significant performance improvements over the previous algorithms.

1.5.4. Image Super-resolution

Since our contributions in this thesis are related to single image super-resolution, we will only mention publications in the same category as our contributions.

J. Yang et.al. in [26] address the problem of image super-resolution from the compressed sensing perspective by assuming that the observed low-resolution image is a down sampled version of a high-resolution image. The patches of the high-resolution image is assumed to have a sparse representation. They also show the effect of the sparsity assumption for solving the ill-posed super-resolution problem. The authors also form a dictionary using a small set of randomly chosen raw patches from training images of similar statistical nature to the input image. The experimental results showed that this method produce superior quality compared to the other super-resolution methods.

In [46], the authors explore the relationship between neighbor pixels and the estimation of the high-resolution image from the observed low-resolution input. The authors propose an algorithm based on matrix completion concept by minimizing the sum of all the augmented matrices' rank. The authors tested the algorithm on different applications and showed that the resulted image has both better visual quality and higher PSNR values.

Another super-resolution algorithm based on clustered sparse representation and adaptive patch aggregation was proposed in [47]. The training patches pairs are collected randomly from example images and using K-means clustering algorithm, the patches are divided into multiple groups. The over-complete dictionary is learned over the clustered pairs. Only one cluster, sub-dictionary, is selected to represent the low-resolution patch. The authors compare the performance of the algorithm against several state-of-the-art algorithms for super-resolution and it showed that the proposed algorithm outperformed the rest of the methods.

In [48], the authors propose an algorithm based on two other super-resolution algorithms, Anchored Neighborhood Regression (ANR) [49] and Simple Functions (SF) [50]. The resulted algorithm (A+) builds on the features and anchored regressors from ANR but instead of learning on the training dictionary, it learns on the entire training material, similar to SF. The authors used different example images and they showed that the PSNR improved by 0.2-0.7 dB over the ANR method, at the same time, the algorithm runs with low computational complexity.

A manifold linear approximation based approach for single image super-resolution was proposed in [28]. The authors exploit the non-linear space of the underlying high-resolution patches by considering it as low dimensional manifold in a high dimensional Euclidean space. The Sparse Subspace Clustering (SSC) algorithm is used to create the set of low-dimensional linear spaces that are considered as tangents in the higher resolution patches. Based on the obtained approximated tangent spaces, the structure of the underlying high-resolution manifold is used to locate the corresponding high-resolution subspace. This approach requires a small number of training high-resolution samples, around 1000 patches, without any prior assumption about the low-resolution images. A comparison of the obtained results with other state-of-the-art methods clearly indicates the viability of the proposed approach.

In [23], the authors proposed a learning-based method for image super-resolution using the low-dimensional manifold representation of high-resolution image patches space. The authors used the image and its down sampled scale to extract a set of training sample points using min-max algorithm. The ℓ_1 norm of the resulted graph is then minimized to cluster these samples into a set of manifold neighborhoods and the high-resolution patch is estimated from these tangent spaces. The experimental results validate the effectiveness of the algorithm when compared to relative methods.

1.6 Contributions

1- Derive two robust algorithms for matrix completion using the Accelerated Proximal Gradient (APG) and the Alternating Direction Method of Multipliers (ADM).

Further, we compare both algorithms against each other and against the iterative reweighted least squares (IRLS-1) algorithm using a variety of noisy images.

2- Extend the efficient matrix completion algorithms to the *N*-dimensional tensor completion problem by first providing a general definition for the Schatten-P function into the *N*-dimensional space. We also derive a multi-dimensional Augmented Lagrangian Multiplier (ALM) optimization algorithm to solve it.

3- Formulate a new objective function for the tensor RPCA problem based on using the truncated and smoothed schatten-*p* functions. We also solve the formulated problem using the Augmented Lagrangian Multiplier (ALM) algorithm, which is the trivial algorithm to use with RPCA. We also proposed two other algorithms based on the Accelerated Proximal Gradient (APG) and the Alternating Directions Methods of Multipliers (ADMM). We refer to our proposed method as the Truncated and Smoothed Robust Tensor Recovery (TSRTR) algorithm.

4- Propose a novel image super-resolution framework that parts away from the similarity assumption between the low dimensional and high dimensional manifolds; instead the subspace similarity measure is used to find the closest HR subspace to each LR subspace. The test patch is projected to the closest low dimensional subspace, and the most similar HR subspace is selected and hence we get a better approximation to the LR test image than directly projecting the LR test patch to the HR space.

1.7 The proposal outline

This thesis is divided into two major parts. In the first part, the low rank matrix and tensor completion problems are explored and we propose new algorithms for both matrix and tensor completion that outperforms state of the art algorithms in this field. The same part also studies the Robust PCA problem and its extension to the multi-dimensional tensor domain; we also propose a new algorithm for the tensor Robust PCA problem that outperforms the state of the art algorithms. The other part of the thesis studies the single image super-resolution problem for which we proposed a novel algorithm that makes no assumption about the similarity between the LR and HR subspaces.

In chapter 2, we review state-of-the-art methods related to matrix and tensor completion and robust PCA topics. Most of these methods will be used as a benchmark to compare the performance of our algorithms. The chapter will also provide a detailed description and show our experimental results of the proposed algorithms for Matrix Completion, Tensor Completion and Robust Tensor Recovery.

In chapter 3, we review the available algorithms for example based single image super-resolution and we present our proposed algorithm along with several experimental results to compare the performance to the state-of-the-art algorithms in this field.

Finally, chapter 4 outlines our current efforts and the ongoing research directions and provides some preliminary discussion and analysis of the methods presented in this thesis.

Chapter 2

Matrix and Tensor Completion and Robust Tensor Recovery

2.1 Matrix Rank Minimization

Matrix completion is a special case of what is known as the affine rank minimization problem, defined as:

$$\min_{X} rank(X) \quad s.t. \; A(X) = b \tag{2.1}$$

Where, $X \in \mathbb{R}^{n_1 * n_2}$ is the decision variable, A is an affine mapping $A : \mathbb{R}^{n_1 * n_2} \rightarrow \mathbb{R}^{p}$, and the vector $b \in \mathbb{R}^{p}$ represents the given samples.

Table 2.1: A comparison between some important terms according to the rank minimization and cardinality minimization concepts

Term	**Rank minimization**	**Cardinality Minimization**
Sparsity inducing norm	ℓ_1	Nuclear
Hilbert space norm	ℓ_2 (Euclidean)	Frobenius
Dual norm	ℓ_∞	Operator
Norm additivity	Disjoint support	Orthogonal row and column spaces
Convex optimization	Linear programming	Semi definite programming

The affine transformation A is an operation on a matrix X such that it preserves all the linear properties among data points of X in the resulting vector b. The solution to the problem (2.1) can be stated as the simplest (lowest rank) feasible model that is affine in the matrix variable. In general, this type of rank minimization problems can be solved using the singular value decomposition [51]. In matrix completion we are interested only in finding the matrix with the lowest rank to fill in the unobserved entries based on the small number of samples that we have, under the constraint that, in the regions where we have samples, the values of the reconstructed samples should be the same as the original ones. The affine rank minimization problem (2.1) is reduced to the cardinality minimization problem of finding the element in the affine space that has the smallest number of nonzero singular values. In this section, we provide a comparison between the rank and cardinality minimization. The main comparison points are listed in Table 2.1 [51].

The main idea behind matrix completion is to find the minimum rank matrix that fits the observed samples, so if there is only one low rank matrix that fits the observed data, this would recover the original matrix M [1]. Mathematically, this can be represented by [1, 2, 3]:

$$\begin{gathered} \min_{X} rank(X) \\ s.t.\ P_{\Omega}(X) = P_{\Omega}(M) \end{gathered} \tag{2.2}$$

Where Ω is the region of the observed samples; M_{ij} is an observed sample and X_{ij} is an estimated sample.

It is important to mention that not all low rank matrices can be recovered from few observed samples. Let us look at the matrix M below [5]:

$$M = e_1 e_n^* = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 \end{bmatrix} \tag{2.3}$$

The matrix M is formed by the multiplication of the first canonical eigen vector $e_1$by the n-th canonical eigen vector e_n^*. Even though M is a rank one matrix, it cannot be recovered from partially observed samples, unless we observe the one in the top right corner. The reason is that the sampling sets will mostly see only zeros and there is no way for us to guess that there is an element of different value unless we observe it. Otherwise, the predicted matrix will always be zero no matter what recovery method we use.

Therefore, not all low rank matrices can be recovered from a set of observed entries, to understand the set of matrices that can be recovered from partially observed samples, we need to look at the SVD decomposition of a matrix X [5], [31].

$$X = \sum_{k=1}^{r} \sigma_k u_k v_k^T \tag{2.4}$$

where u_k and v_k^T are both the left and right singular vectors and σ_k is the singular value at index k. In these terms, the generic set of recoverable low rank matrices can be thought of as the family $\{u_k\}_{1 \le k \le r}$ is selected uniformly at random among all families of r orthonormal vectors; the same applies for $\{v_k\}_{1 \le k \le r}$ and the families may or may not be dependent on each other [5].

To show that the matrix cannot be in the null space of the sampling operator that is providing the observed samples, let us take a look at the SVD representation of the rank-2 symmetric matrix M below [5]:

$$M = \sum_{k=1}^{2} \sigma_k u_k {v_k}^T \tag{2.5}$$

where $u_1 = \frac{e_1+e_2}{\sqrt{2}}, u_2 = \frac{e_1 1 e_2}{\sqrt{2}}$.

This matrix has all zero entries except for the 2×2 top left corner. Again, we will need to observe almost all entries of this matrix in order to recover it using any method. The reason is that the singular vector representation for this matrix is concentrated in one area which is the top-left corner in this example. In conclusion, the singular vectors of the matrix to be recovered need to be spread, uncorrelated with the standard basis, in order to reduce the number of observations required for the recovery [5], [7].

Another important aspect of successfully recovering a low rank matrix from few observed samples, is the way we sample the matrix. For example, we cannot hope to recover a matrix $X$from partially observed samples if the sampling operator misses a full row or column. Assume X is a matrix of rank one and it is formed by multiplying bc^* where $b, c \in \mathbb{R}^n$. Each entry of this matrix can be represented by [5]:

$$X_{i,j} = b_i c_j \tag{2.6}$$

In this case, if we do not observe any sample from the first row for example, then there is no way for us to predict the first element b_1. This example can be extended to all rows and columns. Another example from collaborative filtering is shown in Figure 2.1; assume that the matrix represents the ratings that a user (row) has given to particular movies (column); each element in the matrix represent the rate that a user is given to the corresponding movie on a scale 1(not satisfied) to 5 (extremely satisfied). The question marks represent unrated movies by the corresponding user. The red rectangles represents users who didn’t rate any movies and hence we have no information about their preference or taste in movies. Therefore, this sample of the matrix cannot be recovered using matrix completion.

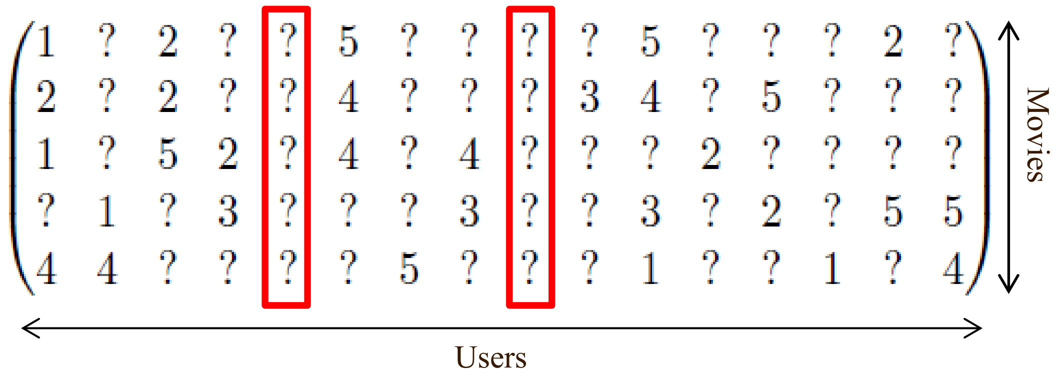


Figure 2.1: Collaborative filtering example of bad sampling for low rank matrix recovery.

So in order to recover a low rank matrix with high probability from partially observed samples, we need a set of samples that are collected uniformly at random from the underlying matrix.

2.2 Matrix Completion

The equation (2.1) is an NP-Hard problem and all algorithms that solve it are doubly exponential in theory and practice [5], [7], [8], [51]. For the past decade, the main algorithms for matrix completion are based on the nuclear norm minimization [5], [31], [32], [34], [52]–[54], which was introduced in [55] after proving that the nuclear norm is the tightest convex approximation to the rank function. Several algorithms have been proposed to recover a low rank matrix from a subset of its observed entries and it has been used in many different fields, for example computer vision [56], sensor networks [57], [58] and control [59]. The core idea of this recovery is based on the nuclear norm minimization [5], [31], [32], [34], [52]–[54].

In an ideal scenario, the convex approximation to the matrix completion problem can be represented by:

$$\begin{aligned} &\min_{X} \ \|X\|_* \\ &s.t.\ P_\Omega(X) = P_\Omega(M) \end{aligned} \tag{2.7}$$

where X is the low rank matrix to be estimated, M is the original matrix.

Since the observed samples are usually contaminated with noise, we need to relax the equality constraint into an inequality constraint in order to make (2.7) more robust to noise [54].

$$\begin{aligned} &\min_{X} \ \|X\|_* \\ &s.t.\ \|P_\Omega(X\ ) - P_\Omega(M)\|_F^2 \leq \varepsilon \end{aligned} \tag{2.8}$$

Subsequently, the authors in [60] proposed to use the reweighted least squares algorithm to recover sparse signals from few observed samples. In [61] the authors introduced the reweighted nuclear norm concept into the matrix completion problem. The same authors proposed an efficient iterative algorithm [35], and they showed that it is more computationally efficient and achieves better performance than the heuristic matrix completion algorithms that require evaluating the computationally expensive Singular Value Decomposition (SVD) in each iteration. The IRLS-p algorithm in [35] used a simple gradient minimization approach to obtain the optimal minimum of a reweighted rank approximation function. Instead of minimizing the convex nuclear norm function in (2.7), the authors in [35] suggested to use a smooth approximation to the rank function and they showed that it converges faster than minimizing the nuclear norm function; they considered the smooth Schatten-p function:

$$f_p(X) = Tr(X^T X + \gamma I)^{\frac{p}{2}} \tag{2.9}$$

Here, I is the identity matrix and $\gamma > 0$. The function (2.9) is convex for $p \geq 1$ [35].

So the matrix rank minimization approximation problem can be written as [35]:

$$\begin{aligned} &\min_X Tr(X^T X + \gamma I)^{\frac{p}{2}} \\ &s.t.\ P_\Omega(X) = P_\Omega(M) \end{aligned} \tag{2.10}$$

The authors also provided a theoretical guarantee that when $p = 1$ the algorithm provides similar results to minimizing the nuclear norm. The main issue with this iterative algorithm is that in each iteration, the authors used a strict equality constraint for the region of observed samples, and hence, when the observed samples are contaminated with noise, the algorithm will converge to the noisy version of the data.

2.3 Tensor Completion

In several computer vision applications, we usually deal with a higher dimensional space, we find few studies considered the *N*-dimensional tensor completion case because of the difficulties associated with higher dimensional space computations [30], [62]. Some researchers, for example in [34], applied their matrix completion algorithms to color images by considering each color channel as a separate matrix. Such algorithms don't consider the correlation among the different channels. Previous algorithms solved the matrix and tensor completion problems based on minimizing the nuclear norm, which has been shown [55] to be the tightest approximation to the rank function [5], [30], [31], [54], [56], [63]. Some other algorithms also used a reweighted nuclear norm, for example in [61], to achieve better results than using only the truncated nuclear norm. However, minimizing the nuclear norm requires evaluating the computationally expensive SVD in each iteration. As mentioned earlier, in [35] the authors proposed an extension of the work in [60] into the matrix completion problem and proposed computationally efficient algorithms for matrix completion. In this thesis we propose two matrix completion algorithms based on the schatten-p functions that are more robust to noise than the previous algorithms.

Tensor completion is a generalization of the matrix completion concept explained in section 2.5.2. We generalize the completion algorithm for the matrix (i.e., 2- mode or 2-order tensor) case to higher-order tensors by solving the following optimization problem [30], [39], [64]:

$$\begin{gathered} \min_{\mathcal{X}} \ \|\mathcal{X}\|_* \\ s.t. \ P_\Omega(\mathcal{X}) = P_\Omega(\mathcal{M}) \end{gathered} \tag{2.11}$$

where both $\mathcal{X}$ and M are n-mode tensors which have the same size in each mode.

One issue with this definition is that the nuclear norm of a high dimensional tensor has no unique definition, the mostly used format in tensor completion application is the average of the nuclear norm of the matrix formed by unfolding the tensor along each dimension [38], [39], [65]. This can be represented mathematically as:

$$\|\mathcal{X}\|_* = \frac{1}{n}\sum_{i=1}^{n}\left\|(\mathcal{X})_{(i)}\right\|_* \tag{2.12}$$

Since the unfolded tensors tend to form a relatively large matrix, the use of SVD decomposition for each unfold will take much longer time. In this thesis, we will also extend these efficient Schatten-p function that we defined for matrix completion in equation (2.9) to the multi-dimensional tensor domain and propose a tensor completion algorithm based on these functions that outperforms state of the art tensor completion algorithms and at the same time reduces the computation time.

2.4 Robust Tensor Recovery

In matrix domain, several algorithms were developed to separate the low rank and sparse models which is known as the robust matrix recovery. The main idea is based on minimizing both the nuclear norm and the ℓ_1 norm of the estimated low rank and sparse components [10], [42], [43], [66].

$$\begin{gathered} \min_{X,E} \|X\|_* + \lambda\|E\|_1 \\ s.t.\ M = X + E \end{gathered} \tag{2.13}$$

where X is the estimated low rank matrix, E is the estimated sparse matrix and M is the observed matrix.

Since most applications in real life, especially in computer vision such as colored images and video, are multidimensional data [15], [16]. Applying the recovery algorithms to each two-dimensional space separately will ignore the internal correlation in the signal. Hence, it is important to process the whole tensor data [16], [67]. In the few past years, some contributions have been made in the robust tensor recovery field. The authors in [15] proposed a robust algorithm for error data correction for image and video signals. The authors in [16] used the multi-linear augmented Lagrangian multiplier for tensor recovery and they applied it to recover corrupted multidimensional data. The authors in [14] applied different tensor recovery algorithms and they did a detailed performance analysis among them. The common problem with these algorithms is that they require computing the SVD of the tensor in every iteration, which is very computationally expensive. In this thesis, we formulate a new objective function for the tensor RPCA problem based on using the truncated and smoothed Schatten-p functions. We refer to our proposed framework as the Truncated-and-Smoothed Robust Tensor Recovery (TSRTR). We solved the proposed objective function using the Augmented Lagrangian Multiplier (ALM) algorithm, which is the trivial algorithm to use with RPCA. We also proposed two other algorithms based on the Accelerated Proximal Gradient (APG) and the Alternating Directions Methods of Multipliers (ADMM).

2.5 Truncated Nuclear Norm

Even though the nuclear norm based algorithms for matrix completion perform well on synthetic data, these algorithms do not perform well in practical applications due to the fact that nuclear norm doesn't accurately approximate the rank function. The rank of a matrix treats all the non-zero singular values equally, while the nuclear norm treats singular values differently. Also, nuclear norm approaches do not converge sometimes due if the underlying matrix does not follow the theoretical bounds such as the coherence property [34].

Instead of modifying all the singular values of the estimated matrix we need to keep the significant singular values unchanged and only modify the smaller singular values which we refer to as the Truncated Nuclear Norm Regularization (TNNR) of the matrix. For a matrix $M \in \mathbb{R}^{m \times n}$ or rank r, TNNR can be represented as $\min(m, n) - r$. The truncated nuclear norm only minimizes the r lower singular values which produces better approximation to the rank function [34], [39]. Now the matrix completion problem can be casted as:

$$\min_{X} \ \|X\|_r$$
$$s.t. \ \|P_\Omega(X) - P_\Omega(M)\|_F^2 \le \varepsilon \tag{2.14}$$

However, $\|.\|_r$ is not convex and hence we have to find a close convex approximation to it. In [34] the authors included a theorem to prove that TNNR can be bounded by a convex function approximation:

$$Tr(AXB^T) \le \sum_{i=1}^{\min(m,n)} \sigma_i(X) \tag{2.15}$$

where $X \in \mathbb{R}^{m\times n}, A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times n}$, $\mathrm{A}A^T = I$ and $BB^T = I$ and I is the identity matrix.

Let the SVD of the matrix $X = U\Sigma V^T$ where $U = (u_1, u_2, \dots\ u_m) \in \mathbb{R}^{m\times m}, V = (v_1, v_2, \dots\ v_n) \in \mathbb{R}^{n\times n}$ and $\Sigma \in \mathbb{R}^{m\times n}$, the equality in (2.15) will hold if we set [34]:

$$A = (u_1, u_2, \dots\ u_r)^T \in \mathbb{R}^{r\times m}\ , B = (v_1, v_2, \dots\ v_r)^T \in \mathbb{R}^{r\times n} \tag{2.16}$$

Thus, the matrix completion optimization function can be represented as:

$$\min_{X} \ \|X\|_* - Tr(AXB^T)$$
$$s.t. \ \|P_\Omega(X) - P_\Omega(M)\|_F^2 \le \varepsilon \tag{2.17}$$

Since the objective function is convex by setting A and B as shown in (2.16), then it can be solved using any convex optimization algorithms. The solution to this function consists of two steps, the first one is to fix X and compute A and B, the next iteration is to estimate the new value of X. An example of this two-step optimization algorithm is shown in Algorithm 2.1 [34].

Similar approach can be applied to the tensor domain for an n -fold tensor, which can be generally represented as [39]:

$$\min_{\mathcal{X}} \sum_{i=1}^{n} \alpha_i \left\|(\mathcal{X})_{(i)}\right\|_r \qquad (2.18)$$

$$s.t.\ P_{\Omega}(\mathcal{X}) = P_{\Omega}(\mathcal{M})$$

where $\alpha_i, 1 \leq i \leq n$ is a weighting parameter for the TNNR of each unfold of the tensor.

Using similar proof to [34], equation (2.18) can be rewritten as:

$$\min_{\mathcal{X}} \sum_{i=1}^{n} (\alpha_i \left\|(\mathcal{X})_{(i)}\right\|_r - \alpha_i A_i (\mathcal{X})_{(i)} {B_i}^T) \qquad (2.19)$$

$$s.t.\ P_{\Omega}(\mathcal{X}) = P_{\Omega}(\mathcal{M})$$

Algorithm 2.1: **TNNR Iterative Scheme Matrix Completion**

```
Input:     $P_{\Omega}(M)$, tolerance $\epsilon$
Steps:     initialize $X^1 = P_{\Omega}(M)$
           Repeat
           1: Given $X_l$
              $[U_l, \Sigma_l, V_l] = SVD(X_l)$
              where $U_l = (u_1, u_2, \dots u_m) \in \mathbb{R}^{m\times m}$
              $V_l = (v_1, v_2, \dots v_n) \in \mathbb{R}^{n\times n}$
              Compute $A_l = (u_1, u_2, \dots u_r)^T \in \mathbb{R}^{r\times m}$
                      $B_l = (v_1, v_2, \dots v_r)^T \in \mathbb{R}^{r\times n}$
           2: Solve
                 $X_{l+1} = \min_{X_l} \|X_l\|_* - Tr\left(A_l X_l {B_l}^T\right)$
                 $s.t.\ \|P_{\Omega}(X\ ) - P_{\Omega}(M)\|_F^2 \leq \varepsilon$
              Until $\|X_{l+1} - X_l\|_{\boldsymbol{F}} \leq \epsilon$
Output:    $X$
```

Chapter 3

Proposed Algorithms for Low Rank Models

3.1 **Matrix Completion Algorithms using Smooth Rank Approximation** [68]

In this work, we are interested in solving the matrix completion problem using the reweighted rank approximation minimization formulation proposed in [35]. To that end, we set the exponent parameter $p = 1$ in the smooth approximation function (2.9). Further, and to make the solution to (2.10) more robust to noise, we also relax the equality constraints into an inequality constraint. Hence, we cast the matrix completion problem as follows:

$$\begin{aligned} &\min_X Tr(X^T X + \gamma I)^{\frac{1}{2}} \\ &s.t. \|P_\Omega(X\text{-}M)\|_F^2 \leq \varepsilon \end{aligned} \tag{3.1}$$

where $\varepsilon > 0$. The Lagrangian equivalent of (3.1) is given by:

$$\min_X Tr(X^T X + \gamma I)^{1/2} + \frac{\lambda}{2}\|P_\Omega(X - M)\|_F^2 \tag{3.2}$$

where λ is the lagrange multiplier.

More importantly, we develop two algorithms to solve the reweighted-rank-approximation matrix completion problem in (3.1). These algorithms are described in the following two subsections.

3.1.1. Accelerated Proximal Gradient Algorithm for Reweighted Rank Approximation

The APG algorithm solves problems of the form [69]:

$$F(x) = f(x) + g(x) \tag{3.3}$$

And the APG solution to (3.2) is given by [69]:

$$Q_\tau(x,z) = f(z) + g(x) + \langle \nabla f(z), x - z \rangle + \frac{\tau}{2} \|x - z\|_F^2 \tag{3.4}$$

Completing the sum of squares and simplifying (3.4) give:

$$Q_\tau(x,z) = f(z) + \frac{\tau}{2} \left\| x - z + \frac{1}{\tau} \nabla f(z) \right\|_F^2 - \frac{1}{2\tau} \|\nabla f(z)\|_F^2 + g(x) \tag{3.5}$$

Now, we cast the reweighted-rank-approximation matrix completion considering the Lagrangian dual problem (3.1), by selecting:

$$g(X) = Tr(X^T X + \gamma I)^{\frac{1}{2}}, f(X) = \frac{\lambda}{2} \|P_\Omega(X - M)\|_F^2 \tag{3.6}$$

Substitute from (3.6) into (3.5) we get:

$$\begin{aligned} \min_{X,Z} Q_\tau(X,Z) = \frac{\lambda}{2} \|P_\Omega(Z - M)\|_F^2 - \frac{\lambda}{2\tau} \|P_\Omega(Z - M)\|_F^2 + \frac{\tau}{2} \left\| X - Z + \frac{\lambda}{\tau} P_\Omega(Z - M) \right\|_F^2 \\ + Tr(X^T X + \gamma I)^{1/2} \end{aligned} \tag{3.7}$$

Using a similar idea to [53] for the nuclear norm minimization case, we introduce another matrix Y.

$$Y^{k+1} = Z^k - \frac{\lambda}{\tau} (P_\Omega(Z^k - M)) \tag{3.8}$$

Now (3.7) becomes:

Algorithm 3.1: **SRAMC-APG**

```
Input:    $P_\Omega(M)$, $\lambda$, $\tau$
Steps:    1: initialize $X^1$=$X^0$=$Y^1$=$Z^1$ $= P_\Omega(M)$, $t^1$=$t^0$=1
          2: while  Not converged
               Evaluate  $Z^{k+1}$ using (2.12)
               Evaluate $Y^{k+1}$ using (2.8)
               Evaluate  $W$ using (2.11)
               Evaluate  $X^{k+1}$ using (2.10)
               Update $t^{k+1}$   using (2.13)
             End While
Output:   $X$
```

$$\min_{X,Y,Z} Q_\tau(X,Y,Z\ )$$

$$= \frac{\lambda}{2}\|P_\Omega(Z-M)\|_F^2 + \frac{\tau}{2}\|X-Y\ \|_F^2 - \frac{\lambda}{2\tau}\|P_\Omega(Z-M)\|_F^2 + Tr(X^TX+\gamma I)^{1/2} \tag{3.9}$$

We can minimize (3.9) using Block Coordinate Descent (BCD), we fix *Z,Y* ; then we differentiate and solve for *X*.

$$X^{k+1} = Y^{k+1} - \tau X^k W \tag{3.10}$$

where W is given by:

$$W = (X^TX+\gamma I)^{-1/2} \tag{3.11}$$

As in [53] Z^k can be found by:

$$Z^{k+1} = X^k + \frac{t^{k-1} - 1}{t^k}(X^k - X^{k-1}) \tag{3.12}$$

And update t^{k+1} using:

$$t^{k+1} = \frac{1 + \sqrt{1 + 4(t^k)^2}}{2} \tag{3.13}$$

Algorithm 3.1 shows the final procedure of our Reweighted Rank Approximation Matrix Completion with APG (SRAMC-APG).

3.1.2. Alternating Direction Method of Multipliers for Reweighted Rank Approximation

ADM solves general problems of the form [64]:

$$\begin{gathered} \min_{x \in \mathbb{R}^q, y \in \mathbb{R}^m} f(x) + g(y) \\ s.t.\ x \in \mathbb{C}_x, y \in \mathbb{C}_y, Gx = y \end{gathered} \tag{3.14}$$

where, $\mathbb{C}_x \subset \mathbb{R}^q, \mathbb{C}_y \subset \mathbb{R}^m$ and $G \in \mathbb{R}^{q*m}$

The Lagrangian dual problem of (2.14) is given by [52], [64]:

$$Q(X, Y, L) = f(X) + g(Y) + \langle L^T, GX - Y \rangle + \frac{\tau}{2}\|GX - Y\|_F^2 \tag{3.15}$$

where $L \in \mathbb{R}^m$ is the Lagrange multipliers matrix.

In order to formulate (3.2) as an ADM problem, we need to split the function into two variables.

$$\begin{gathered} \min_X Tr(X^T X + \gamma I)^{\frac{1}{2}} + \frac{\lambda}{2}\|P_\Omega(Y - M)\|_F^2 \\ s.t.\ X = Y \end{gathered} \tag{3.16}$$

```
Algorithm 3.2:    SRAMC-ADM
Input:    $P_{\Omega}(M)$, $\lambda$, $\tau$
Steps:    1: initialize $X^1=Y^1=P_{\Omega}(M)$, $L^1 = 0$
          2: while  Not converged
              Evaluate  $Y^{k+1}$ using (2.20)
              Evaluate  $W$ using (2.11)
              Evaluate  $X^{k+1}$ using (2.19)
              Evaluate  $L^{k+1}$ using (22)
           End While
Output:   $X$
```

The Lagrangian dual problem of (3.16) after setting G to the identity matrix is given by:

$$\min_{X,Y,L} Q(X,Y,L) = Tr(X^T X + \gamma I)^{\frac{1}{2}} + \frac{\lambda}{2}\|P_{\Omega}(Y - M)\|_F^2 + \langle L^T, X - Y\rangle + \frac{\tau}{2}\|X - Y\|_F^2 \quad (3.17)$$

Using BCD, we reduce (3.17) for X by fixing Y and L. After completing the sum of squares and neglecting the constants we get:

$$\min_{X} Q(X) = Tr(X^T X + \gamma I)^{\frac{1}{2}} + \frac{\tau}{2}\left\|X - Y + \frac{1}{\tau}L\right\|_F^2 \quad (3.18)$$

Differentiating and solving (3.18) for X:

$$X^{k+1} = Y^{k+1} - \tau(X^k W + L) \quad (3.19)$$

where W is defined as in (3.11)

Next, we reduce (3.17) for Y by fixing X and L and neglecting the constant terms; then we differentiate and solve for Y.

$$Y^{k+1} = X^k + \frac{1}{\tau}(L^{k} - \lambda\, P_{\Omega}(Y^k - M)) \tag{3.20}$$

Finally, we simplify (3.17) by completing the sum of squares and solve for L by fixing X and Y and neglecting the constant terms.

$$\min_{L} Q(L) = \frac{\tau}{2}\left\|X - Y + \frac{1}{\tau}L\right\|_F^2 - \frac{1}{2\tau}\|L\|_F^2 \tag{3.21}$$

Differentiating and solving for L:

$$L^{k+1} = L^{k} + \tau(X^{k+1} - Y^{k+1}) \tag{3.22}$$

The implementation of the Reweighted Rank Approximation Matrix Completion with ADM (SRAMC-ADM) is shown in Algorithm 3.2.

3.2 Smooth Rank Approximation Tensor Completion [67]

In this work, we are interested in solving the matrix completion problem using the reweighted rank approximation minimization formulation proposed in [35].

In this thesis we are interested in using a generalized version of the smooth Schatten-p function to solve the tensor completion problem, and hence we need to redefine (2.9) to the higher dimensional space.

Definition 3.1: The smooth Schatten-p function for an N-dimensional signal is given by:

$$f_p(\mathcal{X}) = \frac{1}{N}\sum_{i=1}^{N} Tr((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{p}{2}} \tag{3.23}$$

One can verify the validity of Defintion 2.1 for the 2-dimensional Schatten-p case ($N = 2$) by observing that the trace of the matrix and its transpose are the same; and hence, (3.23) reduces to (2.9).

Now the tensor completion problem based on a smooth rank approximation function can be casted as:

$$\min_{\mathcal{X}} \frac{1}{N}\sum_{i=1}^{N} Tr((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{p}{2}} \tag{3.24}$$

$$s.t.\ P_{\Omega}(\mathcal{X}) = P_{\Omega}(\mathcal{M})$$

Next we derive the ALM optimization method to solve the tensor completion problem in (3.24). Using similar idea for the matrix case in [63], we can rewrite (3.24) in the tensor domain as:

$$\begin{gathered}\min_{\mathcal{X}} \frac{1}{N}\sum_{i=1}^{N} Tr((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{p}{2}} \\ s.t.\ \mathcal{X} + \mathcal{E} = P_{\Omega}(\mathcal{M}), P_{\Omega}(\mathcal{E}) = 0\end{gathered} \tag{3.25}$$

Here, $\mathcal{X}$ represents the non-observed entries of the original tensor $\mathcal{M}$. The partial augmented Lagrange function for (3.25) is given by:

$$\begin{aligned}L(\mathcal{X}, \mathcal{E}, \mathcal{Y}, \mu) = \sum_{i=1}^{N} \alpha_i Tr((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{p}{2}} + \langle \mathcal{Y}, P_{\Omega}(\mathcal{M}) - \mathcal{X} - \mathcal{E} \rangle \\ + \frac{\mu}{2}\|P_{\Omega}(\mathcal{M}) - \mathcal{X} - \mathcal{E}\|_F^2\end{aligned} \tag{3.26}$$

where $\mathcal{Y}$ is the Lagrange multiplier tensor.

In this section we state the solution for (3.26) with respect to each variable ($\mathcal{X}$, $\mathcal{E}$ and $\mathcal{Y}$) as three lemmas; the proofs are presented separately in section 3.5.

Lemma 3.1: The optimal solution for (3.26) with respect to $\mathcal{X}$ is given by:

$$\mathcal{X}^{k+1} = P_{\Omega}(\mathcal{M}) + \frac{1}{\mu}\mathcal{Y}^{k} - \left[\mathcal{E}^{k} + \frac{\frac{1}{\mu}\sum_{i=1}^{N}\beta_i\ \mathcal{Z}_i^{k+1}}{\sum_{i=1}^{N}\beta_i}\right] \tag{3.27}$$

where β is a control parameter and $\mathcal{Z}_i^{k+1}$ is given by:

Algorithm 3.3: SRATC

Input: $\mu > 0, \beta \in \mathbb{R}^N, P_\Omega(\mathcal{M})$

Steps:

```
1: initialize $\mathcal{X}^1 = P_\Omega(\mathcal{M})$, $\mathcal{Y}^1 = \mathcal{E}^1 = 0$
2: while Not converged
  Do: For each dimension $i$
          Evaluate $W_i$ using (2.29)
          Evaluate $\mathcal{Z}_i^{k+1}$ using (2.28)
      End For
      Evaluate $\mathcal{X}^{k+1}$ using (2.27)
      Evaluate $\mathcal{E}^{k+1}$ using (2.30)
      Evaluate $\mathcal{Y}^{k+1}$ using (2.31)
  End While
```

Output: $\mathcal{X}, \mathcal{E}$

$$\mathcal{Z}_i^{k+1} = fold\left(\alpha_i(\mathcal{X}^k)_{(i)} W_i\right)_{(i)} \tag{3.28}$$

W_i is defined as:

$$W_i = ((\mathcal{X}^k)_{(i)}^T (\mathcal{X}^k)_{(i)} + \gamma I)^{1-\frac{p}{2}} \tag{3.29}$$

Lemma 3.2: The optimal solution for (3.26) with respect to $\mathcal{E}$ is given by:

$$P_{\bar{\Omega}}(\mathcal{E}^{k+1}) = P_{\bar{\Omega}}(\mathcal{M} + \frac{1}{\mu}\mathcal{Y}^{k} - \mathcal{X}^{k+1})$$
$$P_{\Omega}(\mathcal{E}^{k+1}) = 0 \tag{3.30}$$

Lemma 3.3: The optimal solution for (3.26) with respect to $\mathcal{Y}$ is given by:

$$\mathcal{Y}^{k+1} = \mathcal{Y}^{k} + \mu(P_{\Omega}(\mathcal{M}) - \mathcal{X}^{k+1} - \mathcal{E}^{k+1}) \tag{3.31}$$

Benefitting from the smooth Schatten-P function formula we used a code optimization idea in order to reduce the computation complexity. Since the trace of the matrix $(\mathcal{X}^{k})_{(i)} \in \mathbb{R}^{p\times q}$ and its transpose are the same; then while computing W_i we used the transpose of $(\mathcal{X}^{k})_{(i)}$ if $q > p$ and we also used the term $((\mathcal{X}^{k})_{(i)}W_i)^T$ in order to maintain the original dimensions of the unfolded tensor. The proposed Smooth Rank Approximation Tensor Completion (SRATC) algorithm is presented in algorithm 3.3.

3.3 **Smooth Rank Approximation for Robust Tensor Recovery** [70]

This work is motivated by recent works [10][11], which show that matrix and tensor completion using the truncated nuclear norm provides a significant performance improvement over the traditional nuclear norm minimization framework. This imporvement can be attributed to the observatrion that minimizing the truncated nuclear norm keeps the high singular values (related to the actual low rank signal) intact, and at the same time it minimizes the lower singular values (related to noise or sparse part).

Some challenges associated with applying the objective function in [11] for the tensor recovery problem are: First, it needs computing the singular value decomposition of the tensor (HoSVD) two times in each step, which is very computationally expensive. Second, the nature of the RPCA problem requires adding a minimization function to the ℓ_1 term to estimate the sparse part of the observed tensor, which makes the derivation of the optimization algorithm quite challenging.

In this thesis, we utilize the generalized multi-dimensional smoothed Schatten-p in [8] which was used to achieve lower execution time in tensor completion. The general objective function that we consider is:

$$\min_{\mathcal{X},\mathcal{E}} \sum_{i=1}^{m} \left[Tr\left[\left((\mathcal{X})_{(i)}^{T} (\mathcal{X})_{(i)} + \gamma I \right)^{\frac{1}{2}} \right] - \beta_i Tr\left(A_i (\mathcal{X})_{(i)} {B_i}^{T} \right) + \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 \right] \tag{3.32}$$

$$s.t.\ \mathcal{M} = \mathcal{X} + \mathcal{E}$$

where $\mathcal{X}$ is the low rank tensor to be estimated and $\mathcal{E}$ is the sparse tensor and $\mathcal{M}$ is the observed tensor. If the SVD decomposition of a matrix X is described as $X = U\Sigma V^T$, then $A_i = (\underline{u}_1 \ldots \underline{u}_r)^T, B_i = (\underline{v}_1 \ldots \underline{v}_r)^T$.

Since the trace function is linear and we are interested in minimizing the nuclear norm function, we set $p = 1$ in the Schatten-p function. Now Equation (3.32) can be casted as:

$$\min_{\mathcal{X},\mathcal{E}} \sum_{i=1}^{m} \left[Tr\left[\left((\mathcal{X})_{(i)}^{T} (\mathcal{X})_{(i)} + \gamma I \right)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{X})_{(i)} {B_i}^{T} \right) \right] + \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 \right] \tag{3.33}$$

$$s.t.\ \mathcal{M} = \mathcal{X} + \mathcal{E}$$

In this thesis, we propose three different solutions to (3.33) using the Augmented Lagrangian Multiplier (ALM), Accelerated Proximal Gradient (APG) and Alternating Direction Methods of Multipliers (ADMM) optimization algorithms. We also provide analytical derivations and detailed proofs for key expressions of all three algorithms.

3.3.1. The Proposed ALM Solution

The ALM is an optimization algorithm that is used to solve constrained optimization problems due to its Q-linear solution to the problem and because of the availability of an exact solution to the ALM [7], [12]. In general, ALM solves problems of the form:

$$\begin{gathered} \min f(x) + g(y) \\ \text{s.t.}\, Mx = y \end{gathered} \tag{3.34}$$

Here x and y are the variables to be estimated from the optimization problem and M is the sampling operator.

Using the Lagrangian method for solving optimization problems by converting the condition into a term in the objective function, the general ALM solution to (3.34) can be represented as:

$$\min f(x) + g(y) + < l, Mx - y > + \frac{\lambda}{2} \|Mx - y\|_F^2 \tag{3.35}$$

where l is the Lagrange multiplier vector.

To solve (3.35) in the multi-dimensional tensor domain, we need to change all variables to tensors and we need to case our objective function (3.33) so that it can be solved using ALM, to this extent, we need to select the variables as shown below:

$$\begin{gathered} f(\mathcal{X}) = \sum_{i=1}^{m} \left[Tr \left[((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{X})_{(i)} {B_i}^T \right) \right] \right] \\ g(\mathcal{E}) = \left\| (\mathcal{E})_{(i)} \right\|_1 \end{gathered} \tag{3.36}$$

After substituting the variables in the general ALM solution (3.35), The ALM solution to the proposed framework can be represented as:

$$L(\mathcal{X}, \mathcal{E}, \mathcal{Y}, \mu) = \sum_{i=1}^{m} \left[\alpha_i Tr\left[((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{X})_{(i)} {B_i}^{T}\right)\right] + \lambda \left\|(\mathcal{E})_{(i)}\right\|_1\right] + \langle \mathcal{Y}, \mathcal{M} - \mathcal{X} - \mathcal{E} \rangle + \frac{\mu}{2}\|\mathcal{M} - \mathcal{X} - \mathcal{E}\|_F^2 \tag{3.37}$$

where $\mathcal{Y}$ is the Lagrange multiplier tensor.

Lemma 3.4: For a tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ that is formed by the superposition of low rank and sparse tensors, $\mathcal{X}$ and $\mathcal{E}$ respectively; the optimal solution for (3.37) with respect to $\mathcal{X}$ is given by:

$$\mathcal{X}^{k+1} = \left[\sum_{i=1}^{m} \alpha_i \, fold\left(Z_i - \frac{1}{\mu}\left(\mathcal{X}_i{}^{k}\right)_{(i)} W_i + \frac{\beta_i}{\mu} A_i^T B_i\right)_{(i)}\right] / \sum_{i=1}^{n} \alpha_i \tag{3.38}$$

where W_i is represented by:

$$W_i = ((\mathcal{X}^{k})_{(i)}^{T}(\mathcal{X}^{k})_{(i)} + \gamma I)^{-\frac{1}{2}} \tag{3.39}$$

and $\mathcal{Z}_i$ is represented by:

$$\mathcal{Z}_i = (\mathcal{M} - \mathcal{E}^{k} + \frac{1}{\mu}\mathcal{Y}^{k})_{(i)} \tag{3.40}$$

Lemma 3.5: For a tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ that is formed by the superposition of low rank and sparse tensors, $\mathcal{X}$ and $\mathcal{E}$ respectively; the optimal solution for (3.37) with respect to $\mathcal{E}$ is given by:

$$\mathcal{E}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\mu}}(\mathcal{M} - \mathcal{X}^{k+1} + \frac{1}{\mu}\mathcal{Y}^{k}) \tag{3.41}$$

The optimal solution for (3.37) with respect to $\mathcal{Y}$ is given by:

$$\mathcal{Y}^{k+1} = \mathcal{Y}^{k} + \mu(\mathcal{M} - \mathcal{X}^{k+1} - \mathcal{E}^{k+1}) \tag{3.42}$$

The proofs for lemmas 3.4 and 3.5 are shown in section 3.5. The proposed TSRTR-ALM algorithm is presented in Algorithm 1.

3.3.2. The Proposed APG Solution

The proximal gradient algorithms solve unconstrained optimization problems. It works by evaluating the proximal operator of the function. The accelerated version of these algorithms is achieved by adding a weighted difference of the estimated iterative solution which helps speeding up the convergence of the algorithm. APG solves problems of the form [13]:

$$F(x) = f(z) + g(x) \tag{3.43}$$

and the APG solution to (3.43) is given by [13]:

$$Q_\tau(x,z) = f(z) + g(x) + \langle \nabla f(z), x - z \rangle + \frac{\tau}{2}\|x - z\|_F^2 \tag{3.44}$$

Completing the sum of squares and simplifying (3.44) give:

$$Q_\tau(x,z) = f(z) + \frac{\tau}{2}\left\|x - z + \frac{1}{\tau}\nabla f(z)\right\|_F^2 - \frac{1}{2\tau}\|\nabla f(z)\|_F^2 + g(x) \tag{3.45}$$

The objective function (3.45) solves for a single variable (x), while our objective function (3.32) consists of two variables ($\mathcal{X}$, $\mathcal{E}$). To solve this, we extend the idea in [5], which was implemented on matrix robust PCA with nuclear norm minimization, to the multi-dimensional tensor domain by defining a pair of two-tensor variables $\mathcal{S} = (\mathcal{S}_\mathcal{X} = \mathcal{X}, \mathcal{S}_\mathcal{E} =$

$\mathcal{E}) \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m} \times \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ and $\mathcal{Z} = (\mathcal{Z}_{\mathcal{X}}, \mathcal{Z}_{\mathcal{E}}) \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m} \times \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ instead of the variables (x and z). Now, we can solve the Truncated Robust Tensor Recovery problem (3.32), by selecting:

$$g(\mathcal{S}) = \sum_{i=1}^{m} \left[Tr\left[((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{X})_{(i)} B_i{}^T \right) \right] + \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 \right]$$
$$f(\mathcal{Z}) = \frac{\mu}{2} \| \mathcal{M} - \mathcal{Z}_{\mathcal{X}} - \mathcal{Z}_{\mathcal{E}} \|_F^2 \tag{3.46}$$

Substitute from (3.46) into (3.45) we get:

$$\min_{\mathcal{S}, \mathcal{Z}} Q_\tau(\mathcal{S}, \mathcal{Z}) = \frac{\mu}{2} \| \mathcal{M} - \mathcal{Z}_{\mathcal{X}} - \mathcal{Z}_{\mathcal{E}} \|_F^2$$
$$+ \sum_{i=1}^{m} \left[\left(Tr(\mathcal{S}_{\mathcal{X}})_{(i)}^T (\mathcal{S}_{\mathcal{X}})_{(i)} + \gamma I \right)^{\frac{p}{2}} - \beta_i Tr \left(A_i (\mathcal{S}_{\mathcal{X}})_{(i)} B_i{}^T \right) \right. \tag{3.47}$$
$$\left. + \lambda \left\| (\mathcal{S}_{\mathcal{E}})_{(i)} \right\|_1 \right]$$

```
Algorithm 3.4:            TSRTR-ALM
Input:    $M$
Steps:    1:initialize, $\mathcal{X}^0 = \mathcal{M}, \Upsilon^0 = \mathcal{E}^0 = 0, \lambda > 0, \beta_i > 0, \alpha_i > 0, \eta > 1, \mu$
          2: while  Not converged
            For each dimension $i$
              $[U, \Sigma, V] = HoSVD((\mathcal{X}_i^k)_{(i)})$
                $A_i = (\underline{u}_1 \dots \underline{u}_{r_i})^T,\ B_i = (\underline{v}_1 \dots \underline{v}_{r_i})^T$
                $W_i = ((\mathcal{X}^k)_{(i)}^T (\mathcal{X}^k)_{(i)} + \gamma I)^{-\frac{1}{2}}$
                $Z_i = (\mathcal{M} - \mathcal{E}^k + \frac{1}{\mu}\Upsilon^k)_{(i)}$
              $(\mathcal{X}^{k+1})_{(i)} = \alpha_i\, fold\left(Z_i - \frac{1}{\mu}(\mathcal{X}_i^k)_{(i)} W_i + \frac{\beta_i}{\mu} A_i^T B_i\right)_{(i)}$
            End For
            $\mathcal{X}^{k+1} = \sum_{i=1}^{m} (\mathcal{X}^{k+1})_{(i)} / \sum_{i=1}^{n} \alpha_i$
          $\mathcal{E}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\mu}}(\mathcal{M} - \mathcal{X}^{k+1} + \frac{1}{\mu}\Upsilon^k)$
          $\Upsilon^{k+1} = \Upsilon^k + \mu(\mathcal{M} - \mathcal{X}^{k+1} - \mathcal{E}^{k+1})$
           $\mu = \eta * \mu$
          End While
Output:    $\mathcal{X}^*, \mathcal{E}^*$
```

To simplify the mathematical solution, we introduce another tensor pair $\Upsilon = (\Upsilon_{\mathcal{X}}, \Upsilon_{\mathcal{E}}) \in \mathbb{R}^{n_1 \times n_2 \dots \times n_m} \times \mathbb{R}^{n_1 \times n_2 \dots \times n_m}$. Similar idea was used in [14] for the nuclear norm minimization case and we define it for each member of the tensor pair.

$$\mathcal{Y}_{\mathcal{X}}^{k+1} = \mathcal{Z}_{\mathcal{X}}^{k} - \frac{\mu}{\tau}(\mathcal{M} - \mathcal{Z}_{\mathcal{X}})$$

$$\mathcal{Y}_{\mathcal{E}}^{k+1} = \mathcal{Z}_{\mathcal{E}}^{k} - \frac{\lambda}{\tau}(\mathcal{M} - \mathcal{Z}_{\mathcal{E}}) \qquad (3.48)$$

$$\text{s.t. } \mathcal{S} = \mathcal{Y}$$

Now (3.48) becomes:

$$\min_{\mathcal{S},\mathcal{Y},\mathcal{Z}} Q_\tau(\mathcal{S},\mathcal{Y},\mathcal{Z})$$

$$= \frac{\mu}{2}\|\mathcal{M} - \mathcal{Z}_{\mathcal{X}} - \mathcal{Z}_{\mathcal{E}}\|_F^2 - \frac{\lambda}{2\tau}\|\mathcal{M} - \mathcal{Z}_{\mathcal{X}} - \mathcal{Z}_{\mathcal{E}}\|_F^2 + \frac{\tau}{2}\|\mathcal{S} - \mathcal{Y}\|_F^2$$
$$+ \sum_{i=1}^{m}\left[\alpha_i Tr\left[((\mathcal{S}_{\mathcal{X}})_{(i)}^T(\mathcal{S}_{\mathcal{X}})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i\left(A_i(\mathcal{S}_{\mathcal{X}})_{(i)}B_i{}^T\right)\right] \qquad (3.49)$$
$$+ \lambda\left\|(\mathcal{E})_{(i)}\right\|_1\right]$$

We use the default APG update rule proposed in [14] for each variable [14]:

$$\mathcal{Z}_{\mathcal{X}}^{k+1} = \mathcal{S}_{\mathcal{X}}^{k} + \frac{t^{k-1}-1}{t^k}(\mathcal{S}_{\mathcal{X}}^{k} - \mathcal{S}_{\mathcal{X}}^{k-1})$$
$$(3.50)$$
$$\mathcal{Z}_{\mathcal{E}}^{k+1} = \mathcal{S}_{\mathcal{E}}^{k} + \frac{t^{k-1}-1}{t^k}(\mathcal{S}_{\mathcal{E}}^{k} - \mathcal{S}_{\mathcal{E}}^{k-1})$$

And update t^{k+1} using:

$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2} \qquad (3.51)$$

Lemma 3.6: For a tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ that is formed by the superposition of low rank and sparse tensors, $\mathcal{X}$ and $\mathcal{E}$ respectively; The optimal solution for the low rank ($\mathcal{S}_{\mathcal{X}}$) and sparse ($\mathcal{S}_{\mathcal{E}}$) tensors using APG is given by:

$$\mathcal{S}_{\mathcal{X}}^{k+1} = \left[\sum_{i=1}^{m} \alpha_i\, fold\left((\Upsilon_{\mathcal{X}}^{k+1})_{(i)} - \frac{1}{\tau}\left(\mathcal{S}_{\mathcal{X}}^{k}\right)_{(i)} W_i + \frac{\beta_i}{\tau} A_i^T B_i\right)_{(i)}\right] / \sum_{i=1}^{n_m} \alpha_i$$
$$\mathcal{S}_{\mathcal{E}}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\mu}}(\Upsilon_{\mathcal{E}}^{k+1}) \tag{3.52}$$

where W_i is as defined in (3.32).

The proof of lemma 3.6 is shown in section 3.5. Algorithm 2 shows the optimization procedure for the Truncated and Smooth Robust Tensor Recovery using Accelerated Proximal Gradient (TSRTR-APG).

3.3.3. The Proposed ADMM Solution

ADMM is an extension of the ALM algorithm. ADMM solves structured optimization problems of the form [15]:

$$\min_{x \in \mathbb{R}^q, y \in \mathbb{R}^m} f(x) + g(y)$$
$$\text{s.t. } x \in \mathbb{C}_x, y \in \mathbb{C}_y, Gx = y \tag{3.53}$$

where, $\mathbb{C}_x \subset \mathbb{R}^q, \mathbb{C}_y \subset \mathbb{R}^m$ and $G \in \mathbb{R}^{q*m}$

Similar to ALM, the optimization is done by alternatively solving for x and y.

```
Algorithm 2:        TSRTR-APG

Input:    $\mathcal{M}, \mu, \lambda, \tau$
Steps:    1: initialize $\mathcal{S}_{\mathcal{X}}^{1}=\mathcal{S}_{\mathcal{X}}^{0}=\mathcal{S}_{\mathcal{E}}^{1}=\mathcal{S}_{\mathcal{E}}^{0}=0,\ t^{1}=t^{0}=1$
          2: while  Not converged
               $\mathcal{Z}_{\mathcal{X}}^{k+1}=\mathcal{S}_{\mathcal{X}}^{k}+\frac{t^{k-1}-1}{t^{k}}(\mathcal{S}_{\mathcal{X}}^{k}-\mathcal{S}_{\mathcal{X}}^{k-1})$
               $\mathcal{Z}_{\mathcal{E}}^{k+1}=\mathcal{S}_{\mathcal{E}}^{k}+\frac{t^{k-1}-1}{t^{k}}(\mathcal{S}_{\mathcal{E}}^{k}-\mathcal{S}_{\mathcal{E}}^{k-1})$
               $\mathcal{Y}_{\mathcal{X}}^{k+1}=\mathcal{Z}_{\mathcal{X}}^{k}-\frac{\mu}{\tau}(\mathcal{M}-\mathcal{Z}_{\mathcal{X}})$
               $\mathcal{Y}_{\mathcal{E}}^{k+1}=\mathcal{Z}_{\mathcal{E}}^{k}-\frac{\lambda}{\tau}(\mathcal{M}-\mathcal{Z}_{\mathcal{E}})$
               For each fold $i$
          $W_i=((\mathcal{X}^{k})_{(i)}^{T}(\mathcal{X}^{k})_{(i)}+\gamma I)^{-\frac{1}{2}}$
                  $(\mathcal{S}_{\mathcal{X}}^{k+1})_{(i)}=\alpha_i\, fold\left((\mathcal{Y}_{\mathcal{X}}^{k+1})_{(i)}-\frac{1}{\tau}\left(\mathcal{S}_{\mathcal{X}}^{k}\right)_{(i)}W_i+\frac{\beta_i}{\tau}A_i^{T}B_i\right)_{(i)}$
               End for
               $\mathcal{S}_{\mathcal{X}}^{k+1}=\left[\sum_{i=1}^{m}\alpha_i\,(\mathcal{S}_{\mathcal{X}}^{k+1})_{(i)}\right]/\sum_{i=1}^{m}\alpha_i$
               $\mathcal{S}_{\mathcal{E}}^{k+1}=\mathfrak{S}_{\frac{\lambda}{\mu}}(\mathcal{Y}_{\mathcal{E}}^{k+1})$
               $t^{k+1}=\frac{1+\sqrt{1+4(t^{k})^{2}}}{2}$
              End While
Output:   $\mathcal{S}_{\mathcal{X}}^{*}, \mathcal{S}_{\mathcal{E}}^{*}$
```

The Lagrangian dual problem of (3.53) is given by [15], [16]:

$$Q(x, y, L) = f(x) + g(y) + \langle L, Gx - y \rangle + \frac{\tau}{2}\|Gx - y\|_F^2 \tag{3.54}$$

where $L \in \mathbb{R}^m$ is the Lagrange multipliers matrix.

Due to the separable structure of ADMM, it is considered an improvement over ALM in solving convex programming problems [17], [18].

Since our cost function (3.32) already has two variables, then we can select the f and g functions as follow:

$$f(\mathcal{X}) = \sum_{i=1}^{m}\left[\alpha_i Tr\left[((\mathcal{X})_{(i)}^T(\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i\left(A_i(\mathcal{X})_{(i)}B_i{}^T\right)\right]\right]$$

$$g(\mathcal{E}) = \sum_{i=1}^{m} \left\|(\mathcal{E})_{(i)}\right\|_1 \tag{3.55}$$

Using the functions defined in (3.55) the ADMM optimization function can be casted as:

$$\min_{\mathcal{X},\mathcal{E},\mathcal{L}} Q(\mathcal{X}, \mathcal{E}, \mathcal{L}) = \sum_{i=1}^{m}\left[\alpha_i Tr\left[((\mathcal{X})_{(i)}^T(\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i\left(A_i(\mathcal{X})_{(i)}B_i{}^T\right)\right] + \lambda\left\|(\mathcal{E})_{(i)}\right\|_1\right] + \langle \mathcal{L}, \mathcal{X} + \mathcal{E} - \mathcal{M} \rangle + \frac{\tau}{2}\|\mathcal{X} + \mathcal{E} - \mathcal{M}\|_F^2 \tag{3.56}$$

Lemma 3.7: The optimal solution for the low rank ($\mathcal{X}$) and sparse ($\mathcal{E}$) and the Lagrange multiplier ($\mathcal{L}$) tensors is given by:

```
Algorithm 3:          TSRTR-ADMM
Input:      $\mathcal{M}, \mu, \lambda, \tau$
Steps:      1: initialize $\mathcal{X}^0 = \mathcal{E}^0 = 0, L^1 = 0$
            2: while Not converged
                 For each fold $i$
                   $W_i = ((\mathcal{X}^k)^T_{(i)}(\mathcal{X}^k)_{(i)} + \gamma I)^{-\frac{1}{2}}$
                   $(\mathcal{X}^{k+1})_{(i)} = (\mathcal{E}^{k+1})_{(i)} - \frac{1}{\tau}\left(\mathcal{X}_i^k\right)_{(i)} W_i + \frac{\beta_i}{\tau} A_i^T B_i$
                 End for
                 $\mathcal{X}^{k+1} = \mathcal{M} - \frac{1}{\tau}\left[\sum_{i=1}^{m} \alpha_i \, (\mathcal{S}_{\mathcal{X}}^{k+1})_{(i)}\right] / \sum_{i=1}^{n_m} \alpha_i + \frac{1}{\tau}\mathcal{L}$
                 $\mathcal{E}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\tau}}(\mathcal{M} - \mathcal{X}^k + \frac{1}{\tau}\mathcal{L})$
                 $\mathcal{L}^{k+1} = \mathcal{L}^k - \tau(\mathcal{X}^{k+1} + \mathcal{E}^{k+1} - \mathcal{M})$
            End While
Output:     $\mathcal{X}^*, \mathcal{E}^*$
```

$$\mathcal{X}^{k+1} = \mathcal{M} - \frac{1}{\tau}\left[\sum_{i=1}^{m} \alpha_i \, fold\begin{pmatrix} (\mathcal{E}^{k+1})_{(i)} - \frac{1}{\tau}\left(\mathcal{X}_i^k\right)_{(i)} W_i + \\ \frac{\beta_i}{\tau} A_i^T B_i \end{pmatrix}_{(i)}\right] / \sum_{i=1}^{m} \alpha_i + \frac{1}{\tau}\mathcal{L}$$

$$\mathcal{E}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\tau}}(\mathcal{M} - \mathcal{X}^k + \frac{1}{\tau}\mathcal{L}) \qquad (3.57)$$

$$\mathcal{L}^{k+1} = \mathcal{L}^k - \tau(\mathcal{X}^{k+1} + \mathcal{E}^{k+1} - \mathcal{M})$$

where W_i is as defined in (3.32).

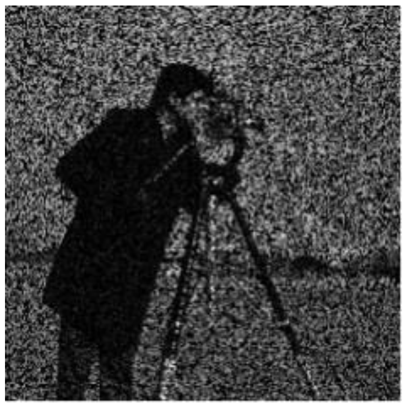
The proof of lemma 3.7 is shown in section 3.5. The implementation of the Truncated and Smooth Robust Tensor Recovery using with ADMM (TSRTR-ADMM) is shown in Algorithm 2.

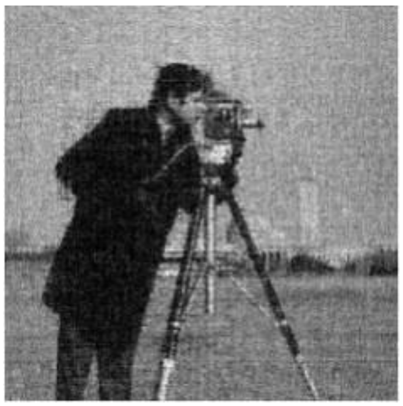
3.4 Experimental Results

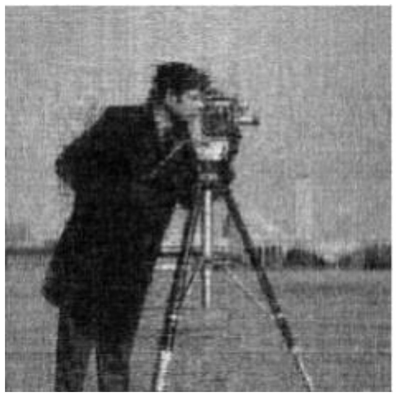
3.4.1. Matrix Completion

In this work we compare our algorithms against each other and against IRLS-1. For the sake of fair comparison, we ran all algorithms using MATLAB R2012b on a computer with 3.3 GHz i-5 CPU and 4GB Memory. We also set the same maximum number of iterations to 1000 for all algorithms. Since IRLS-1 [35] has shown performance and execution time improvements over the matrix completion algorithms with nuclear norm minimization, we will consider comparing our algorithms to each other and to the IRLS-1 only. Figure 3.1 shows the results of recovering the cameraman image (rank 80) with 50% observed entries and additive Gaussian noise (zero mean and 0.007 variance). The results show that SRAMC-ADM achieves the highest SNR followed by SRAMC-APG and both are higher than IRLS-1 algorithm. The reason is because we used the relaxed equality constraints for the smoothed rank approximation algorithms and as a result we minimized the noise variance and reduce the noise effect. Figure 3.2 shows the convergence rate for the algorithms using the same image in Figure 3.1. It shows that both SRAMC-ADM and IRLS-1 need more iterations to converge to the final SNR, while SRAMC-APG converges within less iterations.

(a)


(b)


(c) 14.37


(d) 15.16

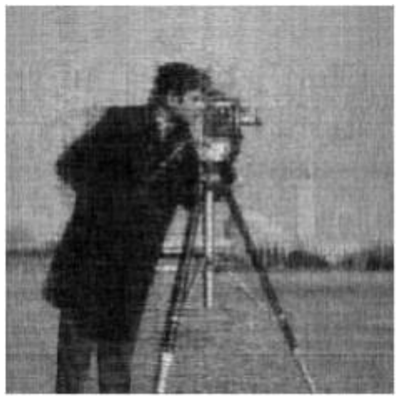

(e) 15.44

Figure 3.1: (a) Original image (rank 80) (b) The observed noisy samples, (c-e) Inpainted image and the corresponding SNR (dB) using (c) ILRS-1, (d) SRAMC-APG, (e) SRAMC-ADM

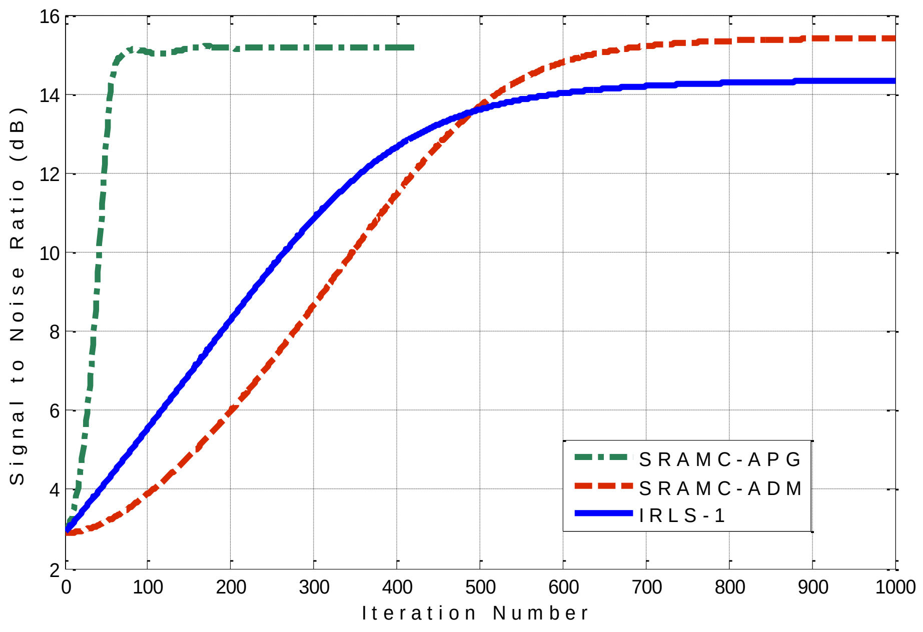


Figure 3.2: Iterations versus signal to noise ratio for cameraman image with 50% observed entries.

Figure 3.3 shows a comparison for the SNR versus the noise variance for all three algorithms using the same cameraman image with 50% random observed noisy entries. From Figure 3.3 we notice that when the noise level is close to zero, there is a slight difference in performance between all algorithms; while once the noise increases we note that SRAMC-ADM achieves the best performance followed by SRAMC-APG and then IRLS-1.

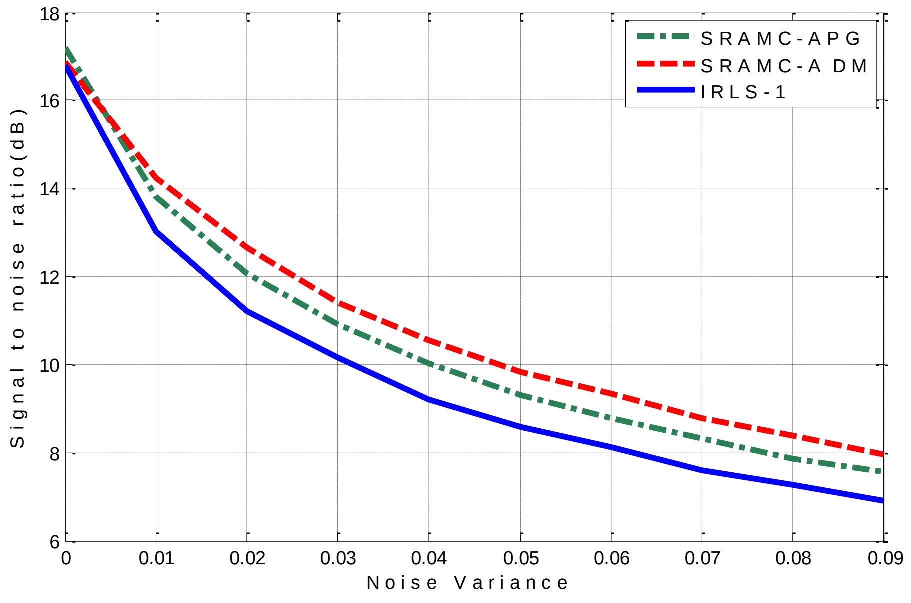


Figure 3.3: SNR versus the noise variance using cameraman image with 50% observed entries.

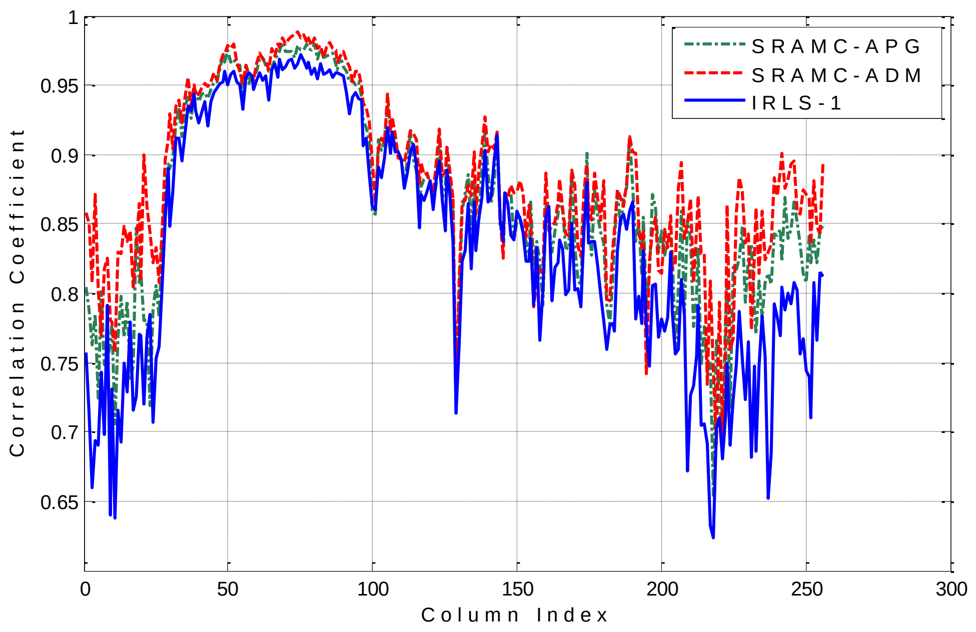


Figure 3.4: Columns correlation coefficient for the recovered cameraman image from 50% observed noisy entries.

Table 3.1: Comparison of average execution time vs. SNR for matrix completion algorithms using a set of 12 grayscale images.

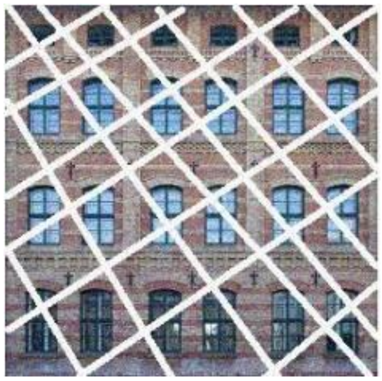
Missing	Method	SNR (dB)	Time (Sec)
30%	IRLS-1	15.43	25
	SRAMC-ADM	**17.62**	20
	SRAMC-APG	16.74	**10**
50%	IRLS-1	14.74	25
	SRAMC-ADM	**16.22**	21
	SRAMC-APG	15.76	**12**
70%	IRLS-1	12.85	24
	SRAMC-ADM	**13.47**	21
	SRAMC-APG	13.16	**13**

Figure 3.4 shows a plot for the correlation between each column of the recovered and the original images. It also shows that the SRAMC-ADM has the largest correlation with the original image followed by the SRAMC-APG and the IRLS-1.

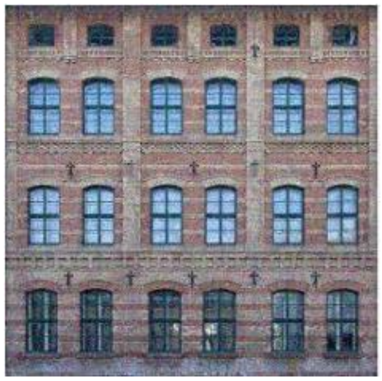
In Table 3.1, we compare the average SNR and the execution time for the three algorithms when applied to a set of 12 standard images[1] (256×256 and rank 60). We also added Gaussian noise with variance 0.01.

It can be observed that the SRAMC-ADM always gives the highest SNR. On the other hand SRAMC-APG achieves the lowest execution time.

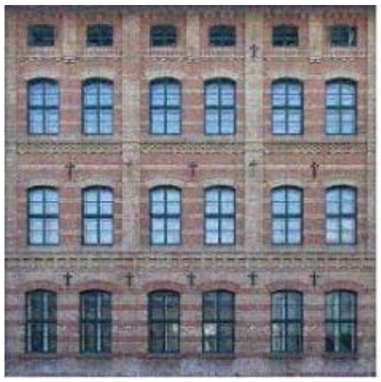
(a)



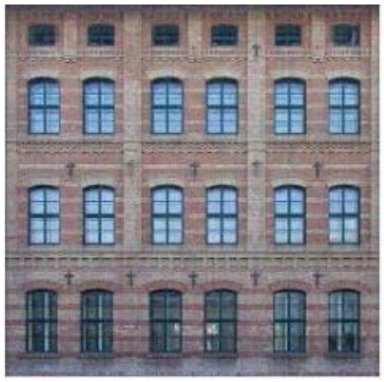
(b)



(c) 15.76



(d) 17.34



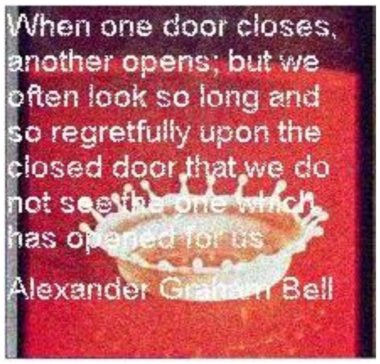
(e) 18.57

Figure 3.5: (a) The original façade image, (b) The noisy observation, (b-d) Inpainted image and the corresponding SNR (dB) using (b) ILRS-1, (c) SRAMC-APG, (d) SRAMC-ADM

After presenting our algorithms and using them in general image completion from random noisy measurements, we compare the performance of our methods to the IRLS-1 algorithm using image inpainting application.

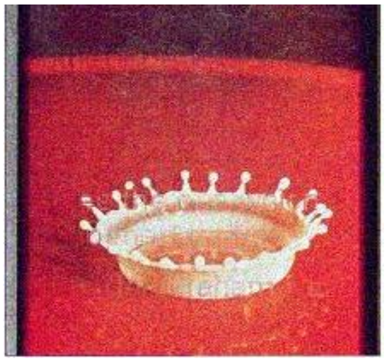
We apply the algorithms for color image inpainting from texture masking by completing each color channel separately as shown in Figure 3.5 for the façade image[2] (256×256) with Gaussian noise (variance 0.005). The figure also shows that our algorithms achieve better results than IRLS-1.

(a)

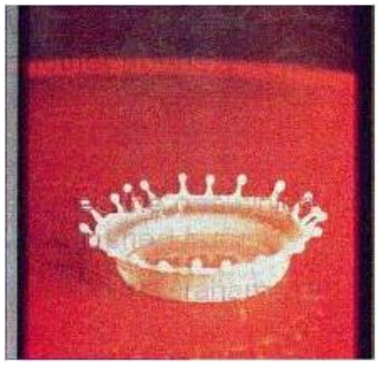


When one door closes,
another opens; but we
often look so long and
so regretfully upon the
closed door that we do
not see the one which
has opened for us
Alexander Graham Bell

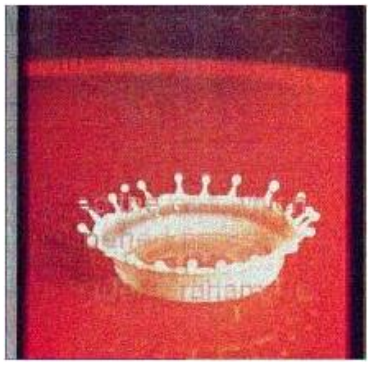
(b)



(c) 14.91



(d) 15.76



(e) 16.69

Figure 3.6: (a) The original drop image, (b) The masked noisy observation, (b-d) Inpainted image and the corresponding SNR (dB) using (b) ILRS-1, (c) SRAMC-APG, (d) SRAMC-ADM

Figure 3.6 shows the results of applying the algorithms for colored image inpainting from a text mask. Again, SRAMC-ADM achieves the highest SNR, followed by SRAMC-APG and the IRLS-1 comes last.

For this application, we used the drop image[3] (size 220×220, rank 200).

3.4.2. Tensor Completion

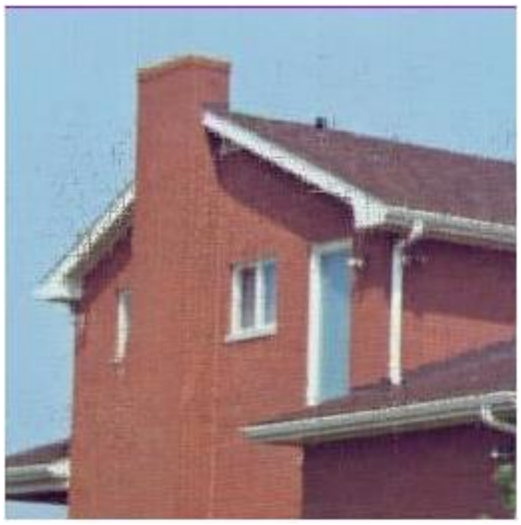
In this section we present the results of the experiments that we performed to compare the performance of our SRATC algorithm to the Low Rank Tensor Completion (LRTC) algorithm in [30] and the Accelerated Proximal Gradient (APG) tensor completion in [71].

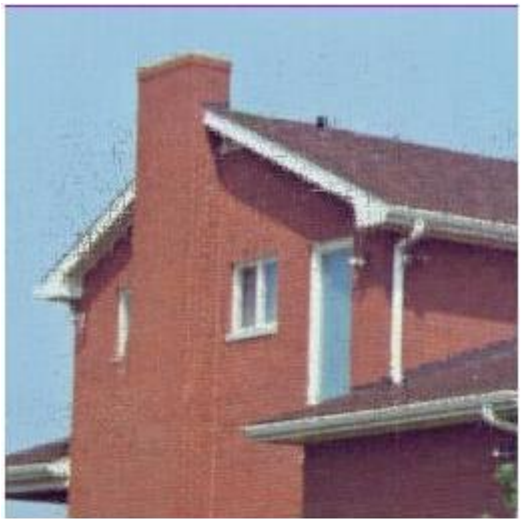
The quality of the recovered data is evaluated using the Peak Signal to Noise Ratio (PSNR) and the convergence rate is evaluated based on the time needed by the algorithm to converge to the final results. We ran the algorithms with MATLAB 2012b on the same desktop computer with a 3.3 GHz i-5 CPU and a 4GB Memory.

In the first application we applied the tensor completion algorithms to recover the house color image (size 256×256) [30] with rank 80 in each color channel from random 50% observed samples; the results are shown in Figure 3.7.

From Figure 3.7 we see that the APG takes less execution time than LRTC but it converges to the lowest PSNR, while the LRTC takes more time to execute than APG but it achieves higher PSNR. Our algorithm achieves comparative PSNR to LRTC, but it needs the lowest execution time. To test the performance of the tensor completion algorithms under high number of missing entries, we used the algorithms to recover the façade image (size 256×256 and rank 80 for each color channel) in [30] and using only 30% observed samples; the results are presented in Figure 3.8.

(a)


(b) PSNR:26.30, Time:76.5


(c) PSNR:25.79, Time:64.1

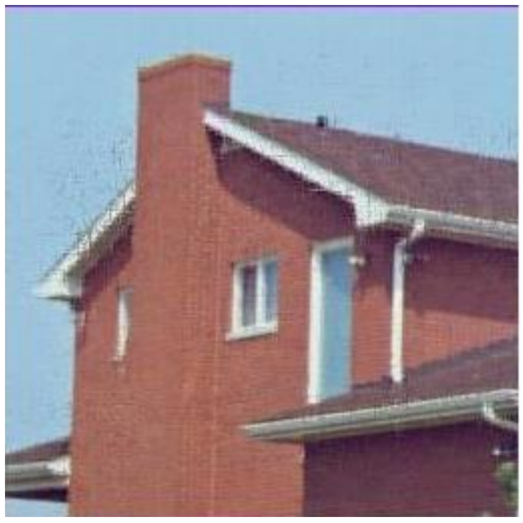
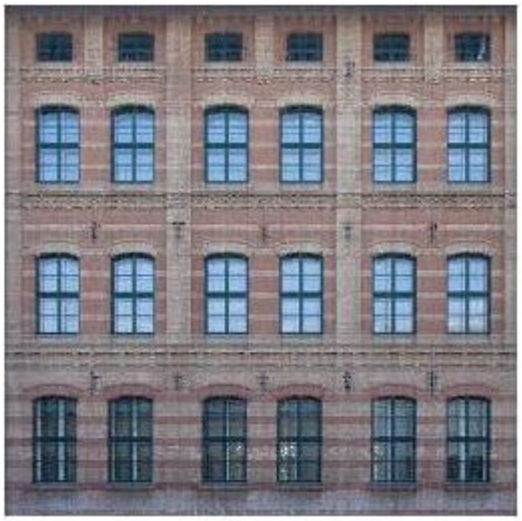

(d) PSNR:26.34, Time:37.8

Figure 3.7: (a) The original house image, (b-d) Recovered images with the corresponding PSNR (dB) and execution time (Sec.) using (b) LRTC, (c) APG, (d) SRATC

(a)

(b) PSNR:21.34,Time: 90.7


(c) PSNR:20.41,Time:75.5

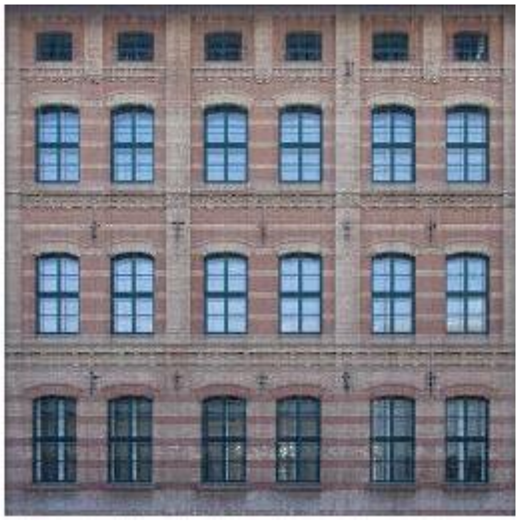

(d) PSNR: 21.27,Time:49.5

Figure 3.8: (a) The original façade image, (b-d) Recovered images with the corresponding PSNR (dB) and execution time (Sec.) using (b) LRTC, (c) APG, (d) SRATC

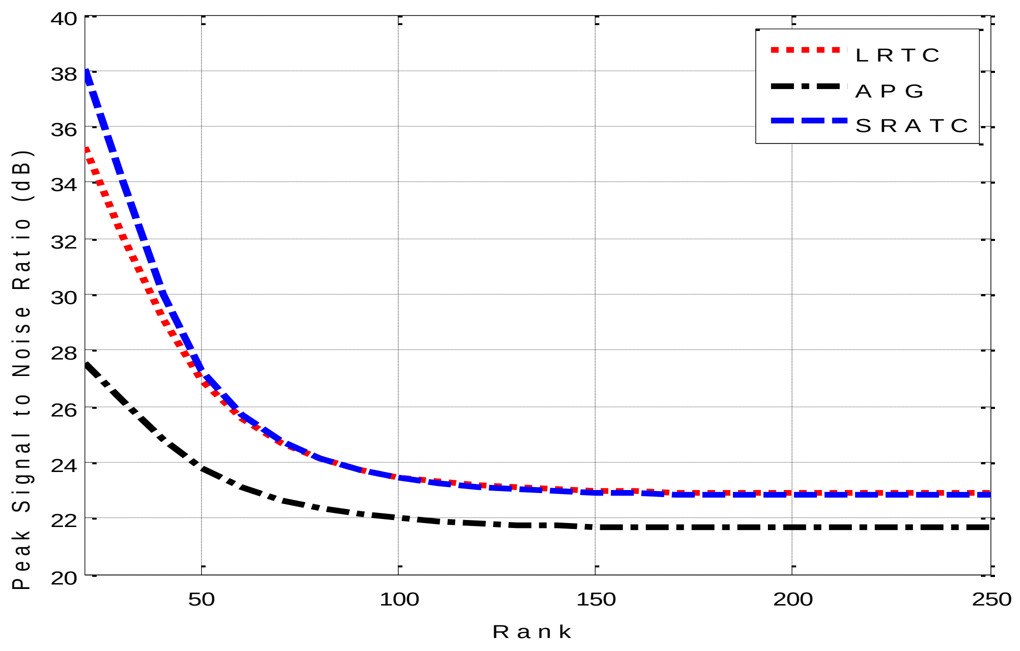


Figure 3.9: Peak Signal to Noise Ratio versus the rank of the matrix using lena colored image with 50% observed samples.

Again, Figure 3.8 shows that APG achieves the lowest PSNR measure but converges faster than LRTC, while the proposed SRATC algorithm converges the fastest with almost the same performance as LRTC. LRTC converges the slowest but it achieves a good PSNR value.

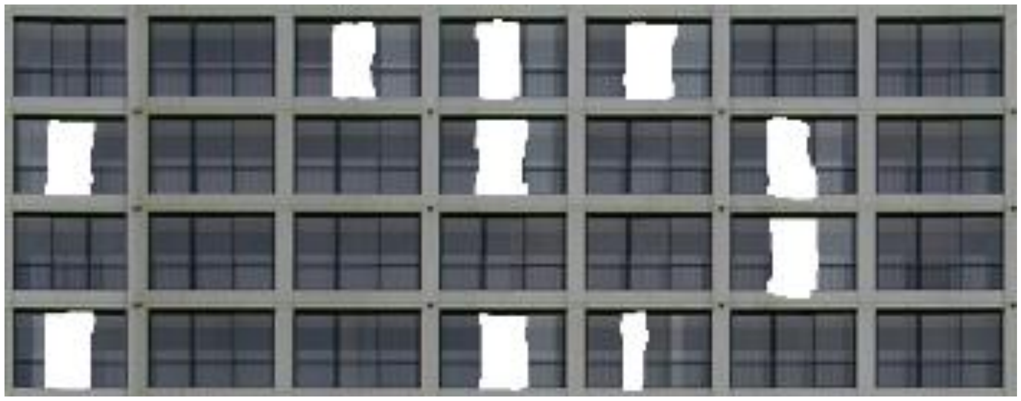
Figure 3.9 shows the effect of changing the rank on the performance of the tensor completion algorithms. In this experiment we used the color lena image (size 256×256) with 50% observed samples.

From Figure 3.9 we see that when the rank of the data is very low, SRATC achieves the highest PSNR. While when the rank starts to increase, the SRATC and LRTC algorithms both have approximately the same PSNR measure. From this experiment, we also noticed that the computation time remains approximately constant as we change the rank.

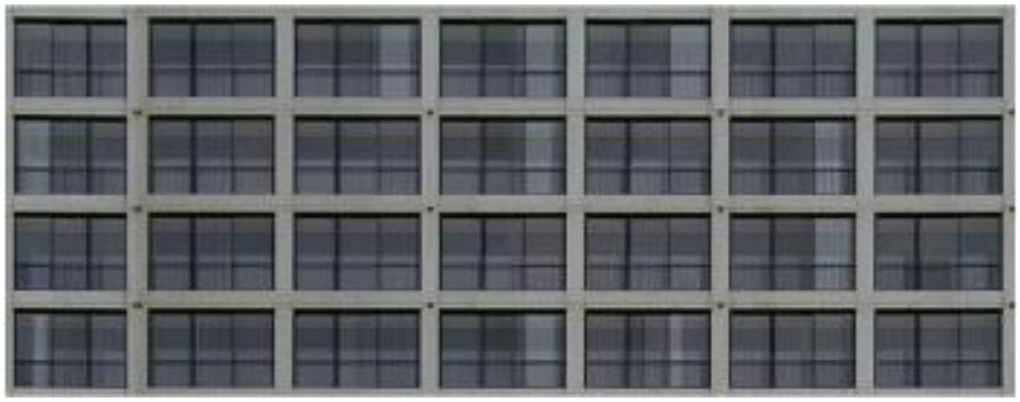
Table 3.2: Average execution time and PSNR comparison for tensor completion algorithms using a set of 10 colored images.

Missing (%)	**Method**	**PSNR (dB)**	**Time (Second)**
30	LRTC	32.84	63.5
	APG	30.51	57.6
	SRATC	**33.85**	**28.7**
50	LRTC	26.59	77.3
	APG	24.93	57.1
	SRATC	**26.74**	**31.5**
70	LRTC	**21.54**	77.3
	APG	19.39	70.0
	SRATC	21.46	**40.3**

To give an accurate performance evaluation for the tensor completion algorithms, we apply the algorithms to recover a set of 10 different color images[1] (size 256×256 and rank 60 for each color channel) from (70, 50 and 30)% observed entries; then we average the PSNR and the execution time; the results are shown in Table 3.2 We notice that regardless of the number of observed entries, the SRATC algorithm achieves the lowest execution time while LRTC and APG require significantly longer time to converge because of using the SVD decomposition in each iteration.

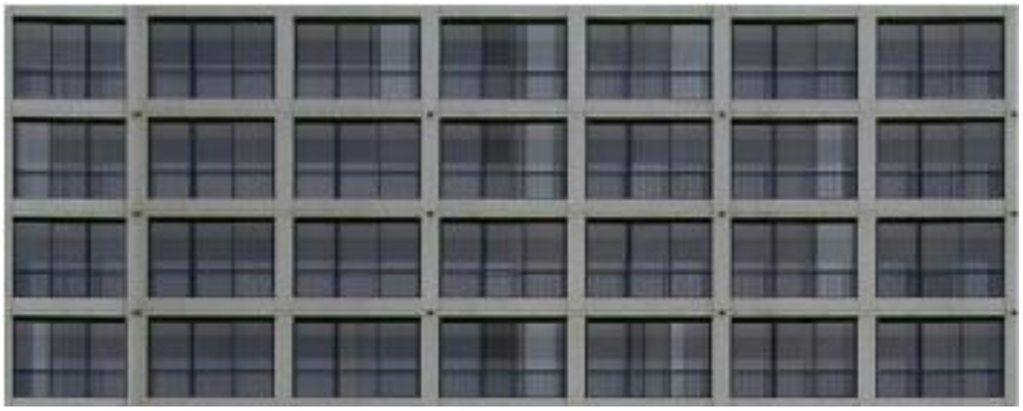


(a)



(b) Time:751.8

Figure 3.10a: (a) Original façade image (b) Recovered images with the corresponding execution time (Sec.) using LRTC.


(a) Time:631.4

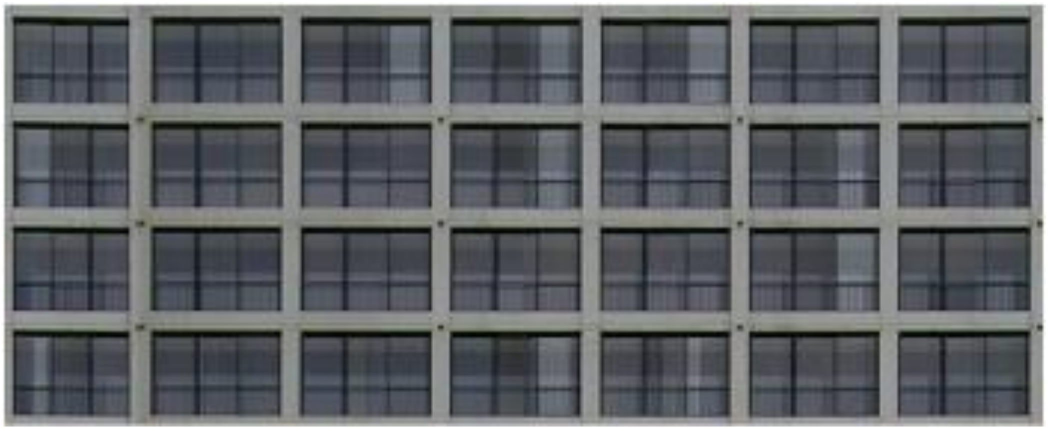

(b) Time:395.9

Figure 3.10b: Recovered images with the corresponding execution time (Sec.) using (a) APG, (b) SRATC.

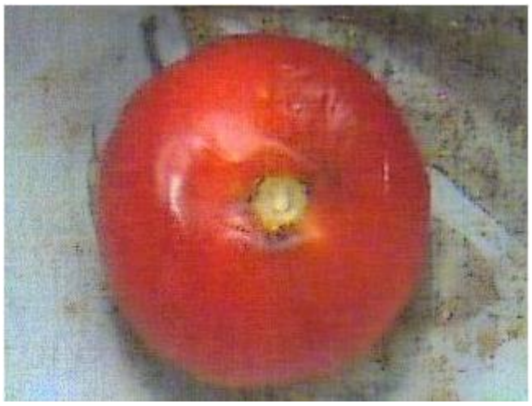
Another application for testing the tensor completion algorithms is blocks image inpainting. We used the façade image of size 318×861 [30]. In this application we are trying to recover the missing parts of the façade image texture by inpainting missing blocks of the image. The results are shown in Figures 3.10a and 3.10b; These results show that our algorithm also converges faster than state-of-the-art tensor completion algorithms. For this particular simulation, we don’t have the original image for evaluating the PSNR values;

however, by observing the visual quality of the different results, one can observe that the proposed SRATC algorithm provides a visual quality similar to the LRTC algorithm while converging with about one-half of the time required by LRTC.

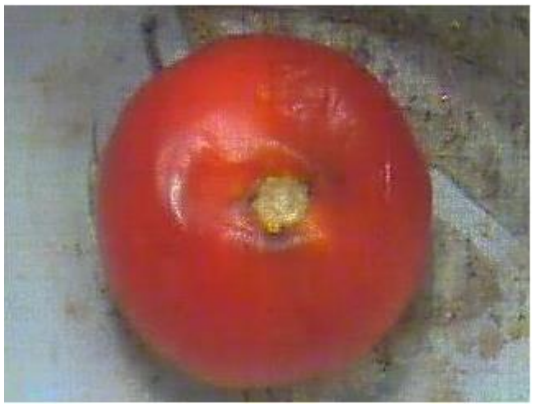
To show the applicability of our SRATC algorithm for higher dimensional data and non-square tensors, we show in Figure 3.11 the result of recovering a video sequence of the tomato video [30]; we used 20 frames with each frame of size 242×320 pixels. We also compared the SRATC result to LRTC and APG. As shown in Figure 3.11, our SRATC algorithm saves about 4 minutes when compared to the APG algorithm in addition to the higher PSNR and the visually better recovered frames. Compared to LRTC, our algorithm saves approximately 10 minutes execution time and achieves the same PSNR value.

(a)

(b) PSNR: 22.82, Time: 1892.6



(c) PSNR:21.01, Time:1530.8



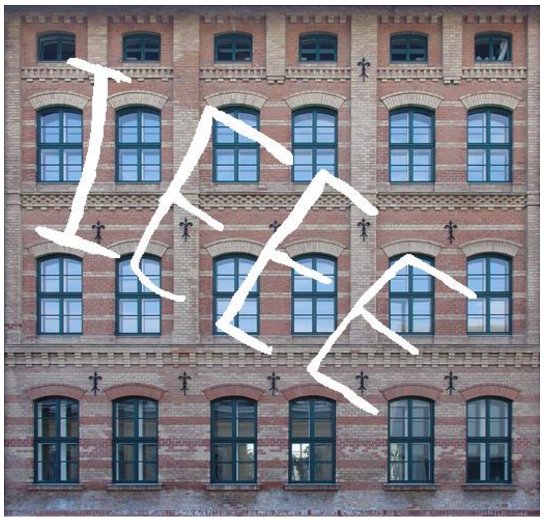
(d) PSNR: 22.86, Time: 1311.4

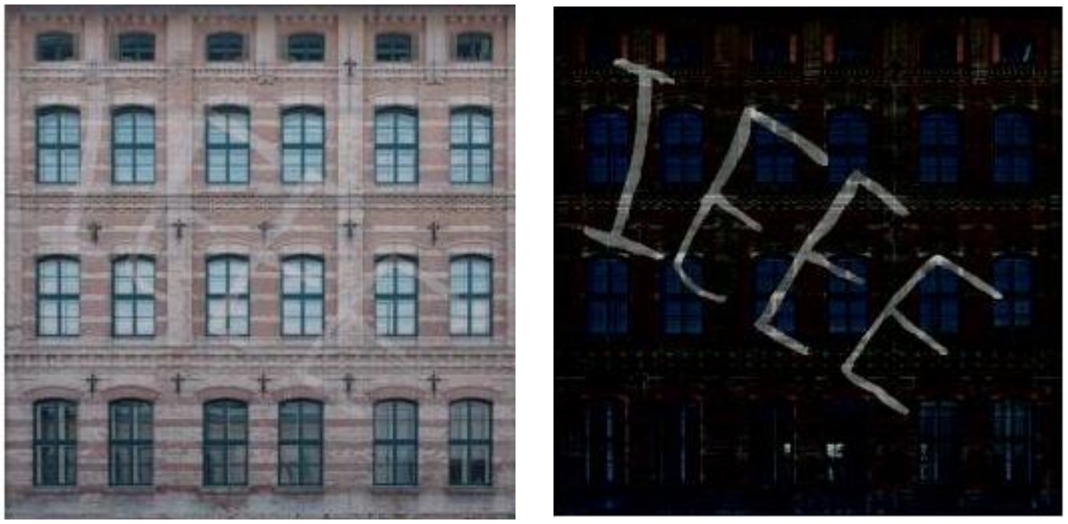
Figure 3.11: (a) The original tomato frame (number 6), (b-d) Recovered frames with the corresponding PSNR (dB) and execution time (Sec.) using (b) LRTC, (c) APG, (d) SRATC

3.4.3. Robust Tensor Recovery

In this section we compare the performance and execution time of our proposed algorithms against the Rank Sparsity Tensor Decomposition (RSTD) algorithm which is proposed in [6], the Multi-linear Augmented Lagrangian Multiplier (MALM) in [7] and finally the Inexact Alternating Direction Augmented Lagrangian (IADAL) in [2]. For a fair comparison, we ran all algorithms using MATLAB R2016a on a computer with 3.3 GHz i-5 CPU and 4GB Memory. We also set the same maximum number of iterations to 700 for all algorithms since most of the algorithms converge before reaching this number of iterations, while the algorithms that do not converge they do not achieve significant PSNR improvements if kept running longer. Further, for all algorithms, we used the relative absolute error $\left\|\mathrm{X}^{k+1} - \mathrm{X}^{k}\right\|_F / \left\|\mathrm{X}^{k}\right\|_F < \epsilon$ as a stopping condition.

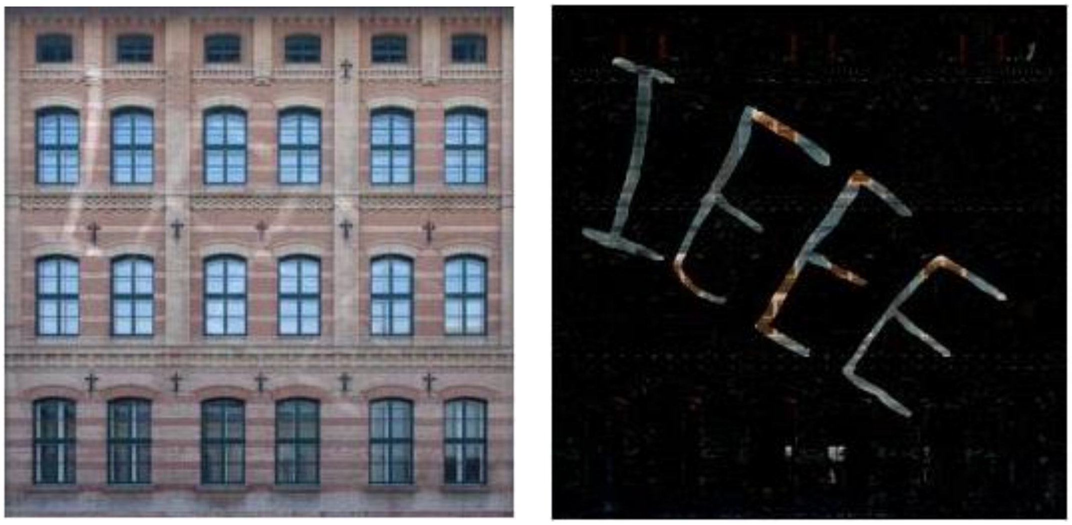
Figure 3.12 shows the results of applying the robust tensor recovery algorithms for image inpainting using the facade image of size 256×256×3. We set the rank for each unfold to 80×80×3. The figure shows that our framework, in general, achieves a significant improvement in PSNR and execution time followed by IADAL, RSTD and finally MALM.



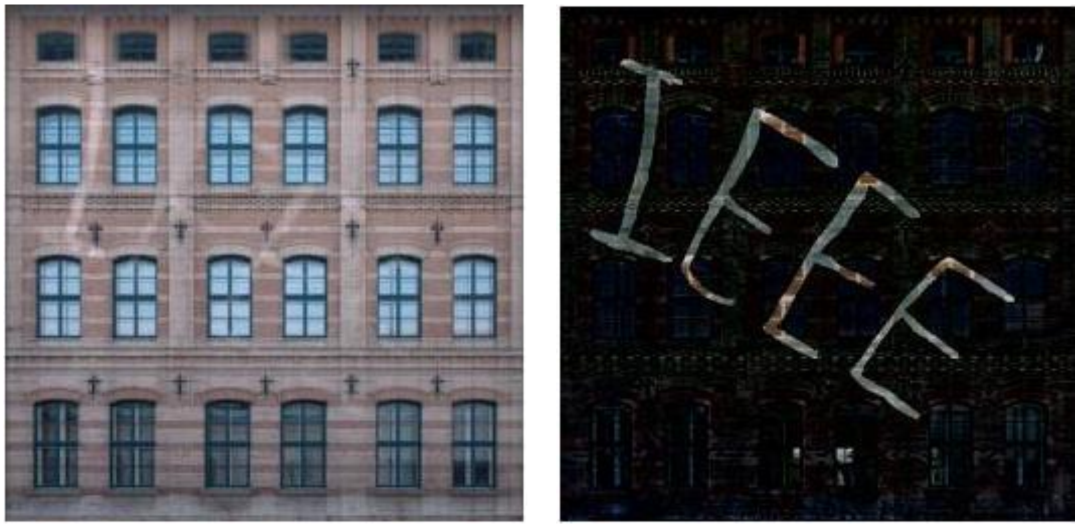


MALM: PSNR:21.43 dB, Time:96 Sec

Figure 3.12a: Top: The observed image. Left: low rank model. Right: Sparse model using MALM.

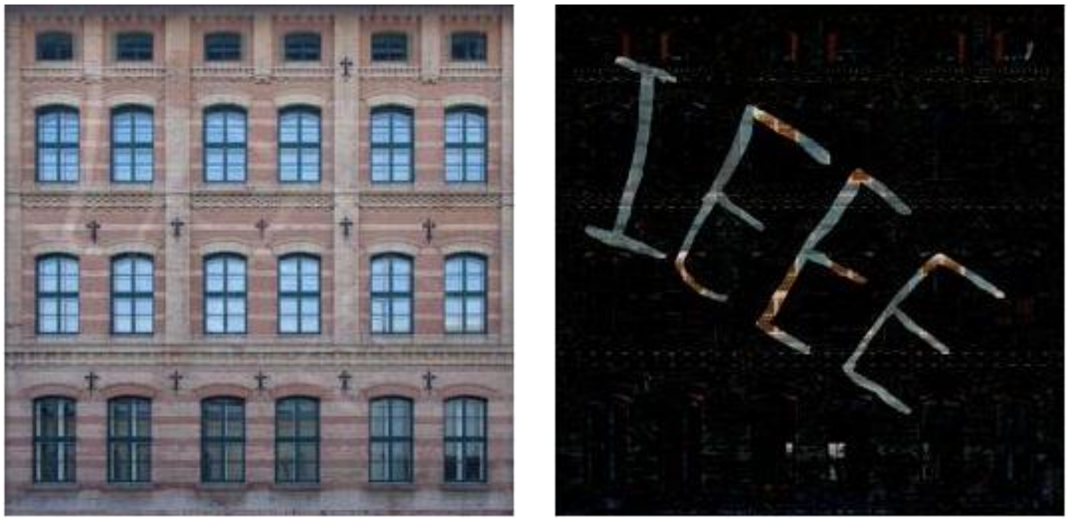


RSTD: PSNR:22.79 dB, Time:113 Sec

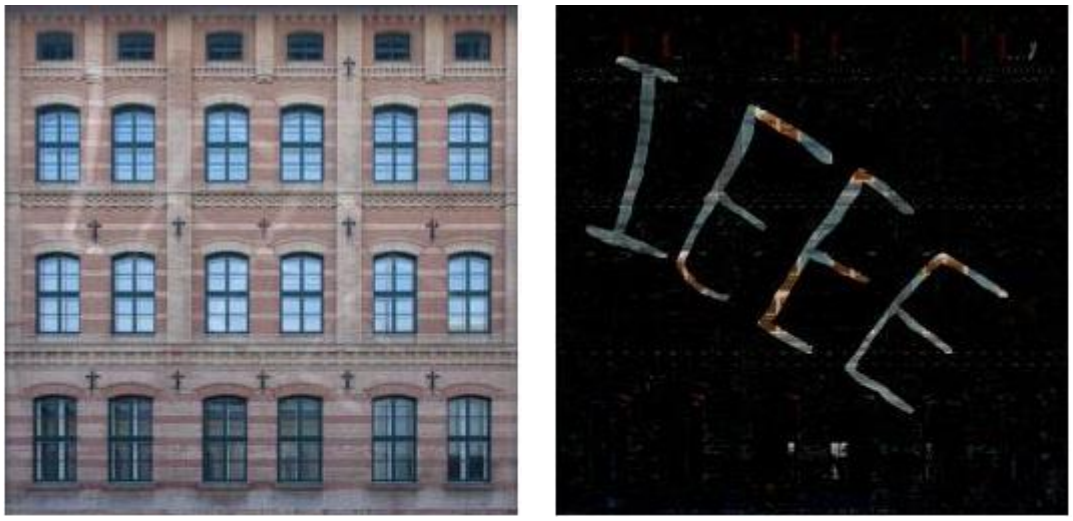


IADAL: PSNR:24.16 dB, Time:76 Sec

Figure 3.12b: Left: low rank model. Right: Sparse model using RSTD (Top) and IADAL (Bottom).



TSRTR-ALM: PSNR: 26.71 dB, Time: 50 Sec



TSRTR-APG: PSNR:25.68 dB, Time: 39 Sec

Figure 3.12c: Left: low rank model. Right: Sparse model using TSRTR-ALM (Top) and TSRTR-APG (Bottom).

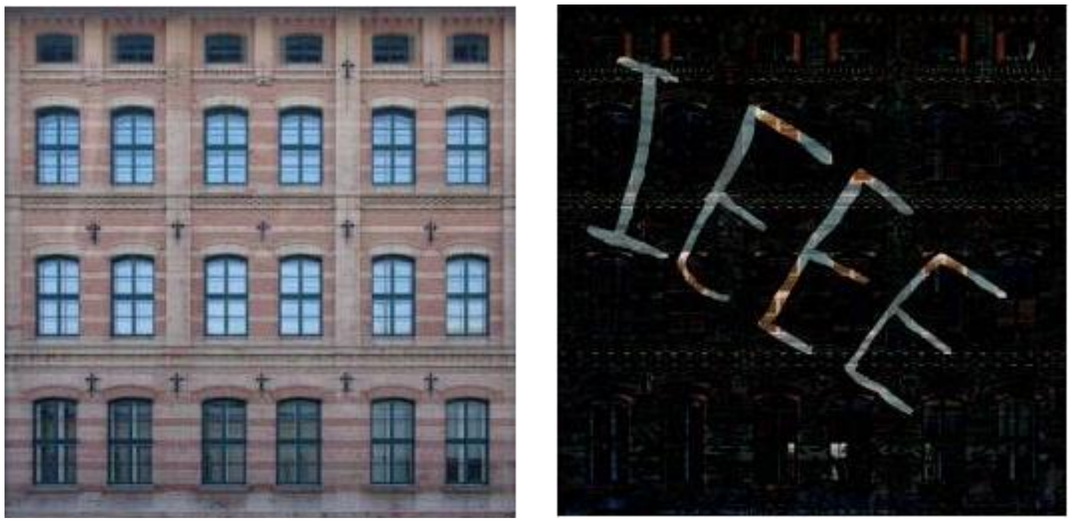

TSRTR-ADMM: PSNR:26.97 dB, Time:69 Sec

Figure 3.12d: Left: low rank model. Right: Sparse model using TSRTR-ADMM.

Among our proposed algorithms, we notice that using TSRTR-APG does accelerate the convergence of the recovery with a small sacrifice in terms of quality. On the other hand, TSRTR-ADMM consumes longer time to converge but achieves the best reconstruction quality due to the advantage of the separable structure over TSRTR-ALM, which came in between both algorithms.

Even though all these algorithms compute the SVD one time in each iteration, we notice that the algorithms derived based on the proposed framework still have the fastest computation time due to the fast PSNR correction throughout iterations.

As shown in Figure 3.13, which plots the progress of PSNR improvements over iterations for all tensor RPCA algorithms, TSRTR-APG converges within 200 iterations, while TSRTR-ADMM consumes the whole 700 iterations without convergence. However, TSRTR-ADMM has the highest PSNR and the best visual quality than the rest of the algorithms. TSRTR-ALM requires less than 300 iterations to converge and its performance is slightly higher than TSRTR-APG.

In comparison to the rest of the algorithms, both MLAM and RSTD consume the maximum 700 iterations; IADAL converges within less than 500 iterations, but the PSNR values of these algorithms are more than 2dB less than the PSNR values of the proposed framework.

Figure 3.14 shows the effect of increasing the rank for the first two unfolds (for example rank 100 means the rank of the unfolded tensor is 100×100×3) of the same façade image on the performance of the tensor recovery algorithms.

We notice that TSRTR-ADMM algorithm achieves the highest PSNR over the whole rank range followed by IADAL by an average of 2 dB. We also see that as the rank increases, the difference in performance between TSRTR-ALM and IADAL increases. In addition to that, when the rank is high, MALM achieves higher PSNR than RSTD. For lower rank signals, RSTD achieves better PSNR than MALM. MALM is the least affected algorithm by increasing the rank of the image.

In Table 3.3, we compare the average PSNR and the execution time for the four algorithms using different masks applied to a set of 10 colored images shown in Figure 3.15 after setting the rank of each one of the unfolded tensor over each dimension to 100×100×3 before adding the mask.

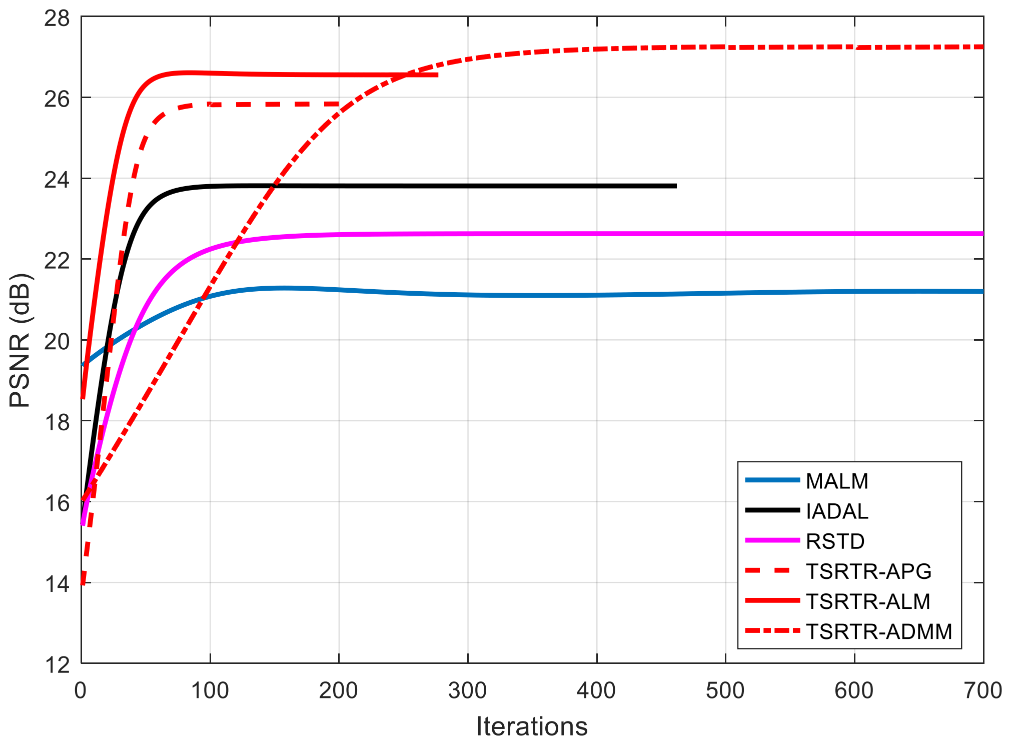


Figure 3.13: PSNR versus iteration number for the façade image in Figure 3.12.

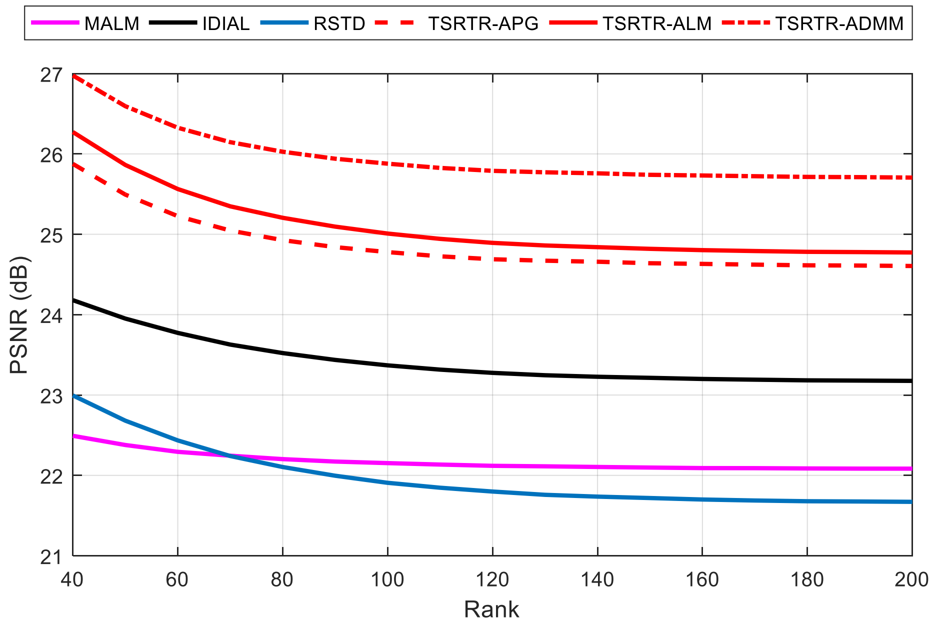


Figure 3.14: PSNR versus rank of the first two folds for the façade image in Figure 3.12.

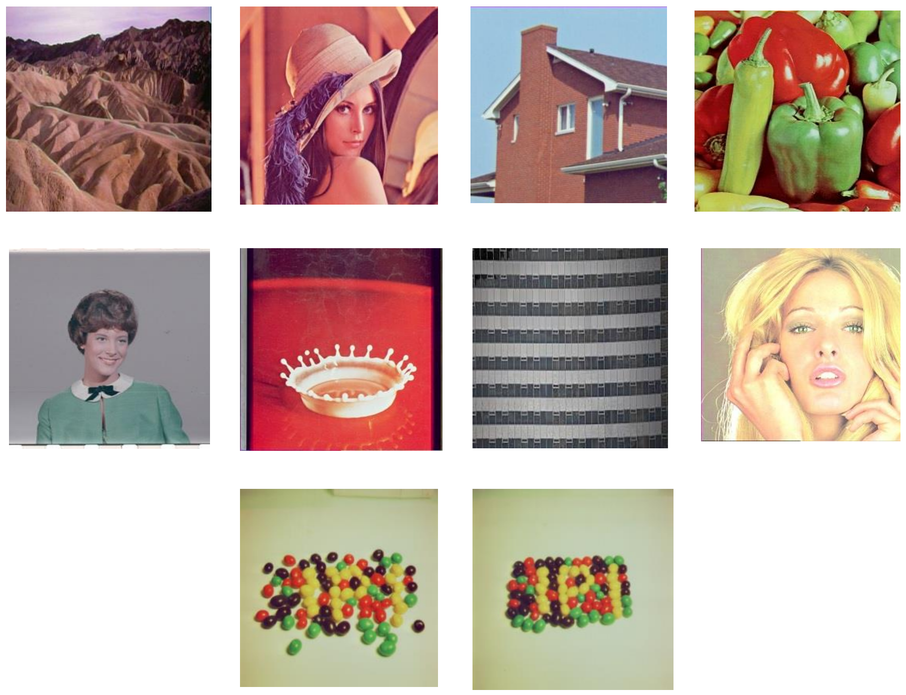
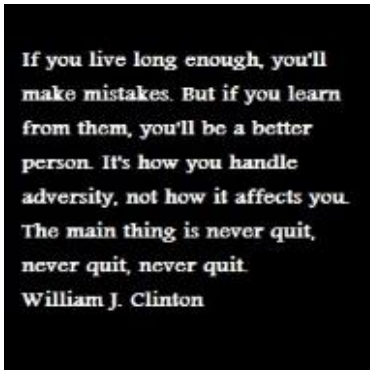
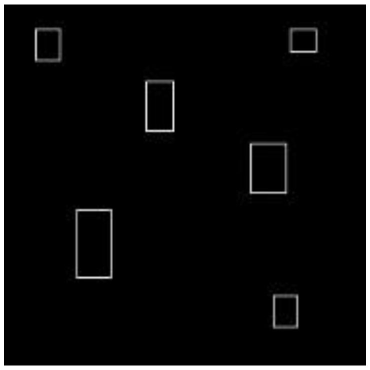


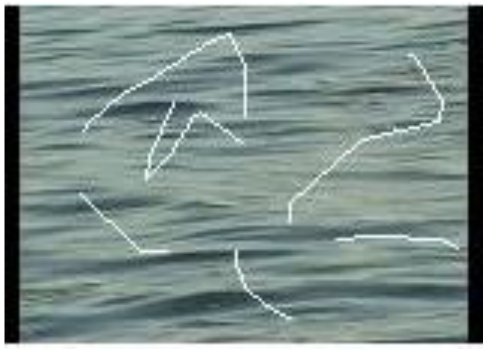
Figure 3.15: Set of images used for performance comparison in Table 3.3.

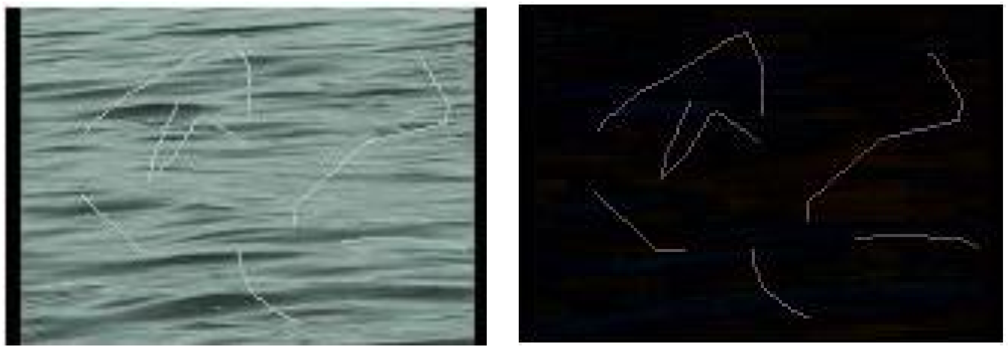
Table 3.3: Average execution time and PSNR comparison for tensor RPCA algorithms using a set of 10 colored images.

Mask	**Algorithm**	**PSNR (dB)**	**Time (Sec)**
 If you live long enough, you'll make mistakes. But if you learn from them, you'll be a better person. It's how you handle adversity, not how it affects you. The main thing is never quit, never quit, never quit. William J. Clinton	MALM	24.91	94
	RSTD	25.38	110
	IADAL	25.71	68
	TSRTR-ALM	**27.46**	**50**
	TSRTR-APG	**26.92**	**43**
	TSRTR-ADMM	**27.83**	**61**
	MALM	30.98	87
	RSTD	32.53	106
	IADAL	32.98	73
	TSRTR-ALM	**35.02**	**56**
	TSRTR-APG	**34.18**	**41**
	TSRTR-ADMM	**35.64**	**64**
	MALM	30.16	83
	RSTD	30.92	97
	IADAL	31.74	60
	TSRTR-ALM	**33.57**	**41**
	TSRTR-APG	**33.22**	**38**
	TSRTR-ADMM	**34.06**	**59**

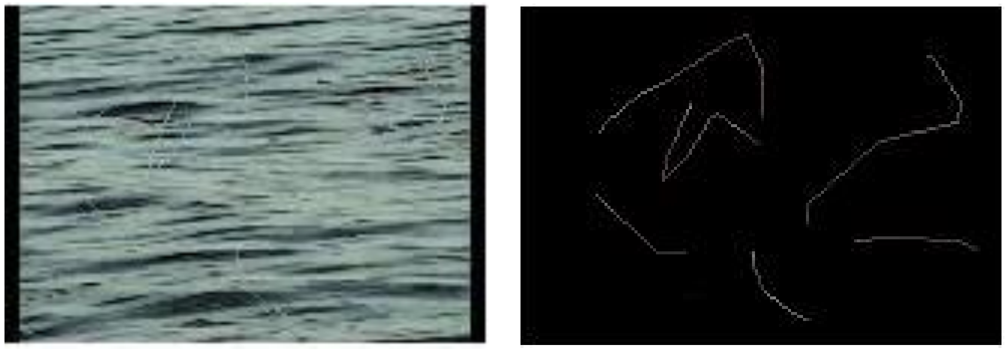
It shows clearly that on average our proposed framework achieves higher dB improvements over state-of-the-art algorithms. For example, TSRTR-ALM has average performance compared to the others, yet it achieves about 2 dB improvement in PSNR, and TSRTR-ADMM achieves about 1dB over that. One important reason for this improvement is using the truncated Schatten-*p* functions, which will only modify the smaller singular values and leave the higher singular values intact, which capture the main information of the image, intact. In addition to that and despite the fact that our framework has similar degree of complexity to the other tensor recovery algorithms, it needs less iteration to converge and it reduces the execution time by an average of 18%.

Finally, we apply the RPCA algorithms to recover the ocean video sequence [6] of size 112×160×3×32 and rank 50×50×3×10 that is corrupted by different random masks for different frames; the results are shown in Figure 3.16 for frame number 16 as a representative example of visual quality. From Figure 3.16 we also see that our proposed framework achieves about 2 dB enhancement in PSNR compared to IADAL, which achieved the second-best performance, after the algorithms based on the proposed framework, followed by RSTD and MALM. Even when the proposed algorithms are tested on four-dimensional data, TSRTR-ADMM still achieves the highest PSNR among the proposed algorithms with relatively high execution time, followed by TSRTR-ALM and then TSRTR-APG, which provided the lowest PSNR values when compared to the proposed algorithms. In terms of execution time, TSRTR-APG is still the fastest converging algorithm among the others followed by TSRTR-ALM by 31 seconds and then TSRTR-ADMM, which requires 58 seconds more than TSRTR-APG.



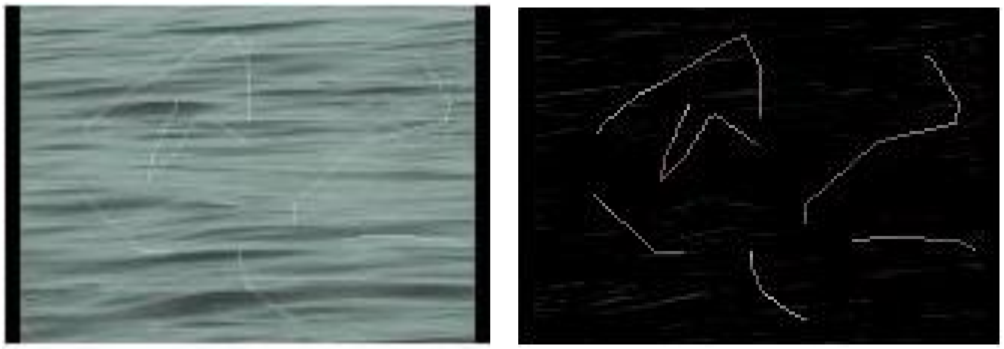


MALM-PSNR:23.75 dB, Time:214 Sec

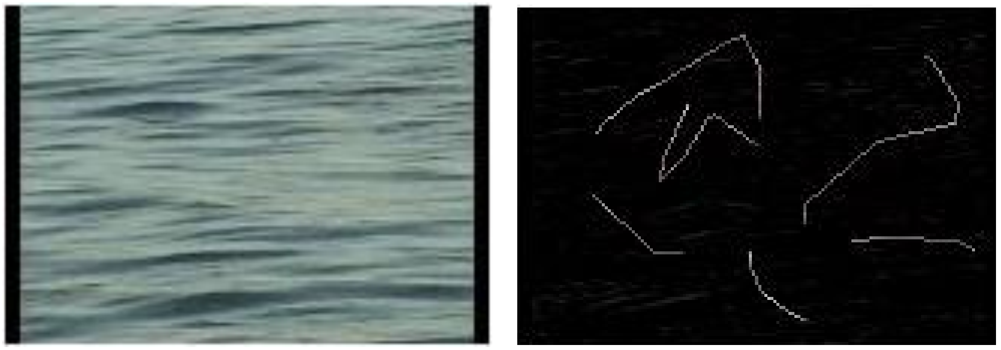


RSTD-PSNR: 25.21 dB, Time:263 Sec

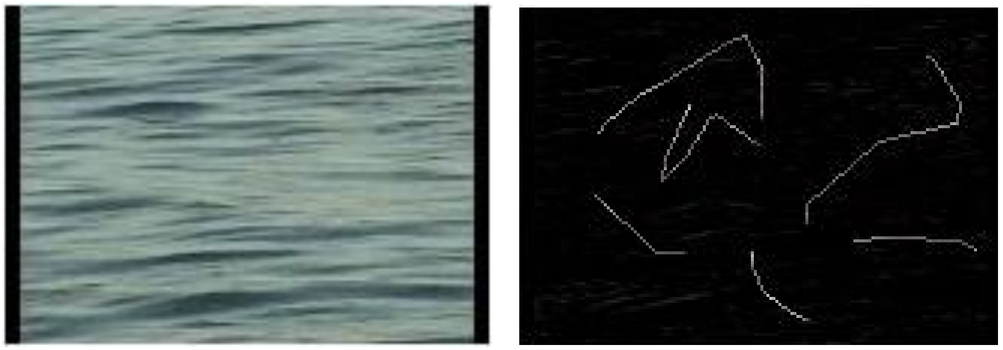
Figure 3.16a: Top: The observed image. Left: low rank model. Right: Sparse model using MALM (Top) and RSTD (Bottom).



IADAL-PSNR:27.12 dB, Time:185 Sec

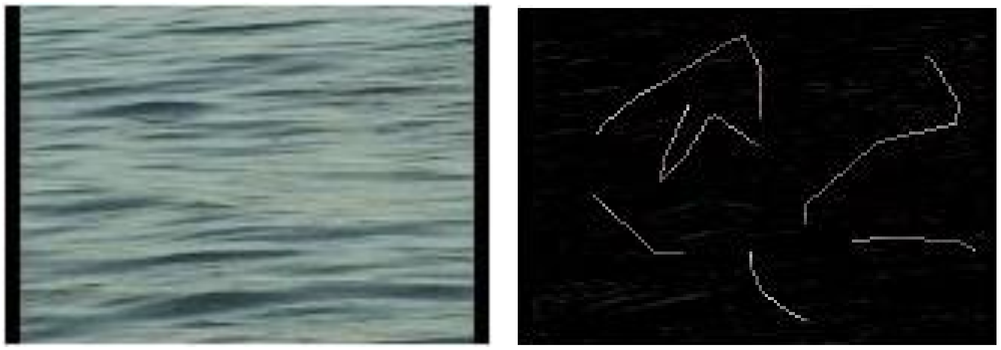


TSRTR-ALM: PSNR:29.08 dB, Time:116 Sec



TSRTR-APG: PSNR:28.41 dB, Time:85 Sec

Figure 3.16b: Left: low rank model. Right: Sparse model using IADAL (Top), TSRTR-ALM (Middle) and TSRTR-APG (Bottom).



TSRTR-ADMM: PSNR:29.67dB, Time:143 Sec

Figure 3.16c: Left: The observed image. Middle: low rank model. Right: Sparse model using TSRTR-ADMM.

3.5 Proof of Lemmas

3.5.1. Proof of Lemma 3.1:

From (3.26) we complete the sum of squares:

$$L(\mathcal{X}, \mathcal{E}, \mathcal{Y}, \mu) = \sum_{i=1}^{N} \alpha_i Tr((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{\frac{p}{2}} + \frac{\mu}{2} \left\| P_\Omega(\mathcal{M}) - \mathcal{X} - \mathcal{E} + \frac{1}{\mu} \mathcal{Y} \right\|_F^2 - \frac{1}{\mu} \|\mathcal{Y}\|_F^2 \tag{A1}$$

Deriving and solving (A1) for $\mathcal{X}$:

$$\sum_{i=1}^{N} \frac{\alpha_i ((\mathcal{X})_{(i)}^T)}{((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{1-\frac{p}{2}}} - \mu (P_\Omega(\mathcal{M}) - \mathcal{X} - \mathcal{E} + \frac{1}{\mu} \mathcal{Y}) = 0 \tag{A2}$$

For simplicity, we assume W_i is given by (3.32). Then the solution for $\mathcal{X}$ is:

$$\mathcal{X}^{k+1} = P_\Omega(\mathcal{M}) - \mathcal{E} + \frac{1}{\mu}\mathcal{Y} - \frac{\frac{1}{\mu}\sum_{i=1}^{N} \beta_i \, fold\,(\alpha_i \, (\mathcal{X})_{(i)}^{T} W_i)_{(i)}}{\sum_{i=1}^{N} \beta_i} \tag{A3}$$

For further simplification, we assume ${\mathcal{Z}_i}^{k+1}$ as presented in (3.28) and hence $\mathcal{X}^{k+1}$ is reduced to (3.27).

3.5.2. Proof of Lemma 3.2:

We start from (A1) and derive for $\mathcal{E}$. But here we need to include another constraint to force the entries of $\mathcal{E}$ at the observed samples entries to be zero.

$$\begin{gathered} \mu(P_\Omega(\mathcal{M}) - \mathcal{X} - \mathcal{E} + \frac{1}{\mu}\mathcal{Y}) = 0 \\ s.t. P_\Omega(\mathcal{E})=0 \end{gathered} \tag{A4}$$

Now solving for $\mathcal{E}$:

$$\begin{gathered} \mathcal{E}^{k+1} = P_\Omega(\mathcal{M}) - \mathcal{X} + \frac{1}{\mu}\mathcal{Y} \\ P_\Omega(\mathcal{E}^{k+1})=0 \end{gathered} \tag{A5}$$

Or we can rewrite (A4) as in (3.30).

3.5.3. Proof of Lemma 3.3:

Also starting from (A1), we derive for $\mathcal{Y}$:

$$P_\Omega(\mathcal{M}) - \mathcal{X} - \mathcal{E} + \frac{1}{\mu}\mathcal{Y} - \frac{1}{\mu}\mathcal{Y} = 0 \tag{A6}$$

Solving for $\mathcal{Y}$:

$$\mathcal{Y}^{k+1} = \mathcal{Y}^{k} + \mu(P_{\Omega}(\mathcal{M}) - \mathcal{X} - \mathcal{E}) \tag{A7}$$

Which is the same result shown in (3.31).

3.5.4. Proof of Lemma 3.4:

First we complete the sum of squares:

$$L(\mathcal{X}, \mathcal{E}, \mathcal{Y}, \mu) = \sum_{i=1}^{m} \left[\alpha_i Tr\left[((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i(\mathcal{X})_{(i)} B_i{}^{T} \right) \right] \right.$$
$$\left. + \ \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 \right] + \frac{\mu}{2} \left\| \mathcal{M} - \mathcal{X} - \mathcal{E} + \frac{1}{\mu}\mathcal{Y} \right\|_F^2 - \frac{1}{\mu} \|\mathcal{Y}\|_F^2 \tag{A8}$$

Differentiating (A1) for $\mathcal{X}$:

$$\sum_{i=1}^{m} \frac{\alpha_i \, fold((\mathcal{X})_{(i)}^{T})}{((\mathcal{X})_{(i)}^{T}(\mathcal{X})_{(i)} + \gamma I)^{1-\frac{p}{2}}} - \beta_i(A_i^T B_i) - \mu(\mathcal{M} - \mathcal{X} - \mathcal{E} + \frac{1}{\mu}\mathcal{Y}) = 0 \tag{A9}$$

Assuming W_i is given by (3.32). Then the solution for $\mathcal{X}$ is:

$$\mathcal{X}^{k+1} = \mathcal{M} - \mathcal{E} + \frac{1}{\mu}\mathcal{Y} -$$
$$\frac{\frac{1}{\mu}\sum_{i=1}^{m} \alpha_i \, fold\,(\,(\mathcal{X})_{(i)}^{T} W_i)_{(i)} - \beta_i(A_i^T B_i))}{\sum_{i=1}^{N} \alpha_i} \tag{A10}$$

By letting $\mathcal{Z}_i{}^{k+1}$ as presented in (10), $\mathcal{X}^{k+1}$ is reduced to (8).

3.5.5. Proof of Lemma 3.5:

We start from (A1) and differentiate the equation for $\mathcal{E}$.

$$\frac{\partial}{\partial \mathcal{E}} L(\mathcal{X}, \mathcal{E}, \mathcal{Y}, \mu) = \frac{\partial}{\partial \mathcal{E}} [\sum_{i=1}^{m} \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 + \frac{\mu}{2} \left\| \mathcal{M} - \mathcal{X} - \mathcal{E} + \frac{1}{\mu} \mathcal{Y} \right\|_F^2] \tag{A11}$$

The solution to (A4) for E is by soft thresholding [4] :

$$\mathcal{E}^{k+1} = \mathfrak{S}_{\frac{\lambda}{\mu}} (\mathcal{M} - \mathcal{X}^{k+1} + \frac{1}{\mu} \mathcal{Y}^{k}) \tag{A12}$$

3.5.6. Proof of Lemma 3.6:

To find the low rank (S_X) and sparse (S_E) tensors, we can minimize (3.47) using Block Coordinate Descent (BCD), we fix Y, Z ; then we differentiate and solve for S_X and S_E separately.

$$\min_{\mathcal{S}_{\mathcal{E}}} Q_\tau(\mathcal{S}_{\mathcal{E}}) = \frac{\tau}{2} \| \mathcal{S}_{\mathcal{E}} - \ \mathcal{Y}_{\mathcal{E}} \|_F^2 + \ \lambda \| \mathcal{S}_{\mathcal{E}} \|_1 \tag{A13}$$

By removing all constant terms, equation (A6) becomes a standard ℓ_1 minimization equation [19] represented in the tensor domain, and by applying the thresholding solution we can easily get the optimal solution for the sparse tensor $\mathcal{S}_{\mathcal{E}}$ in (3.52).

Now, we solve (3.47) for $\mathcal{S}_X$ and we start by removing the constant terms.

$$\begin{aligned} \min_{\mathcal{S}_X} Q_\tau(\mathcal{S}_X) = \frac{\tau}{2} \left\| \mathcal{S}_X - \ \mathcal{Y}_X \right\|_F^2 \\ + \sum_{i=1}^{m} \left[\alpha_i Tr \left[((\mathcal{S}_X)_{(i)}^T (\mathcal{S}_X)_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{S}_X)_{(i)} {B_i}^T \right) \right] \right] \end{aligned} \tag{A14}$$

Diffrenciating (A8) and adding a weighting parameter $0 \leq \alpha_i \leq 1$ to control the weight of each fold of the tensor.

$$\tau(\mathcal{S}_{\mathcal{X}} - \mathcal{Y}_{\mathcal{X}}) + \frac{\sum_{i=1}^{m} \left[\alpha_i\, fold\left((\mathcal{S}_{\mathcal{X}})_{(i)}^T W_i\right)_{(i)} - \beta_i (A_i^T B_i))\right]}{\sum_{i=1}^{m} \alpha_i} = 0 \tag{A15}$$

The final step is to find the low rank tensor S_X which is equivalent to (3.52).

$$\mathcal{S}_{\mathcal{X}} = \mathcal{Y}_{\mathcal{X}} - \frac{1}{\tau}\frac{\sum_{i=1}^{m} \left[\alpha_i\, fold\left((\mathcal{X})_{(i)}^T W_i\right)_{(i)} - \beta_i (A_i^T B_i))\right]}{\sum_{i=1}^{m} \alpha_i} \tag{A16}$$

3.5.7. Proof of Lemma 3.7:

Before we start solving (3.47), we need to simplify the mathematical representation by completing the sum of squares.

$$\min_{\mathcal{X},\, \mathcal{E},\, \mathcal{L}} Q(\mathcal{X}, \mathcal{E}, \mathcal{L})$$

$$= \sum_{i=1}^{m} \left[\alpha_i Tr \left[((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}} - \beta_i \left(A_i (\mathcal{X})_{(i)} {B_i}^T\right)\right] + \lambda \left\|(\mathcal{E})_{(i)}\right\|_1\right] + \frac{\tau}{2}\left\|\mathcal{X} + \mathcal{E} - \mathcal{M} + \frac{1}{\tau}\mathcal{L}\right\|_F^2 - \frac{1}{\tau}\|\mathcal{L}\|_F^2 \tag{A17}$$

To find $\mathcal{X}$, we solve (3.52) using BCD. First, we fix $\mathcal{E}, \mathcal{L}$; and differentiate for $\mathcal{X}$. We also add a weighting parameter $0 \leq \alpha_i \leq 1$ to control the weight of each fold of the tensor.

$$\frac{\sum_{i=1}^{m} \dfrac{\alpha_i\, fold((\mathcal{X})_{(i)}^T)}{((\mathcal{X})_{(i)}^T (\mathcal{X})_{(i)} + \gamma I)^{\frac{1}{2}}} - \beta_i (A_i^T B_i)}{\sum_{i=1}^{m} \alpha_i} + \tau\left(\mathcal{X} + \mathcal{E} - \mathcal{M} + \frac{1}{\tau}\mathcal{L}\right) = 0 \tag{A18}$$

Using the definition of W_i in (3.32) and solving for X.

$$\mathcal{X}^{k+1} = \mathcal{M} - \left[\sum_{i=1}^{m} \alpha_i \, fold \begin{pmatrix} (\mathcal{E}^{\,k+1})_{(i)} - \frac{1}{\tau}\left(\mathcal{X}_i^{\,k}\right)_{(i)} W_i + \\ \frac{\beta_i}{\tau} A_i^T B_i \end{pmatrix}_{(i)} \right] / \sum_{i=1}^{m} \alpha_i + \frac{1}{\tau}\mathcal{L} \qquad \text{(A19)}$$

To solve for $\mathcal{E}$, we fix the other tensor variables and remove all constant terms, equation (A10) is reduced to:

$$\min_{\mathcal{E}} Q(\mathcal{E}) = \sum_{i=1}^{m} \lambda \left\| (\mathcal{E})_{(i)} \right\|_1 + \frac{\tau}{2} \left\| \mathcal{X} + \mathcal{E} - \mathcal{M} + \frac{1}{\tau}\mathcal{L} \right\|_F^2 \qquad \text{(A20)}$$

Equation (A10) can also be solved using the thresholding solution used to solve the standard ℓ_1 optimization equation [19] represented in the tensor domain.

Now, we differentiate (A14) and solve it for $\mathcal{E}$, which results in:

$$\mathcal{E}^{\,k+1} = \mathfrak{S}_{\frac{\lambda}{\tau}}(\mathcal{M} - \mathcal{X}^k + \frac{1}{\tau}\mathcal{L}) \qquad \text{(A21)}$$

The update step for $\mathcal{L}$ can be easily found by fixing $\mathcal{X}$ and $\mathcal{E}$ and neglecting the constant terms.

$$\min_{\mathcal{L}} Q(\mathcal{L}) = \frac{\tau}{2} \left\| \mathcal{X} + \mathcal{E} - \mathcal{M} + \frac{1}{\tau}\mathcal{L} \right\|_F^2 - \frac{1}{2\tau} \| \mathcal{L} \|_F^2 \qquad \text{(A22)}$$

Differentiating and solving for $\mathcal{L}$:

$$\mathcal{L}^{k+1} = \mathcal{L}^k - \tau(\mathcal{X}^{k+1} + \mathcal{E}^{\,k+1} - \mathcal{M}) \qquad \text{(A23)}$$

Chapter 4

Single Image Super-Resolution

4.1 Example Based Single Image Super-resolution

Most previous example based single image super-resolution algorithms use two training dictionaries one for LR and the other for HR patches. These methods assume that there is a one-to-one mapping between LR and HR image patches. The training is usually done over a large set of pairs of LR and HR patches. During the testing phase, the test patch is approximated by a linear combination of LR patches from the over complete dictionary; and because of the one-to-one mapping assumption, the corresponding HR patches are substituted instead [26], [27], [29], [48], [72]. However, the down-sampling, noise and blur operations that the HR patches went through to produce the LR patches are non-unique transformations and hence the one-to-one mapping assumption is not accurate. To mitigate some of the issues with prior work, our recent effort in this area was based on the assumption that the HR patches span a lower dimensional space, and hence, by projecting the test LR patch directly to the HR manifold, one may expect optimal approximations can be achieved [28], [73]. However, one key issue with this assumption is the problem of projecting between two spaces that are different in both geometry and dimension. We have observed that this difference in geometry and dimension can lead to inaccurate results. In particular, since the test LR patch is in general smooth; then, when projecting a LR patch into a HR manifold, the projection is steered and biased toward smooth subspaces of the HR manifold. Hence, some of the HR details are not being exploited. This issue limited the level of improvements that could be achieved. In addition, our prior work [28] used sparse subspace clustering (SSC), which is not guaranteed to be robust against noise [74]. Since one can't guarantee that the examples images are noise free, this can be another drawback for this approach. Finally, since in [28] the number of clusters is provided as an input, it may force patches that truly belong to different subspaces to be placed in the same cluster.

4.2 Subspace Clustering

Many applications in real-life deal with high dimensional data, for example in computer vision applications, a video can have thousands of frames. Same issue happens in machine learning, data mining, etc. Dealing with such high dimensional data decreases the performance of the algorithms and increase the computation time. This problem is known as the curse of dimensionality [75], [76].

For the past decades, several methods have been proposed to reduce the dimensionality of the data to avoid this problem. The most commonly used method is the Principal Component Analysis (PCA). The drawback of PCA is that it models the data into a single low-dimensional subspace, while in most applications the data need to be represented in multiple low-dimensional subspaces with known, or unknown, memberships to each subspace. Hence, we need to segment the data into multiple subspaces and fit each group into a low-dimensional subspace [74]. Subspace clustering algorithms can be categorized into:

- **Iterative:** Examples of these approaches are K-subspaces [77] and median K-flats which approximates data by a mixture of flats (subspaces) [78]. In these methods, data points are assigned to the closest subspace and subspaces are fitted into clusters. The problem with such algorithms is that they are sensitive to initialization values and they require knowledge about the dimensionality of the subspaces [75].

- **Factorization:** For example [79], [80]. These methods are usually based on segmenting the similarity matrix that is built from the data points. Although these methods do not require knowledge about the dimensionality of subspaces, they only work if the subspaces are independent, they are also sensitive to noise and outliers in the data [75].
- **Geometric:** For example [81] in which the subspaces are represented by a set of homogenous polynomials whose degree is the number of subspaces. A distance function is used to assign the data points into subspaces. The problems with this approach is that it is sensitive to noise and outliers and its computational complexity increases exponentially with the number and dimensions of subspaces [75].
- **Statistical:** Some of these methods are iterative, such as [82] which is based on the assumption that the distribution of data points in subspaces follow a Gaussian distribution. Similar to iterative methods, these algorithms require a knowledge of the number of subspaces and they are also sensitive to initialization [75]. Another robust statistical methods such as RANSAC [83] which assigns data points into subspaces until there is enough inlier data points. The problem with RANSAC is that the number and dimension of subspaces should be equal and known.

- **Spectral:** There are two main groups under this category, the local spectral clustering which uses local information of each data point to build a similarity matrix and then a spectral clustering algorithm is applied to segment the similarity matrix into subspaces. Examples of these approaches can be found in [84], [85]. The problem with these approaches is dealing with points that are close to two, or more, subspaces and they are require selecting the size of the neighborhood that they use to compute the distance between local data points.

  The other group is global spectral clustering, in which the similarity between all data points is used to assign the data points to subspaces. To capture the global relationship between data points, researchers rely on the concept of compressed sensing[1], [2] or matrix completion. Example of these methods is the Sparse Subspace Clustering (SSC)[75], which tries to enforce the sparsity assumption on the global similarity matrix that is built from all the data points. The benefit of these algorithms is that there is no need to know the number or dimension of subspaces.

4.3 Low Rank Subspace Clustering (LRSC)

LRSC belongs to the global spectral subspace clustering group and it is built on the foundation of Robust Principal Component Analysis (RPCA) that attempts to recover a low-rank component X and a sparse component E from a given observation matrix: $M = X + E$. The sparse matrix E can represent an error (or noise) signal that is corrupting the low-rank matrix X. This recovery can be achieved by minimizing the ℓ_1 norm of the error matrix (noise) and the nuclear norm $\|X\|_*$ of the data matrix X [10]:

$$\min_{X,E} \|X\|_* + \alpha\|E\|_1 + \frac{\lambda}{2} \|M - (X + E)\|_F^2 \tag{4.1}$$

In LRSC, the self-expressive property is assumed for the low-rank matrix $X = XC$. Hence, the observed data matrix M is assumed to be the graph obtained from using a noisy version of the final clean low rank adjacency matrix (C), in addition to additive sparse noise:

$$M = XC + E \tag{4.2}$$

The LRSC formula is obtained by substituting (4.2) for the RPCA equation (4.1) and adding a condition to maintain the symmetric property of C [74]:

$$\begin{gathered}\min_{C,X,E} \|C\|_* + \alpha\|E\|_1 + \frac{\lambda}{2} \|M - XC - E\|_F^2 \\ s.t.\ C = C^T\end{gathered} \tag{4.3}$$

Using the self-expressive property of X, i.e. $X = XC$ and substituting in (4.3), the minimization equation can be casted as [74]:

$$\begin{gathered}\min_{C,X,E} \|C\|_* + \alpha\|E\|_1 + \frac{\tau}{2}\|X - XC\|_F^2 + \frac{\lambda}{2} \|M - X - E\|_F^2 \\ s.t.\ C = C^T\end{gathered} \tag{4.4}$$

The output from (4.4) is the noise free adjacency matrix of the clustered graph (C^*) and the noise free data matrix (X^*).

4.4 Dual-manifold clustering and subspace similarity [86]

Similar to the previous example based methods, the proposed algorithm consists of two main phases; training phase and testing phase. Each phase is explained in details in the following sections.

4.4.1. Training Phase:

For training, the algorithm picks 1000 randomly generated LR patches of size 8x8 pixels. For each LR patch, the spatially equivalent HR patch, in addition to its 4 direct neighbors, are selected. The experimental results showed that this approach helps capturing not only the equivalent pixels locations from the HR patch, as in previous methods, but also the direct neighbor pixels that contributed to generating that patch during the down-sampling and blurring processes. In total we generate less than 6000 training patches.

A low-resolution patch (l) used for training is generated from a HR patch (h) by down-sampling, blurring and adding noise:

$$l = DBh + N \tag{4.5}$$

where D, B and N are the down-sampling, blur and additive noise matrices, respectively.

The LRSC algorithm is applied to both LR and HR patches, and hence, two clustering-based approximations of the LR and HR manifolds are generated independently. Here, it is important to emphasize that unlike previous algorithms, we don't assume any similarity in the geometry between the LR and the HR manifolds; instead, an affinity (similarity) measure (m) is used to find the closest HR subspace to the selected LR subspace.

$$s_{max} = max\, m(L_i, H_j) \tag{4.6}$$

where $1 \leq i \leq k_1$, $1 \leq j \leq k_2$. k_1, k_2 represent the number of LR and HR clusters respectively.

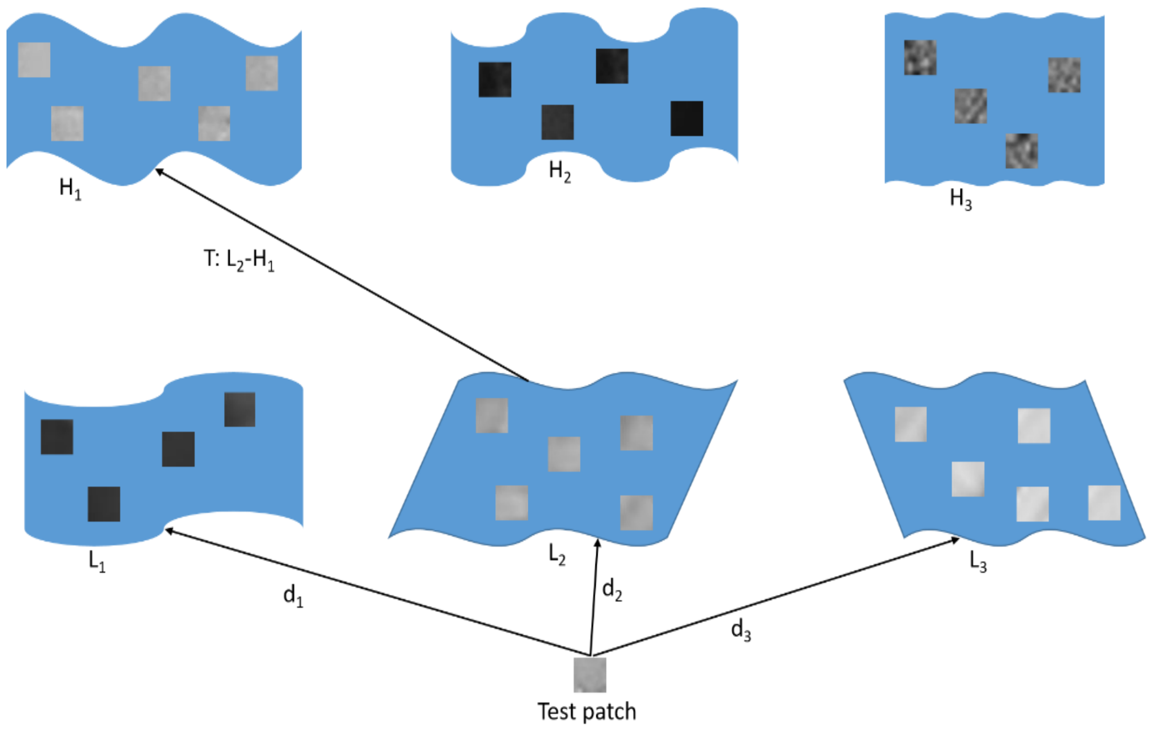


Figure 4.1: The test phase summary of the DMCSS algorithm. d_1,d_2 and d_3 represent the distance between the LR test patch and each cluster in the LR manifold. T:L_2-H_1 is an entry in the subspace similarity table (T) and it represents the closest HR subspace to the corresponding LR subspace.

The similarity between two subspaces L_i and H_j with PCA dimensions δ_i and δ_j is given by [87], [88]:

$$m(L_i, H_j) = \frac{r}{\sum_{k=1}^{r} \sigma_k} \tag{3.7}$$

where $r = \min(\delta_i, \delta_j)$, σ_k represents the k-th singular value of the orthonormal basis product ($P_i^T P_j$). $P_i \in \mathbb{R}^{\delta \times \delta_i}$ and $P_j \in \mathbb{R}^{\delta \times \delta_j}$ are the orthogonal basis matrices of L_i and H_j respectively. $\delta = 64$ is the length of the vector that results from the vectorization of each patch.

Since the similarity measure applies to subspaces with different dimensions, it fits quite well with our application.

$$\begin{gathered} P_i^T P_j = U\Sigma \mathrm{V}^T \\ \Sigma = diag(\sigma_1 \dots \sigma_r) \end{gathered} \tag{4.8}$$

As the experimental results show, using the similarity measure between LR and HR subspaces gives better approximation to the test image patches than projecting the patch directly to the HR manifold. This helps preventing the problem of selecting the wrong HR subspace in case there are two HR subspaces that are close to each other or in case of the availability of a smooth subspace.

Algorithm 4.1: The training and testing phases for the proposed DMCSS algorithm.

```
Algorithm 4.1:          DMCSS
-------------------------------------------------------------
Training Phase
Input: Training Images, patch size ($p$), $\tau, \alpha$ and $\lambda$
Build the manifolds L and H for LR and HR patches by random selection
    from training images.
Cluster L and H independently using (3.4).
For each LR subspace ($L_i$)
    Evaluate $m(L_i, H_j)$ to all HR clusters ($H_j$) using (3.7).
    Store the indices of the pair $(L_i, H_j)$ that satisfies (3.6) in T.
End For
-------------------------------------------------------------
Testing Phase
Input: T, a set of LR and HR clusters, $p$, overlap (o) and test image (G).
For each test patch $(g)$.
    Find the closest $L_i$ to $g$ using (3.9).
        Use T to find the corresponding $H_j$.
        Average the HR patches in $H_j$.
End for
-------------------------------------------------------------
Output:                 $G^*$
```

4.4.2. Testing Phase:

In this phase, the bicubic interpolation method is applied first to the test LR image to increase its size by the desired scaling factor. The image is then divided to 8x8 patches with 7 overlapping pixels. Finding the closest LR subspace to the test patch (g) can be simply done by projecting the test patch onto the LR subspaces (L_i) and selecting the subspace that produces the minimum distance.

$$d_{min} = min\, d(g, L_i) \tag{4.9}$$

where $1 \leq i \leq k,$ and k is the number of clusters.

The similarity table that was built in the training phase is then used to locate the closest HR subspace. The average of the patches in the HR cluster is used to find the final HR patch that is equivalent to the LR test patch. Similar to previous methods, the overlapped pixels in the final HR image are averaged and we added a condition to maintain the global smoothness of the result similar to [28], [89].

A demonstration of the testing phase is shown in Figure 4.1.The summary of the Dual-Manifold Clustering and Subspace Similarity (DMCSS) algorithm is shown in Algorithm 4.1.

4.5 Experimental Results

In this section the performance of DMCSS algorithm is compared against state-of-the-art super-resolution algorithms. We compared the algorithm against the standard MATLAB built-in bicubic interpolation function, the Adjusted Anchored Neighborhood Regression for Fast Super-Resolution algorithm (A+) [48], the sparse representation method (ScSR) [26] and SSC-based Linear Approximation of Manifolds algorithm (SLAM) [28], which provided superior results when compared. For the sake of fair comparison, we ran all algorithms using MATLAB R2012b on a computer with a 3.3 GHz i-5 CPU and a 4GB

Memory. Similar to the previous image super-resolution methods, we used the YC_BC_R image representation model.

Since the human eye is only sensitive to luminance, we used bicubic interpolation to scale up the C_B and C_R channels and applied the algorithm to the luminance channel only. We used the same training set in [28] and added 5 more extra images with different texture from [26]. The HR training set is shown in Figure 4.2.

Figure 4.3. shows the results of applying the super-resolution algorithms to scale up the Lena test image of size 85×85 by a magnification factor of 3. It clearly shows that our algorithm recovers most of the edges and details with more accuracy than the other algorithms. It also achieved about 0.9dB improvement in PSNR and 0.029 improvement in SSIM.

Next we applied the algorithms to other standard test images which are shown in Figure 4.4 and recorded the results as shown in Table 4.1. It also shows that DMCSS achieves an average of 1dB improvement in PSNR and over 0.02 SSIM improvement. Since most of these algorithms depend on random selection of training patches, we ran each algorithm for 20 times and took the average of the results.

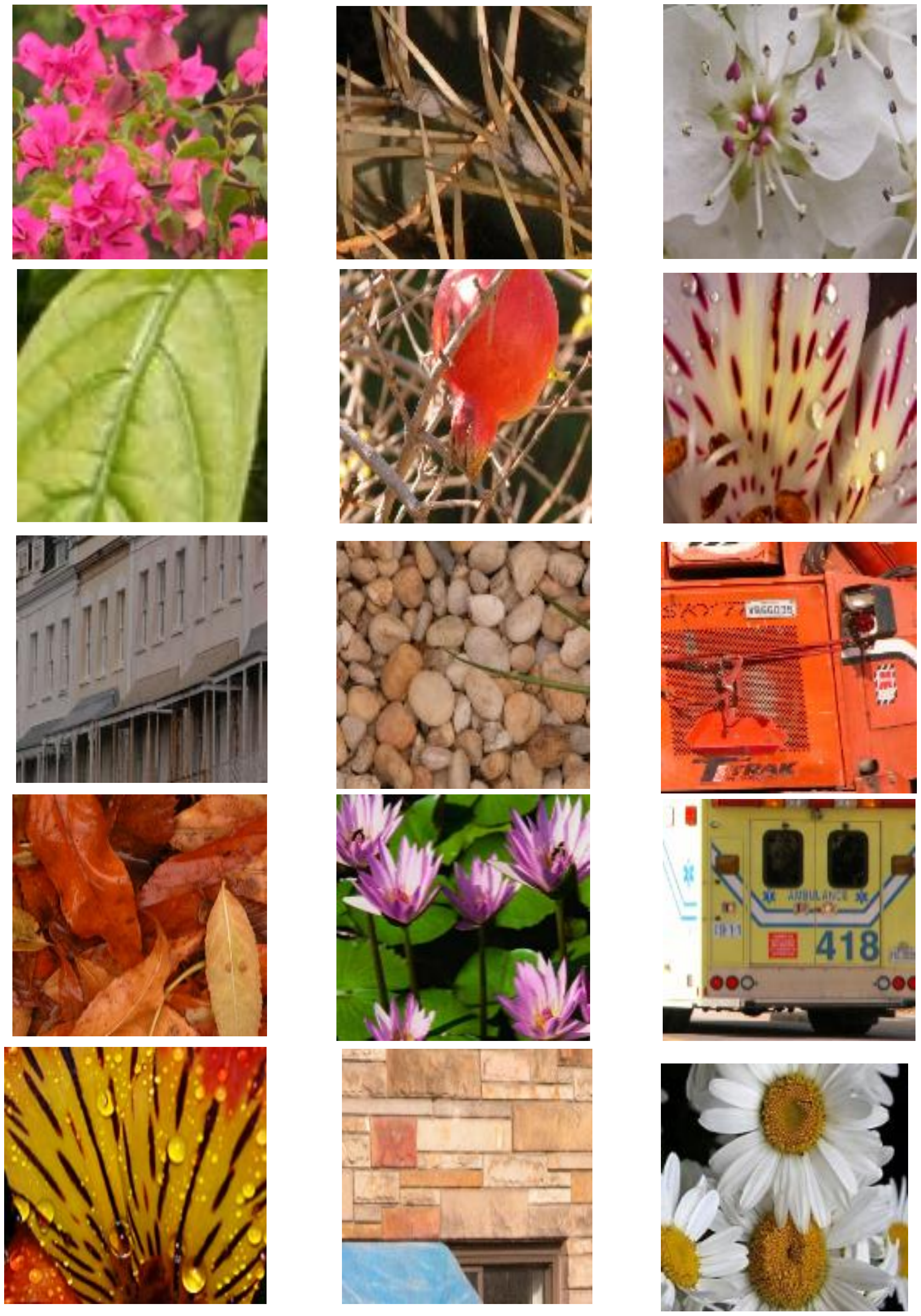


Figure 4.2: The training images used for the DMCSS algorithm. The pictures are proportionally resized to fit in the table.

(a) Bicubic PSNR:29.70, SSIM:0.836

(b) A+ PSNR:30.65, SSIM:0.852

(c) ScSR (PSNR:31.05, SSIM:0.861)

(d) SLAM (PSNR:31.30, SSIM:0.873)

(e) DMCSS (PSNR:32.11, SSIM:0.902)

(f) Original HR image

Figure 4.3: The results of applying different super-resolution methods to lena image with magnification factor of 3.

(a) Baby

(b) Raccoon

(c) Truck

(d) Freckles

(e) House

(f) Mountain

Figure 4.4: Testing images for super-resolution algorithms.

Table 4.1: Performance of super-resolution algorithms on different test images with a magnification factor of 3.

Image title	**Bicubic**	**A+**	**ScSR**	**SLAM**	**DMCSS**
Baby	31.73	32.54	32.49	32.98	**33.02**
	0.768	0.794	0.795	0.852	**0.896**
Raccoon	28.33	28.84	29.14	28.97	**30.43**
	0.715	0.758	0.771	0.762	**0.781**
Truck	27.43	27.76	28.39	28.31	**29.12**
	0.842	0.849	0.860	0.874	**0.892**
Freckles	30.14	30.83	31.05	31.46	**32.38**
	0.820	0.852	0.870	0.890	**0.901**
House	24.32	24.81	24.80	24.94	**26.62**
	0.685	0.694	0.731	0.730	**0.763**
Mountain	27.13	27.64	27.61	27.89	**28.79**
	0.710	0.745	0.746	0.762	**0.771**
Average	28.18	28.74	28.91	29.09	**30.06**
	0.757	0.782	0.796	0.812	**0.834**

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In the first part of this work, we have reviewed the main challenges associated with low rank matrix and tensor completion and robust tensor recovery problems, and described several state-of-the-art methods dealing with these problems and provided solutions to them based on the nuclear norm minimization. We proposed new algorithms for matrix completion using the smoothed Schatten-p function. Moreover, we extended the Schatten-p functions to the multi-dimensional tensor domain and proposed other algorithms for tensor completion and robust recovery. The experimental results showed, as expected, that in general all the proposed algorithms provide higher PSNR and better visual quality due to employing the truncated nuclear norm which leaves the most significant singular values untouched while minimizing the insignificant ones. TSRTR-ADMM achieves the highest PSNR but it has higher convergence time compared to TSRTR-ALM; on the other hand, TSRTR-APG is the fasted converging algorithm due to the acceleration of the gradient convergence, but its PSNR level falls below TSRTR-ALM. Several experiments have been carried out on three-dimensional and four-dimensional data. It has been shown that in general, our proposed framework performs much better, in terms of visual quality, PSNR and execution time, over the previous state-of-the-art algorithms. The reason for the quality improvement is due to selecting the truncated schatten-p function that approximated minimizing the rank of the low rank data more accurately than the traditional nuclear norm minimization function. The reason for the faster execution time is due to selecting more efficient and less complex optimization solutions to the Robust Tensor Recovery problem that require less extensive computation and converge faster than the other algorithms.

Based on the results obtained in this thesis and from an application point of view, we can categorize our algorithms as follow: if for a specific application timing is important and obtaining an acceptable quality results within small amount of time, then TSRTR-APG will be the best algorithm to be selected for that application. If for another application accurate results are required for the recovery no matter how much time it takes, then TSRTR-ADMM is the best option to select as it always achieves the highest PSNR. For general use and to provide very satisfying results within realistic timing, TSRTR-ALM satisfies that balance between execution time and quality.

The second part of this thesis is related to the ill-posed problem of image super-resolution. We reviewed the current main methods for image super-resolution and we introduced a new image super-resolution algorithm that makes no assumption about the structural similarity between the HR and LR manifolds. The LR and HR manifolds are trained independently and the subspace similarity measure is utilized to find the closest HR subspace to the selected LR subspace. Second, we used LRSC to prevent the effect of noise and gross errors on the algorithm performance. The experimental results showed that the algorithm achieves better results than state-of-the-art algorithms in this application using both subjective and objective measures.

5.2 Future works

5.2.1. Matrix and Tensor Completion and Robust Tensor Recovery

Throughout this thesis, the algorithms we proposed for Matrix and Tensor completion and Tensor RPCA problems are applied to recover small size visual data from few observed samples or highly corrupted multi-dimensional signals. For many applications, we usually deal with huge low rank datasets, such as in the NETFLIX problem, although our algorithms are designed to get rid of the SVD or HoSVD computations, it still very time consuming if applied to big datasets. For future works we will focus on reducing the computation time of these algorithms. Since matrix operations are separable and can run on different machines and/or parallel processed and the results can be combined, we will be looking into applying our algorithms into parallel processing environment by either applying them on a powerful parallel computing devices such as GPUs or distribute it on several computers and combine the results. This should help reducing the computation time of the algorithms even further.

5.2.2. Sparse\redundant representation for a single video frame

An interesting future direction is to adapt and apply our algorithm for is multi-frame image super-resolution, in which the training manifold will consist of different LR images of the same target image but under different conditions. Hence the HR manifold will be absent and we have to modify our algorithm to work for this scenario.

We will also attempt to improve and apply this method on video sequences and other applications that require real time results. One of the known issues when dealing with video signals is that each frame needs to be super-resolved in real time inorder for the human eye to watch the video smoothly without any interruption in the video; this makes it very crucial to make the algorithms that applies on videos to be run in real time. Since our test phase for each frame run in an average of around 1/3 of a second on our computer to super-resolve a colored image of size 85x85 by a factor of 3. This makes our algorithm applicable for only videos of 3 Frames Per Second (FPS) maximum, which is nowhere near a good quality video play rate of an average of 30-60 FPS. This is a great motivation for us to start looking into other ways to improve the speed of our algorithm using different methods such as doing fast interpolation to background and constant colored areas and pay more attention to edges and other borders that blur or makes block effect when using the normal interpolation methods. This should help reducing the number of batches being processed for super-resolution and it is promising to improve the running times of our algorithm.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Abdolreza Abdolhosseini Moghadam, “COMBINATORIAL METHODS FOR COMPRESSED SENSING,” Michigan State University, 2014.

[2] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[3] E. J. Cand, “The Restricted Isometry Property and Its Implications for Compressed Sensing,” pp. 1–4, 2008.

[4] E. Cand and J. Romberg, “Practical Signal Recovery from Random Projections,” vol. 2291, no. 626.

[5] E. J. Candès and B. Recht, “Exact matrix completion via convex optimization,” *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, 2009.

[6] P. Chen, D. Suter, and S. Member, “Recovering the Missing Components in a Large Noisy Low-Rank Matrix : Application to SFM,” vol. 26, no. 8, pp. 1051–1063, 2004.

[7] E. J. Candes and Y. Plan, “Matrix completion with noise,” *Proc. IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[8] E. J. Candès and T. Tao, “The power of convex relaxation: Near-optimal matrix completion,” *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[9] C. T. Dang., “Sparse and redundant models for data mining and consumer video summarization,” Michigan State University, 2015.

[10] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?,” *J. ACM*, vol. 58, no. 3, pp. 1–37, May 2011.

[11] G. Lerman and T. Zhang, “Report on p -Recovery of the most significant subspace among multiple subspaces with outliers , by Lerman and Zhang,” *Constr. Approx.*, vol. 40, no. 1, p. 5, 2014.

[12] X. Chen, Y. Qi, B. Bai, Q. Lin, and J. G. Carbonell, “Sparse latent semantic analysis,” *Proc. 11th SIAM Int. Conf. Data Mining, SDM 2011*, pp. 474–485, 2011.

[13] C.-P. Wei, C.-F. Chen, and Y. F. Wang, “Robust face recognition with structurally incoherent low-rank matrix decomposition.,” *Tip*, vol. 23, no. 8, pp. 3294–307, 2014.

[14] D. Goldfarb and Z. Qin, “Robust low-rank tensor recovery: Models and algorithms,” *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 1, pp. 225–253, 2014.

[15] Y. Li, Y. Zhou, J. Yan, J. Yang, and X. He, “Tensor error correction for corrupted values in visual data,” *Image Process. (ICIP), …*, 2010.

[16] H. Tan, B. Cheng, J. Feng, G. Feng, and Y. Zhang, “Tensor Recovery via Multi-linear Augmented Lagrange Multiplier Method,” *2011 Sixth Int. Conf. Image Graph.*, pp. 141–146, Aug. 2011.

[17] J. Yuan, S. Du, and X. Zhu, “Fast super-resolution for license plate image reconstruction,” *2008 19th Int. Conf. Pattern Recognit.*, pp. 1–4, 2008.

[18] F. Lin, C. Fookes, V. Chandran, and S. Sridharan, “Super-resolved faces for improved face recognition from surveillance video,” *Adv. Biometrics*, pp. 1–10, 2007.

[19] O. Le Meur and C. Guillemot, “Super-Resolution-based Inpainting,” *Eccv*, pp. 1–14, 2012.

[20] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, “Fast and robust multiframe super resolution,” *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1327–1344, 2004.

[21] S. S. Patil and M. M. Patil, “Multi Frame Iage Super Resolution,” vol. 4, no. 5, pp. 1686–1690, 2015.

[22] X. Li, Y. Hu, X. Gao, D. Tao, and B. Ning, “A multi-frame image super-resolution method,” *Signal Processing*, vol. 90, no. 2, pp. 405–414, 2010.

[23] C. Dang, M. Aghagolzadeh, and H. Radha, “Image Super-Resolution via Local Self-Learning Manifold Approximation,” *IEEE Signal Process. Lett.*, vol. 21, no. 10, pp. 1245–1249, 2014.

[24] Y. Tang and P. Yan, “Single-image super-resolution via local learning,” *Int. J. Mach. Learn.*, pp. 15–23, 2011.

[25] X. Lu, P. Yan, Y. Yuan, and X. Li, “Utilizing Homotopy for Single Image Superresolution,” pp. 316–320, 2011.

[26] J. Yang, J. Wright, Y. Ma, and T. Huang, “Image Super-Resolution as Sparse Representation of Raw Image Patches,” *IEEE Trans. Image Process.*, pp. 1–13, 2010.

[27] H. C. H. Chang, D.-Y. Y. D.-Y. Yeung, and Y. X. Y. Xiong, “Super-resolution through neighbor embedding,” *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2004. CVPR 2004.*, vol. 1, 2004.

[28] C. T. Dang, M. Aghagolzadeh, a. a. Moghadam, and H. Radha, “Single image super resolution via manifold linear approximation using sparse subspace clustering,” *2013 IEEE Glob. Conf. Signal Inf. Process.*, pp. 949–952, 2013.

[29] D. L. Karambelkar and P. J. Kulkarni, “Super-Resolution Using Manifold Learning,” *2011 Int. Conf. Comput. Intell. Commun. Networks*, pp. 707–710, 2011.

[30] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *Comput. Vision, 2009 IEEE 12th Int. Conf.*, pp. 2114–2121, Jan. 2009.

[31] S. Ma, D. Goldfarb, and L. Chen, “Fixed point and Bregman iterative methods for matrix rank minimization,” *Math. Program.*, vol. 128, no. 1–2, pp. 321–353, 2011.

[32] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[33] K. Toh and S. Yun, “An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems,” *Pacific J. Optim.*, vol. 6, pp. 615–640, 2010.

[34] D. Zhang, Y. Hu, Jj. Ye, X. Li, and X. He, “Matrix completion by truncated nuclear norm regularization,” *Comput. Vis. Pattern Recognit.*, pp. 2192–2199, Jun. 2012.

[35] K. Mohan and M. Fazel, “Iterative reweighted algorithms for matrix rank minimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 3441–3473, 2012.

[36] R. Tripathi, B. Mohan, and K. Rajawat, “Adaptive Low-Rank Matrix Completion,” vol. 65, no. 14, pp. 3603–3616, 2017.

[37] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–20, Jan. 2013.

[38] M. Yuan and C. Zhang, “On Tensor Completion via Nuclear Norm Minimization,” 2014.

[39] L. Huang and H. So, “Truncated nuclear norm minimization for tensor completion,” *Sens. Array …*, pp. 417–420, Jun. 2014.

[40] T. Yokota and A. Cichocki, “TENSOR COMPLETION VIA FUNCTIONAL SMOOTH COMPONENT DEFLATION,” no. 1, pp. 2–6, 2016.

[41] J. A. Bengua, H. N. Phien, H. D. Tuan, and M. N. Do, “Efficient Tensor Completion for Color Image and Video Recovery : Low-Rank Tensor Train,” vol. 26, no. 5, pp. 2466–2479, 2017.

[42] J. Wright, A. Ganesh, and S. Rao, “Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization,” *Proc. Neural Inf. Process. Syst.*, vol. 3, 2009.

[43] Q. Zhao, D. Meng, Z. Xu, W. Zuo, and L. Zhang, “Robust Principal Component Analysis with Complex Noise,” 2010.

[44] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, “Tensor Robust Principal

Component Analysis : Exact Recovery of Corrupted Low-Rank Tensors via Convex Optimization," 2016.

[45] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[46] F. Changjun, J. Xiangyang, Z. Yongbing, and D. Qionghai, "A Single Frame Super-Resolution Method Based on Matrix Completion," *Data Compression Conf.*, pp. 297–306, Apr. 2012.

[47] W. Huang, L. Xiao, Z. Wei, X. Fei, and K. Wang, "Single image super-resolution by clustered sparse representation and adaptive patch aggregation," *China Commun.*, vol. 10, no. 5, pp. 50–61, 2013.

[48] R. Timofte, V. De Smet, and L. Van Gool, "A plus : Adjusted Anchored Neighborhood Regression for Fast Super-Resolution," *Comput. Vis. - Accv 2014, Pt Iv*, vol. 9006, pp. 111–126, 2015.

[49] R. Timofte, V. De, and L. Van Gool, "Anchored Neighborhood Regression for Fast Example-Based Super-Resolution," *Comput. Vis. (ICCV), 2013 IEEE Int. Conf.*, pp. 1920–1927, 2013.

[50] C. Yang and M. Yang, "Fast Direct Super-Resolution by Simple Functions," *IEEE Int. Conf. Comput. Vis.*, 2013.

[51] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *Optim. Online*, vol. 52, no. 3, pp. 1–33, 2007.

[52] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Front. Math. China*, vol. 7, no. 2, pp. 365–384, 2012.

[53] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific J. Optim.*, vol. 6, no. 15, pp. 615–640, 2010.

[54] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *J. Mach. Learn. Res.*, vol. 99, pp. 2057–2078, 2010.

[55] M. Fazel, "Matrix rank minimization with applications," Stanford University, 2002.

[56] R. S. Cabral, F. Torre, J. P. Costeira, and A. Bernardino, "Matrix completion for multi-label image classification," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2011.

[57] F. Fazel, M. Fazel, and M. Stojanovic, "Random access sensor networks: Field reconstruction from incomplete data," *2012 Inf. Theory Appl. Work.*, pp. 300–305, Feb. 2012.

[58] K. Xie, X. Ning, X. Wang, D. Xie, J. Cao, G. Xie, and J. Wen, “Recover Corrupted Data in Sensor Networks : A Matrix Completion Solution,” vol. 16, no. 5, pp. 1434–1448, 2017.

[59] Z. Liu and L. Vandenberghe, “Interior-point method for nuclear norm approximation with application to system identification,” *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 3, pp. 1235–1256, Jan. 2009.

[60] I. Daubechies and R. DeVore, “Iteratively reweighted least squares minimization for sparse recovery,” *Commun. Pure Appl. Math.*, vol. 63, no. 1, pp. 1–38, 2010.

[61] K. Mohan and M. Fazel, “Reweighted nuclear norm minimization with application to system identification,” *Am. Control Conf. (ACC), 2010*, no. 5, 2010.

[62] M. Signoretto, V. de P. Raf, D. M. Bart, and J. A. Suykens, “Tensor versus matrix completion: a comparison with application to spectral data,” *Signal Process. Lett.*, vol. 18, no. 7, pp. 403–406, Jul. 2011.

[63] Z. Lin, M. Chen, and Y. Ma, “The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices,” *arXiv Prepr. arXiv1009.5055*, 2010.

[64] S. Gandy, B. Recht, and I. Yamada, “Tensor completion and low-n-rank tensor recovery via convex optimization,” *Inverse Probl.*, vol. 27, no. 2, 2011.

[65] S. Friedland and L. Lim, “Nuclear norm of higher-order tensors,” pp. 1–23, 2016.

[66] Z. Lin, M. Chen, and Y. Ma, “The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices,” *arXiv Prepr. arXiv1009.5055*, 2010.

[67] M. Al-Qizwini and H. Radha, “Fast smooth rank approximation for tensor completion,” *48th Annu. Conf. Inf. Sci. Syst.*, no. 4, pp. 1–5, Mar. 2014.

[68] M. Al-Qizwini and H. Radha, “Smoothed Rank Approximation Algorithms for Matrix Completion,” *Asilomar*, 2014.

[69] W. Zuo and Z. Lin, “A generalized accelerated proximal gradient approach for total-variation-based image restoration.,” *Image Process. IEEE Trans.*, vol. 20, no. 10, pp. 2748–2759, Oct. 2011.

[70] M. Al-Qizwini and H. Radha, “Truncated and Smoothed Schatten-p Function for Robust Tensor Recovery,” *Conf. Inf. Sci. Syst.*, pp. 31–34, 2015.

[71] Z. Shi, T. Zheng, and J. Han, “Trace norm regularized tensor classification and its online learning approaches,” *arXiv Prepr. arXiv1109.1342*, pp. 1–11, 2011.

[72] T. Peleg, S. Member, and M. Elad, “A Statistical Prediction Model Based on Sparse Representations for Single Image Super-Resolution,” *IEEE Trans. Image Process.*, vol. 23, no. 6, pp. 2569–2582, 2014.

[73] C. Dang and H. Radha, “Fast Image Super-resolution via Selective Manifold Learning of High-Resolution Patches,” pp. 1319–1323, 2015.

[74] R. Vidal and P. Favaro, “Low rank subspace clustering (LRSC),” *Pattern Recognit. Lett.*, vol. 43, no. 1, pp. 47–61, 2014.

[75] S. Member, E. Elhamifar, and S. Member, “Sparse Subspace Clustering :,” pp. 1–19.

[76] R. Bellman, *Dynamic Programming*. new jersey: Princeton University Press Princeton., 1957.

[77] P. Tseng, “Nearest q-Flat to m Points,” *J. Optim. Theory Appl.*, vol. 105, no. 1, pp. 249–252, 2000.

[78] J. Yang, S. Member, J. Wright, and T. S. Huang, “Image Super-Resolution Via Sparse Representation,” vol. 19, no. 11, pp. 2861–2873, 2010.

[79] P. Costeira, “A Multibody Factorization Method for Independently Moving Objects,” vol. 29, no. 3, pp. 159–179, 1998.

[80] K. Kanatani, “Motion Segmentation by Subspace Separation and Model Selection Kenichi,” pp. 586–591, 2001.

[81] Y. Ma and S. Sastry, “Generalized Principal Component Analysis (GPCA),” vol. 27, no. 12, pp. 1945–1959, 2005.

[82] C. M. Bishop, “Mixtures of Probabilistic Principal Component Analyzers,” vol. 482, no. 1901, pp. 443–482, 1999.

[83] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography,” vol. 24, no. 6, 1981.

[84] A. Goh, “Segmenting Motions of Different Types by Unsupervised Manifold Clustering,” 2007.

[85] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, “Hybrid Linear Modeling via Local Best-Fit Flats,” no. May, pp. 217–240, 2012.

[86] M. Al-Qizwini, C. Dang, M. Aghagolzadeh, and H. Radha, “Image Super-resolution via Dual-Manifold Clustering and Subspace Similarity,” *Int. Conf. Image Process.*, vol. 683, 2016.

[87] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, “Robust subspace clustering,” *Ann. Stat.*, vol. 42, no. 2, pp. 669–699, 2014.

[88] R. Wang, S. Shan, X. Chen, and W. Gao, “Manifold-manifold distance with application to face recognition based on image set,” *Comput. Vis. Pattern*

Recognition, 2008. CVPR 2008. IEEE Conf., pp. 1–8, 2008.

[89] M. Irani and S. Peleg, “Super resolution from image sequences,” *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, vol. ii. pp. 115–120 vol.2, 1990.