SRF: MATRIX COMPLETION BASED ON SMOOTHED RANK FUNCTION

Hooshang Ghasemi[1], Mohmmadreza Malek-Mohammadi[1], Massoud Babaie-Zadeh[1], Christian Jutten[2]

[1]Sharif University of Technology, Department of Electrical Engineering, Tehran, Iran
[2]GIPSA-lab, Grenoble, and Institut Universitaire de France, France
Emails: hooshang.ghasemi@yahoo.com, m.rezamm@ieee.org, mbzadeh@yahoo.com, Christian.Jutten@inpg.fr

ABSTRACT

In this paper, we address the matrix completion problem and propose a novel algorithm based on a smoothed rank function (SRF) approximation. Among available algorithms like FPCA and OptSpace, there is no solution that can simultaneously cover wide range of easy and hard problems. This new algorithm provides accurate results in almost all scenarios with a reasonable run time. It especially has low execution time in hard problems where other methods need long time to converge. Furthermore, when the rank is known in advance and is high, our method is very faster than previous methods for the same accuracy. The main idea of the algorithm is based on a continuous and differentiable approximation of the rank function and then, using gradient projection approach to minimize it.

Index Terms— Matrix completion, nuclear norm, Compressed Sensing, Sparse Signal Processing

1. INTRODUCTION

Matrix completion can be treated as a generalization of compressive sensing of vectors which has been attracting a lot of researchers during the last decade. Matrix completion arises in wide range of applications including: *collaborative filtering*, *triangulation from incomplete data* and *linear system identification* [1, 2]. Consider a matrix that only few of its entries are known and we would like to exactly recover the matrix based on the revealed entries. It is clear that solving this problem is not generally possible. Now, suppose that the matrix is of dimension $n_1 \times n_2$ and of rank r. It is easy to show that degree of freedom of this matrix is equal to $r(n_1 + n_2 - r)$. So, to exactly recover the matrix, it is necessary that the number of revealed entries be more than this degree of freedom. Consequently, recovery of the matrix from sufficient number of known entries is possible if the matrix is low-rank and entries are selected uniformly [1]. Indeed, the issue of matrix completion is to set a matrix with minimum rank to a given subset of entries. If the number of revealed entries is sufficiently large and the entries have been sampled sufficiently uniformly, then the setting is unique.

This work has been partially supported by Iran Telecommunication Research Center (ITRC) and Iran NSF (INSF).

Recovery of a matrix by means of minimizing rank function and constraints on revealed entries is an NP-hard problem that stems from non-convexity and discontinuously of the rank function [3]. Complexity and time required to solve this problem grow double exponentially by matrix dimensions [3]. To overcome non-convexity of the rank function, Fazel proposed using nuclear norm instead of the rank function [4]. The nuclear norm of a matrix is the sum of singular values of the matrix and is a convex envelope of the rank function [4]. Surprisingly, the relation between nuclear norm and rank of a matrix is similar to relationship between l_1 norm and l_0 norm of a vector [2]. So, the matrix completion problem would be converted to recovery of the matrix by means of minimizing the nuclear norm and constraints on revealed entries. Solution of nuclear norm minimization might completely differ from the solution of minimizing rank function. However, Candès and Retch [1] show that under incoherency of column and row spaces of a matrix with standard basis, if the number of revealed entries is large enough, then the solution of minimizing nuclear norm is unique and, with a high probability, is equal to the solution of minimizing rank function.

Most of the algorithms such as FPCA [5], OptSpace [6] and SDP (Semi-Definite Programming) [4] use nuclear norm heuristic to solve matrix completion problem. FPCA is appropriate for hard problems where the number of revealed entries is close to the degree of freedom. But, for easy problems, it is inaccurate. On the other hand, OptSpace is suitable for easy problems, especially problems where the rank of the matrix is very low.

In this paper, we propose a new approach based on a continuous and differentiable approximation of the discontinuous rank function. Our algorithm has two main advantages in comparison to the other ones. Firstly, when the rank of the matrix is known and high, this algorithm is faster than all previous methods. Secondly, it achieves more accurate results in almost all situations. Our work is inspired by the work of Mohimani *et al* [7] that uses smoothed l_0-norm to obtain sparse solutions of underdetermined system of linear equations.

This paper is organized as follows. In section 2, we formulate the problem and in section 3, propose the novel

algorithm. Section 4 presents some simulation results and section 5 concludes the paper.

2. MATRIX COMPLETION PROBLEM

Consider an unknown matrix $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ of rank r for which only m entries of it have been revealed (in this paper we examine the case where revealed entries are noiseless). Let m be sufficiently large and locations of observed entries are sufficiently uniformly distributed. As explained in the introduction, matrix completion problem is to find a matrix $\boldsymbol{X_0}$ that is a solution to the optimization problem:

$$\begin{aligned} &\min rank(\boldsymbol{X}) \\ &s.t \;\; \boldsymbol{X_{ij}} = \boldsymbol{M_{ij}}, \quad (i,j) \in \Omega, \end{aligned} \tag{1}$$

where Ω is the set of locations corresponding to observed entries and $|\Omega| = m$, where $|\cdot|$ denotes the cardinality of a set, and $rank(\boldsymbol{X})$, the rank function, is the number of non-zero singular values of $\boldsymbol{X}$. As mentioned, because of non-convexity of the rank function, nuclear norm heuristic is used to solve the matrix completion problem. Nuclear norm of a matrix is $\|\boldsymbol{X}\|_* = \sum_{k=1}^{n} \sigma_k(\boldsymbol{X})$, where $\sigma_k(\boldsymbol{X})$ denotes the k-th largest singular value of matrix $\boldsymbol{X}$ and $n = \min(n_1, n_2)$. By using the nuclear norm definition, non-convex optimization problem (1) can be reduced to the convex problem:

$$\begin{aligned} &\min \;\; \|\boldsymbol{X}\|_* \\ &s.t. \;\; \boldsymbol{X_{ij}} = \boldsymbol{M_{ij}}, \quad (i,j) \in \Omega. \end{aligned} \tag{2}$$

Candès and Retch [1] show that for random matrices there are numerical constants C, c such that if the number of revealed entries of the matrix $\boldsymbol{M}$ is larger than $Cn^{5/4} r \log n$, then with probability at least $1 - cn^{-3}$ the minimizer to the problem (2) is unique and equal to $\boldsymbol{M}$.

3. PROPOSED ALGORITHM

3.1. Main Idea

Our main idea to solve matrix completion problem is to use a continuous approximation of the rank function and then to apply continuous optimization techniques to minimize it. Let $F: \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$ be any continuous approximation of the rank function, then equation (1) can be written as:

$$\begin{aligned} &\min F(\boldsymbol{X}) \\ &s.t. \;\; \boldsymbol{X_{ij}} = \boldsymbol{M_{ij}}, \quad (i,j) \in \Omega. \end{aligned} \tag{3}$$

If $F(\boldsymbol{X})$ is differentiable, then it can be used to solve (3) using a gradient projection approach. One of such functions is the Gaussian function that is used in [7],

$$F_\delta(\boldsymbol{X}) = n - \sum_{k=1}^{n} e^{-\sigma_k^2(\boldsymbol{X})/2\delta^2}. \tag{4}$$

When δ approaches to zero in (4), $F_\delta(\cdot)$ will approach to the rank function. But decreasing δ to zero will result in a highly non-smooth $F_\delta(\cdot)$ with many local maxima, and the gradient projection method might be captured in local minima and fail to converge. In contrast, for large values of δ, the approximation of the rank function will not be accurate enough. To overcome this problem, we will gradually decrease δ in each iteration. This technique, used for minimizing non-convex functions, is referred to as GNC (Graduated Non-Convexity) [8]. For a given δ, this technique uses the minimizer of the previous iteration (larger δ) as the new starting point to search for the minimizer of this smaller δ. Theoretically, the initial value of δ is infinite but practically three or four times of the largest singular value of an initialization matrix is enough because it virtually acts as infinity in the exponential functions in (4). Now, we are looking for the gradient of the approximating function, $F_\delta(\cdot)$.

Theorem 1: For matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ with singular value decomposition (SVD) $\boldsymbol{X} = \boldsymbol{U}\text{Diag}\{\sigma_1, \sigma_2, \cdots, \sigma_n\}\boldsymbol{V}^H$, the gradient of $F_\delta(\boldsymbol{X})$ is

$$\frac{\partial F_\delta(\boldsymbol{X})}{\partial \boldsymbol{X}} = \boldsymbol{U}\text{Diag}\{{}^{\sigma_1}\!/_{\delta^2}\, e^{-\sigma_1^2/2\delta^2}, \cdots, {}^{\sigma_n}\!/_{\delta^2}\, e^{-\sigma_n^2/2\delta^2}\}\boldsymbol{V}^H. \tag{5}$$

Proof: Let represent $F_\delta(\cdot)$ as $F_\delta(\boldsymbol{X}) = f\big(\boldsymbol{\sigma}(\boldsymbol{X})\big) = f \circ \boldsymbol{\sigma}(\boldsymbol{X})$ where $f: \mathbb{R}^n \to \mathbb{R}$ is defined as $f(\boldsymbol{y}) = n - \sum_{k=1}^{n} e^{-y_k^2/2\delta^2}$ and $\boldsymbol{\sigma}(\boldsymbol{X}): \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^n$ has the singular values of the matrix $\boldsymbol{X}$. In [10, Cor 2.5], it is shown that if a function f is *absolutely symmetric* (a function $f(\boldsymbol{y})$ is called absolutely symmetric if $f(\boldsymbol{y})$ is invariant under arbitrary permutations and sign changes of the components of $\boldsymbol{y}$) and the matrix $\boldsymbol{X}$ has $\boldsymbol{\sigma}(\boldsymbol{X})$ in the domain of f, then the sub-differential of $F_\delta(\cdot)$ is given by:

$$\partial(f \circ \boldsymbol{\sigma}(\boldsymbol{X})) = \big\{\boldsymbol{U}(\text{Diag}\boldsymbol{\mu})\boldsymbol{V}^H \big| \boldsymbol{\mu} \in \partial f\big(\boldsymbol{\sigma}(\boldsymbol{X})\big)\big\}. \tag{6}$$

Since $f\big(\boldsymbol{\sigma}(\boldsymbol{X})\big)$ is differentiable at $\boldsymbol{\sigma}(\boldsymbol{X})$, $\partial f\big(\boldsymbol{\sigma}(\boldsymbol{X})\big)$ is a singleton and consequently $\partial(f \circ \boldsymbol{\sigma}(\boldsymbol{X}))$ becomes a singleton. Therefore, $\partial(f \circ \boldsymbol{\sigma}(\boldsymbol{X}))$ is converted to $\nabla(f \circ \boldsymbol{\sigma}(\boldsymbol{X}))$ and equation (5) is obtained (the theorem results also from the "if part" of [10, Thm 3.1], which does not require convexity of f as stated in its proof). ■

3.2. Gradient Projection

Let $\mathcal{A}: \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ be a projection operator such that for a given matrix $\boldsymbol{X}$,

$$\mathcal{A}(\boldsymbol{X}) = \begin{cases} M_{ij}, & (i,j) \in \Omega \\ X_{ij}, & (i,j) \notin \Omega \end{cases}$$

So, the gradient projection at the k-th iteration will be,

$$\boldsymbol{X}_{k+1} \leftarrow \mathcal{A}\big(\boldsymbol{X}_k - \mu_k \nabla F_\delta(\boldsymbol{X}_k)\big), \tag{7}$$

where μ_k is the step size at the k-th step. Now, by considering the gradient given in (5), gradient projection method can be represented as the following two steps:

Step 1: Gradient,

$$\begin{aligned} \widetilde{\boldsymbol{X}}_{k+1} \leftarrow \boldsymbol{U}_k Diag\Big\{&\sigma_1 - \mu_k\, {}^{\sigma_1}\!/_{\delta^2}\, e^{-\sigma_1^2/2\delta^2}, \cdots, \sigma_n - \\ &\mu_k\, {}^{\sigma_n}\!/_{\delta^2}\, e^{-\sigma_n^2/2\delta^2}\Big\}\boldsymbol{V}_k{}^H. \end{aligned} \tag{8}$$

Step 2: Projection,

$$\boldsymbol{X}_{k+1} \leftarrow \mathcal{A}(\widetilde{\boldsymbol{X}}_{k+1}), \quad (9)$$

where $\boldsymbol{X}_k = \boldsymbol{U}_k Diag\{\sigma_1, \sigma_2, \cdots, \sigma_n\} {\boldsymbol{V}_k}^H$ is the SVD of the matrix $\boldsymbol{X}_k$.

3.3. Finalized Algorithm

Before finalizing the algorithm, two remaining points should be clarified. Firstly, the initial solution of the algorithm is a solution of (3) with $F_\delta(\boldsymbol{X})$ defined in (4) when δ approaches infinity. We claim that when δ goes to infinity every full rank matrix satisfying the constraints of (3) is a solution of the minimization problem (3). Let $\boldsymbol{A}$ be a full rank matrix that fulfills the constraints of (3). Since all singular values of $\boldsymbol{A}$ are non-zero, $e^{-\sigma_k^2(\boldsymbol{X})/2\delta^2}$ in equation (4) approaches to one as δ goes to infinity. Therefore, $F_\delta(\boldsymbol{A})$ will be zero for infinite values of δ. It is easy to seen that $F_\delta(\boldsymbol{A}) \geq 0$, hence $\boldsymbol{A}$ is a solution of (3).

Secondly, closeness of the approximated matrices in two consecutive iterations is our criterion to stop the algorithm. In our simulations, we declared that algorithm has been finished if $\|\boldsymbol{X}_{k+1} - \boldsymbol{X}_k\|_F / \sqrt{n_1 n_2}$ is smaller than a given threshold which we call *error threshold*.

Putting all together, it results in the algorithm shown in Table 1, called smoothed rank function (SRF) algorithm.

4. NUMERICAL SIMULATIONS

In this section, the performance of the SRF algorithm is experimentally evaluated and is compared to other algorithms. For generating a testing random matrix $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ of rank r in our simulations, we used the following procedure. We generate two random matrices $\boldsymbol{M}_L \in \mathbb{R}^{n_1 \times r}$ and $\boldsymbol{M}_R \in \mathbb{R}^{r \times n_2}$ whose entries are independent and identically drawn from a Gaussian distribution with zero mean and unit variance and then, construct the matrix $\boldsymbol{M} = \boldsymbol{M}_L \cdot \boldsymbol{M}_R$. Revealed entries index set, Ω, is selected randomly in a uniform manner. The final estimated matrix is denoted by $\widetilde{\boldsymbol{M}}$ and to measure the closeness of the estimated matrix to the original matrix, we use *relative error* defined as $\|\boldsymbol{M} - \widetilde{\boldsymbol{M}}\|_F / \|\boldsymbol{M}\|_F$.

We used run time of the algorithms and their relative errors as two criteria to compare their performances. Although CPU time is not an accurate estimate of the run time, we use it as a rough estimation to compare algorithm complexities. Our simulations were performed in MATLAB® 7.6 environment using an Intel® i5 M520, 2.4GHz processor with 4GB of RAM, under Microsoft Windows® 7 operating system. We run every simulation 20 times with different randomly generated $\boldsymbol{M}$, and results are averaged run time and relative error. In our simulations, we used $n_1 = n_2 = n$. In tables, we denote *Error* and d_r, the relative error and degree of freedom, respectively. Moreover, by the term *easy problems*, we mean problems in which the ratio m/d_r is larger than 3, otherwise problems are called *hard problems*.

Our algorithm's input parameters are the number of iterations of the inner loop, error threshold, and decreasing factor of δ and their default values are 8, 10^{-5}, and 0.9 respectively. However, for easy problems, we set the decreasing factor of δ equal to 0.4 to decrease computational time. To obtain a given relative error, it is sufficient to set the error threshold approximately equal to the expected relative error. Experimentally, the number of iterations of the inner loop, L, is chosen equal to 8. For larger values of L, obtained matrix will be more accurate although this increases the run time.

Moreover, as explained in the previous section, we need to start the algorithm with a full-rank initialization matrix $\boldsymbol{M}_E$ satisfying the constraints of (3). In our simulations, we used the following initialization matrix:

$$\boldsymbol{M}_E = \begin{cases} M_{ij}, & (i,j) \in \Omega \\ 0, & (i,j) \notin \Omega \end{cases}$$

which was always full-rank with our randomly generated matrices.

Some algorithms inherently need to know the rank of the original matrix and it is given as input or it is approximated by the algorithm. For example, see OptSpace and ADMiRA [9]. In our algorithm, if the rank is known, we can exploit it by doing a partial singular value decomposition instead of a complete one (in MATLAB, this can be 'svds' function instead of 'svd' function), and update only this singular values in (8). This will result in decreasing the computational load.

4.1. Unknown Rank Scenario

In this scenario, the rank of original matrix is unknown. Algorithms like FPCA and SRF inherently do not need to know the rank and never estimate it, but OptSpace algorithm needs to estimate the rank before starting. In this part, we run algorithms by their default parameters. Table 2 shows some results of this comparison. In all cases, SRF has more

Table 1. Pseudo code of the SRF Algorithm

Algorithm 1: Smoothed Rank Function (SRF)

Inputs: $\boldsymbol{M}_E$ (initialization matrix), dec_fac, $er\_threshold$, L, Ω.
Output: $\widetilde{\boldsymbol{M}}$.
Initialize: δ = largest singular value of $\boldsymbol{M}_E$, $d = 0$, $\boldsymbol{X} = \boldsymbol{M}_E$.
While $d > er\_threshold$
  Let $\boldsymbol{Y} = \boldsymbol{X}$.
  Inner loop:
  For $i = 1{:}\,L$
    Compute SVD of $\boldsymbol{X} = \boldsymbol{U} Diag\{\sigma_1, \sigma_2, \cdots, \sigma_n\} \boldsymbol{V}^H$
    $\widetilde{\boldsymbol{X}} = \boldsymbol{U} Diag\{\sigma_1 - \mu\, \sigma_1 / \delta^2\, e^{-\sigma_1^2/2\delta^2}, \cdots, \sigma_n - \mu\, \sigma_n / \delta^2\, e^{-\sigma_n^2/2\delta^2}\} \boldsymbol{V}^H$
    Compute projection $\boldsymbol{X} \leftarrow \mathcal{A}(\widetilde{\boldsymbol{X}})$.
  End
  Compute $\delta = \delta \times dec\_fac$, $d = \|\boldsymbol{Y} - \boldsymbol{X}\|_F / \sqrt{n_1 n_2}$.
End
$\widetilde{\boldsymbol{M}} = \boldsymbol{X}$.

Table 2. Simulation results of easy problems

Problem	SRF		OptSpace		FPCA	
(n, r, m/d_r)	Time	Error	Time	Error	Time	Error
(50,5,4)	0.34	1.29e-8	0.29	6.18e-2	0.38	2.15e-2
(100,2,10)	0.93	1.49e-8	0.35	8.37e-8	0.12	4.50e-5
(100,5,3.3)	2.06	4.36e-8	0.77	1.24e-1	0.16	1.69e-3
(100,10,5)	1.68	2.89e-9	1.99	2.79e-8	1.24	7.67e-1
(100,10,3.3)	1.52	3.49e-9	2.62	1.37e-1	0.32	4.84e-5
(200,10,4)	7.90	4.92e-9	9.12	8.19e-7	0.57	8.31e-5
(200,20,3)	9.38	1.83e-9	45.3	5.60e-2	0.81	3.32e-5
(300,10,3.3)	36.9	3.68e-7	19.5	4.62e-2	1.6	2.40e-3
(300,10,4)	27.8	1.70e-7	19.4	2.27e-6	1.01	1.82e-4

Table 3. Simulation results of hard problems

Problem	SRF		OptSpace		FPCA	
(n, r, m/d_r)	Time	Error	Time	Error	Time	Error
(50,10,2)	0.36	7.78e-8	0.53	4.35e-1	3.23	8.44e-7
(100,5,2.5)	4.16	1.01e-5	0.52	4.69e-1	4.8	7.14e-6
(100,10,2.5)	1.8	1.52e-7	1.97	3.77e-1	10.1	1.98e-6
(100,10,2)	3.11	6.49e-7	1.10	6.36e-1	8.28	3.86e-6
(100,10,1.7)	6.82	5.26e-5	0.63	8.09e-1	7.52	6.62e-6
(100,20,2.5)	1.93	4.35e-9	12.4	2.66e-1	12.0	7.87e-8
(100,20,2)	1.81	2.24e-8	2.55	7.69e-1	11.5	6.03e-7
(100,20,1.7)	2.25	2.06e-7	1.41	7.88e-1	10.9	1.76e-6
(300,10,2.5)	69.2	1.12e-6	19.5	1.51e-2	74.7	7.17e-6

accurate results in comparison to other algorithms. With respect to OptSpace algorithm, besides the better accuracy, SRF begins to outperform OptSpace in computation time when the matrix dimensions increase. In Table 3, we show some results of hard problems. As seen, OptSpace algorithm fails in all cases. SRF has the lowest run time in all mentioned simulations and is more accurate than FPCA in most cases.

4.2. Known Rank Scenario

In this scenario, we set parameters of all algorithms such that relative error be in order of 10^{-6} and then we will compare their computation times. This is showed in figure 1 and as can be seen, for ranks greater than 4, SRF algorithm has a lower run time. For comparison, we use a matrix of dimension 200×200 where only 20,000 randomly chosen entries (50% of the total entries) of it are revealed.

We declare that when the rank is unknown and the problem is easy our method achieves the most accurate results, while in hard problems it is the fastest among the simulated algorithms. Also, it was shown that for the same relative error, if the rank is known and is more than 2% of the matrix dimension, SRF is faster than others.

5. CONCLUSION

In this work, a new approach for solving matrix completion problem was introduced. After presenting the main idea, we experimentally compared the proposed method with OptSpace and FPCA methods, two well known methods in matrix completion literature. Thorough simulations, we showed that the novel algorithm is more precise whether the problem is easy or hard. In addition, when the rank is known and high, for the same reconstruction performance, our algorithm is faster than the others. Extension of this idea to noisy matrix completion problem, where the noise is Gaussian or spiky, is currently under study in our group.

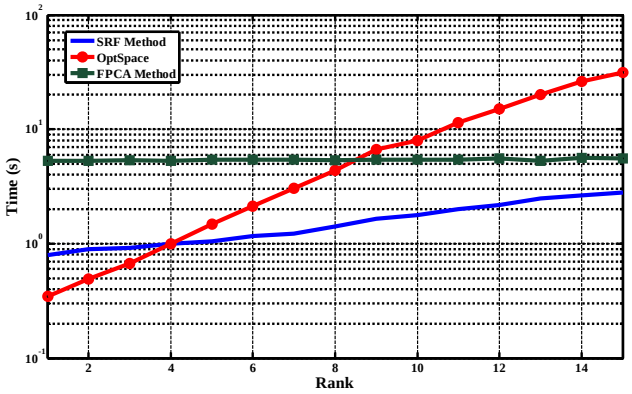

Fig. 1 Execution time vs. rank

6. REFERENCES

[1] E.J. Candès, B. Recht, "Exact Matrix Completion via Convex Optimization," *Foundations of Computational mathematics*, 2009, to Appear.

[2] B. Recht, M. Fazel, P. Parrilo, "Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization," *SIAM Rev.*, Preprint available at arXiv.

[3] A.L. Chistov, Yu. Grigoriev, "Complexity of quantifier elimination in the theory of algebraically closed fields," *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science*, Springer Verlag, 1984.

[4] M. Fazel, Matrix Rank Minimization with Applications, PhD thesis, Stanford University, 2002.

[5] D. Goldfarb, M. Shiqian, W. Zaiwen, "Solving low-rank matrix completion problems efficiently," *47th Allerton Conference on Communication, Control, and computing*, Illinois, 2009.

[6] R.H. Keshavan, A. Montanari, Oh. Sewoong, "Matrix Completion from a Few Entries," *IEEE Trans. On Information Theory*, Vol. 56, no. 1, pp. 2980-2998, May 2010.

[7] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed L0 norm," *IEEE Trans. on Signal Proc*, vol. 57, no. 1, pp. 289–301, January 2009.

[8] A. Blake, A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, MA, 1987.

[9] L. Kiryung, Y. Bresler, "ADMiRA: Atomic Decomposition for Minimum Rank Approximation," *IEEE Trans. On Information Theory*, Vol. 56, no. 1, pp. 4402-4416, September 2010.

[10] A.S. Lewis, "The convex analysis of unitarily invariant matrix norms," *Journal of Convex Analysis*, Vol. 2, pp. 173-183, 1995.