Efficient Algorithm for Low-rank Matrix Factorization with Missing Components and Performance Comparison of Latest Algorithms

Takayuki Okatani   Takahiro Yoshida   Koichiro Deguchi
Tohoku University, Japan
okatani@fractal.is.tohoku.ac.jp

Abstract

This paper examines numerical algorithms for factorization of a low-rank matrix with missing components. We first propose a new method that incorporates a damping factor into the Wiberg method to solve the problem. The new method is characterized by the way it constrains the ambiguity of the matrix factorization, which helps improve both the global convergence ability and the local convergence speed. We then present experimental comparisons with the latest methods used to solve the problem. No comprehensive comparison of the methods that have been proposed recently has yet been reported in literature. In our experiments, we prioritize the assessment of the global convergence performance of each method, that is, how often and how fast the method can reach the global optimum starting from random initial values. Our conclusion is that top performance is achieved by a group of methods based on Newton-family minimization with damping factor that reduce the problem by eliminating either of the two factored matrices. Our method, which belongs to this group, consistently shows a 100% global convergence rate for different types of affine structure from motion data with a very high population of missing components.

1. Introduction

The problem of factorizing a matrix $\mathtt{Y}$ with missing components into the product of two smaller matrices $\mathtt{U}$ and $\mathtt{V}$ as

$$\mathtt{Y} \to \mathtt{U}\mathtt{V}^\top, \tag{1}$$

is commonly encountered in several fields of science. In the field of computer vision, it is seen in SfM (structure from motion) and in the photometric analysis of multiple images that are taken under different illumination conditions etc. The problem can be stated as follows: Letting $\mathtt{Y}$ be an $m \times n$ matrix that is ideally (i.e., without any noise) of rank r, we wish to obtain two factors explaining $\mathtt{Y}$, an $m \times r$ matrix $\mathtt{U}$ and an $n \times r$ matrix $\mathtt{V}$,

A standard formulation of the problem is to determine $\mathtt{U}$ and $\mathtt{V}$ while minimizing the following L_2 norm error with respect to the existing components:

$$f(\mathtt{U}, \mathtt{V}) = \|\mathtt{W} \odot (\mathtt{Y} - \mathtt{U}\mathtt{V}^\top)\|_F^2, \tag{2}$$

where $\mathtt{W}$ is an indicator matrix of the same size as $\mathtt{Y}$ such that $w_{ij} = 1$ if the component y_{ij} of $\mathtt{Y}$ exists and $w_{ij} = 0$ otherwise ($i = 1, \ldots, m$, $j = 1, \ldots, n$); $\odot$ represents the component-wise product of the matrices. Although norms other than L_2 can also be used depending on the purpose, we consider only the L_2 norm here. (Please refer to [4] for a recently proposed method for the L_1 norm as well as a survey on related work.)

If $\mathtt{Y}$ does not have missing components, the minimization (2) can be easily solved. If $\mathtt{Y}$ has missing components, it requires iterative computation, for which global convergence is not generally guaranteed. In the field of computer vision, many methods (e.g., [1, 9, 5, 13, 11, 3, 15, 10]) have been proposed for this problem since Shum et al.[12] introduced the study of Wiberg [14]. These methods can be classified into two types; one involving global optimization ([1, 11, 3, 10, 15] and many earlier ones) and the other involving sequential repetition of local optimization (e.g., [6, 9, 5, 13]). In this paper, we consider only the former type approach.

This paper has two purposes. The first is to present a new numerical method for Eq.(2), incorporating a damping factor into the Wiberg method in an effective manner. The new method is characterized by its way of constraining the ambiguity of solutions. There is an infinite number of solutions to Eq.(2) reflecting the ambiguity of the factorization. That is, for any nonsingular 3×3 matrix $\mathtt{A}$, it holds that

$$\mathtt{U}\mathtt{V}^\top = \mathtt{U}\mathtt{A}^{-1}\mathtt{A}\mathtt{V}^\top = (\mathtt{U}\mathtt{A}^{-1})(\mathtt{A}\mathtt{V}^\top) = \mathtt{U}'\mathtt{V}'^\top. \tag{3}$$

As will be shown in the experimental results, convergence performance seems to greatly be affected by how to deal with this ambiguity. Our method constrains the ambiguity in such a way that the incorporated damping factor behaves ideally; that is, the method improves the global convergence ability without sacrificing the local convergence speed.

The second purpose is to experimentally compare the latest methods including our method for solving the problem. Recently, several methods have been proposed in the literature [11, 3, 10, 15]. Although each study includes the comparison of its proposed method against the existing methods, there are some problems with the comparisons; for example, not all existing methods are taken into account or the experimental conditions are not consistent with other studies. These problems hinder the accurate evaluation of the performances of these methods. In this study, we compare these methods through several experiments conducted under the same conditions.

In the experiments, we prioritize the investigation of the global convergence ability, which can be evaluated by considering how often the method reaches the global minimum starting from random initial values. The reason for this prioritization is our empirical finding that particularly for SfM data, some methods can achieve a success rate of nearly 100% even when a large portion of the data is missing.

In [1], Buchanan and Fitzgibbons evaluated a large number of algorithms that had been proposed till then and proposed (the use of) the damped Newton method. The conclusion of their study is that the damped Newton method is superior to other methods. Moreover, it is a benchmark method that has been cited in all subsequent studies. Thus, in this paper, we evaluate the performance of the six methods that have been proposed since then, namely, the damped Newton method [1], Chen's LM_S/LM_M [3], SALS of Zhao-Zhang [15], MFLRSDP of Mitra et al. [10], and our method, which we call the damped Wiberg method.

2. Revisiting the Wiberg method

The Wiberg method is a method that eliminates either $\mathtt{U}$ or $\mathtt{V}$ from the problem by using the (bi)linearity of $f(\mathtt{U},\mathtt{V})$ and applies the Gauss-Newton method to the reduced problem. When $\mathtt{U}$ is chosen for the elimination, we compute the minimizer $\mathtt{U} = \hat{\mathtt{U}}(\mathtt{V})$ of $f(\mathtt{U},\mathtt{V})$ for a fixed $\mathtt{V}$ and substitute it in $f(\mathtt{U},\mathtt{V})$, yielding

$$g(\mathtt{V}) \equiv \min_{\mathtt{U}} f(\mathtt{U},\mathtt{V}) = f(\hat{\mathtt{U}}(\mathtt{V}),\mathtt{V}). \tag{4}$$

Then, we minimize $g(\mathtt{V})$ with respect to $\mathtt{V}$.

We use the following notations in what follows. Let $\mathtt{U} = [\mathbf{u}_1,\dots,\mathbf{u}_m]^\top$ and $\mathtt{V} = [\mathbf{v}_1,\dots,\mathbf{v}_n]^\top$, where $\mathbf{u}_i$ and $\mathbf{v}_j$ are both r-vectors, and denote their vectorized versions as $\mathbf{u} = [\mathbf{u}_1^\top,\dots,\mathbf{u}_m^\top]^\top$ and $\mathbf{v} = [\mathbf{v}_1^\top,\dots,\mathbf{v}_n^\top]^\top$. Additionally, let $\mathbf{y}$ be the $p-$vector that contains the existing components of $\mathtt{Y}$ in a row-first order (p is the number of existing components). Thus, the equation $f(\mathtt{U},\mathtt{V}) = f(\mathbf{u},\mathbf{v})$ can be rewritten as

$$f(\mathbf{u},\mathbf{v}) = ||\mathbf{y} - \mathtt{F}\mathbf{u}||^2 = ||\mathbf{y} - \mathtt{G}\mathbf{v}||^2, \tag{5}$$

where $\mathtt{F}$ is a $p\times mr$ matrix containing only $\mathbf{v}_1,\dots,\mathbf{v}_n$ and $\mathtt{G}$ is a $p\times nr$ matrix containing only $\mathbf{u}_1,\dots,\mathbf{u}_m$; they have the following structure:

$$\mathtt{F} = \begin{bmatrix} \mathtt{F}_1 & & & \\ & \mathtt{F}_2 & & \\ & & \ddots & \\ & & & \mathtt{F}_m \end{bmatrix}, \quad \mathtt{G} = \begin{bmatrix} \mathtt{G}_1 \\ \mathtt{G}_2 \\ \vdots \\ \mathtt{G}_m \end{bmatrix}, \tag{6}$$

where the submatrices $\mathtt{F}_i$ and $\mathtt{G}_i$ ($i = 1,\dots,m$) are further defined as follows. We define an index set $S_i \equiv \{j \,|\, w_{ij} \neq 0\}$ and denote each of its elements as α_k ($k = 1,\dots,|S_i|$), such that $1 \leq \alpha_1 < \dots < \alpha_{|S_i|} \leq n$. $\mathtt{F}_i$ is a $|S_i| \times r$ dense matrix whose k-th row is given by $\mathbf{v}_{\alpha_k}^\top$; $\mathtt{G}_i$ is a sparse $|S_i| \times nr$ matrix whose k-th row vector stores only $\mathbf{u}_i^\top$ in indices $[(\alpha_k - 1)r + 1 : \alpha_k r]$.

Using these notations, the computation of the Gauss-Newton step minimizing $g(\mathbf{v})(= g(\mathtt{V}))$ is simplified to alternately perform the following two minimization problems:

$$\mathbf{u} \leftarrow \operatorname*{argmin}_{\mathbf{u}} f(\mathbf{u},\mathbf{v}), \tag{7a}$$

$$\mathbf{v} \leftarrow \mathbf{v} + \delta\mathbf{v}, \quad \text{where} \quad \delta\mathbf{v} = \operatorname*{argmin}_{\delta\mathbf{v}} ||\mathtt{Q}_\mathtt{F}\mathtt{G}\delta\mathbf{v} - \mathtt{Q}_\mathtt{F}\mathbf{y}||^2, \tag{7b}$$

where

$$\mathtt{Q}_\mathtt{F} = \mathtt{I} - \mathtt{F}(\mathtt{F}^\top\mathtt{F})^{-1}\mathtt{F}^\top. \tag{8}$$

Note that the factorization ambiguity of Eq.(3) appears as the fact that $\mathtt{Q}_\mathtt{F}\mathtt{G}$ in (7b) is rank-deficient; it is shown in [11] that its rank is deficient by r^2.

The elimination of the parameter (i.e., $\mathtt{U}$ or $\mathbf{u}$ in the above derivation) has two advantages, one theoretical and the other empirical. The theoretical one is that the problem size decreases and the computational efficiency increases. This is more pronounced when the height m and the width n of $\mathtt{Y}$ differ significantly; the elimination of the longer parameter decreases the problem size. The other empirical advantage is that the elimination of the parameter boosts global convergence ability, as will be shown in the results.

It should be noted that the Wiberg approach is very similar to a well known technique used in bundle adjustment of SfM for eliminating either the three-dimensional coordinates or the camera poses in the computation of the Newton step. The elimination is done by using the Schur complement of the Hessian matrix. In fact, Eqs.(7) can also be derived by computing the Schur complement for the Gauss-Newton approximation of the Hessian matrix, i.e., $\mathtt{A} \equiv \mathtt{J}^\top\mathtt{J}$, where $\mathtt{J} = [\mathtt{F},\mathtt{G}]$. The prominent deviation of our study from this technique of bundle adjustment is that by exploiting the linearity of the problem, we aim to improve the global convergence ability, which has thus far not been considered in the context of bundle adjustment.

In the next section, we consider incorporating a damping factor, which is similar to the one used in the damped Newton method [1] and Chen's LM_S/LM_M [3], to the Wiberg method. In [11], we pointed out that the standard Wiberg method (without a damping factor) is quite effective for problems in the field of computer vision, particularly as compared with the method of alternated least squares (ALS) and its variants, which were popular in those days. However, there remains room for improvement. Although the standard Wiberg method shows fairly good performances for easy data with low missing rate, its performance deteriorates quickly as missing rate increases and data become more difficult. Improving this convergence performance is the motivation of the next section.

3. Damped Wiberg method

3.1. Incorporating a damping factor

In each iteration of the Wiberg method given in Eqs.(7), Eq.(7b) is dominant in terms of computational cost. The matrix on the left hand side is $p \times nr$. In [3], it is argued that this matrix becomes too large for real problems, and thus, the Wiberg method is omitted in the comparative experiments. However, this argument is invalid, because converting (7b) to the normal equation reduces the problem to a size comparable to that in the methods proposed in [3].

By multiplying $(\mathtt{Q_F G})^\top$ with the terms inside the norm in Eq.(7b), we obtain the identical normal equation as

$$\mathtt{G}^\top \mathtt{Q_F G} \delta \mathbf{v} = \mathtt{G}^\top \mathtt{Q_F} \mathbf{y}, \tag{9}$$

where we use $\mathtt{Q_F} = \mathtt{Q_F}^\top$ and $\mathtt{Q_F}^2 = \mathtt{Q_F}$. Note that $\mathtt{Q_F G}$ is rank-deficit, as mentioned earlier, and so is $\mathtt{G}^\top \mathtt{Q_F G}$; thus, $\delta \mathbf{v}$ cannot be uniquely determined. The simplest solution is choosing $\delta \mathbf{v}$ minimizing $\|\delta \mathbf{v}\|^2$. Since we know that $\mathtt{G}^\top \mathtt{Q_F G}$ has the rank $nr - r^2$ [11], this $\delta \mathbf{v}$ can be computed in a numerically stable manner. The step $\delta \mathbf{v}$ thus determined gives a Gauss-Newton step.

Now, we consider the incorporation of a damping factor in Eq.(9) to improve the global convergence performance. A straightforward way of doing this is to add $\lambda \mathtt{I}$ in Eq.(9), yielding

$$(\mathtt{G}^\top \mathtt{Q_F G} + \lambda \mathtt{I}) \delta \mathbf{v} = \mathtt{G}^\top \mathtt{Q_F} \mathbf{y}. \tag{10}$$

We then control λ to ensure that the cost $g(\mathtt{V})$ decreases in each iteration, expecting that the method possesses both the good global convergence ability of gradient descent methods and the fast local convergence speed of the Newton method. If λ is controlled appropriately, the additive term $\lambda \mathtt{I}$ guarantees that the matrix on the left hand side is positive definite, and thus, $\delta \mathbf{v}$ is uniquely determined. However, adding the damping factor as in Eq.(10) is inadvisable because the matrix on the left hand side becomes nearly singular as $\lambda \to 0$. This can cause numerical instability when λ does approach zero or lower the local convergence speed if λ is controlled so as not to approach zero.

3.2. Constraining the ambiguity

To cope with this difficulty, we consider the fact that the null space of $\mathtt{G}^\top \mathtt{Q_F G}$ can be expressed in an explicit form.

Proposition 1. The null space of $\mathtt{G}^\top \mathtt{Q_F G}$ coincides with the column space of an $nr \times r^2$ matrix $\mathtt{N}$ defined as

$$\mathtt{N} = \left[\mathtt{N}_1^\top, \mathtt{N}_2^\top, \ldots, \mathtt{N}_n^\top \right]^\top, \tag{11}$$

where $\mathtt{N}_j$ is an $r \times r^2$ block diagonal matrix having r subblocks $\mathbf{v}_j^\top$:

$$\mathtt{N}_j = \begin{bmatrix} \mathbf{v}_j^\top & & \\ & \ddots & \\ & & \mathbf{v}_j^\top \end{bmatrix}, \quad j = 1, \ldots, n. \tag{12}$$

Proof. Similar to $\mathtt{N}_j$, let $\mathtt{M}_i$ be an $r \times r^2$ block diagonal matrix having r subblocks $\mathbf{u}_i$ and also let $\mathtt{M} = [\mathtt{M}_1^\top, \ldots, \mathtt{M}_m^\top]^\top$. It is easy to prove that $\mathtt{FM} = \mathtt{GN}$, and thus,

$$\mathtt{G Q_F G N} = \mathtt{G Q_F F M} = \mathtt{0}. \tag{13}$$

□

Although this proof is almost identical to the proof of the proposition presented in [11] to show that $\mathtt{Q_F G}$ has rank $nr - r^2$, we are concerned with the expression (11) itself.

Proposition 2. Among the solutions to Eq.(9), the one minimizing $\|\delta \mathbf{v}\|^2$ satisfies

$$\mathtt{N}^\top \delta \mathbf{v} = \mathbf{0}, \tag{14}$$

and vice versa.

Proof. We denote the orthogonal decomposition of a solution $\delta \mathbf{v}$ to Eq.(9) with respect to the column space of $\mathtt{N}$ as $\delta \mathbf{v} = \delta \mathbf{v}_\perp + \delta \mathbf{v}^*$, where $\delta \mathbf{v}_\perp$ is orthogonal to the column space. Since $\|\delta \mathbf{v}\|^2 = \|\delta \mathbf{v}_\perp\|^2 + \|\delta \mathbf{v}^*\|^2$, the solution that minimizes $\|\delta \mathbf{v}\|^2$ is given as $\delta \mathbf{v} = \delta \mathbf{v}_\perp$. This satisfies $\mathtt{N}^\top \delta \mathbf{v} = \mathtt{N}^\top \delta \mathbf{v}_\perp = \mathbf{0}$. The proof of the converse is omitted. □

The same solution minimizing $\|\delta \mathbf{v}\|^2$ is also obtained by solving the simultaneous equations (9) and (14). Using the fact that the vector on the left hand side of Eq.(9) is also orthogonal to the column space of $\mathtt{N}$, the two equations can be converted into a single equation:

$$(\mathtt{G}^\top \mathtt{Q_F G} + \mathtt{N N}^\top) \delta \mathbf{v} = \mathtt{G}^\top \mathtt{Q_F} \mathbf{y}. \tag{15}$$

It is evident that the matrix on the left hand side is of full rank. We incorporate the damping factor $\lambda \mathtt{I}$ here:

$$(\mathtt{G}^\top \mathtt{Q_F G} + \mathtt{N N}^\top + \lambda \mathtt{I}) \delta \mathbf{v} = \mathtt{G}^\top \mathtt{Q_F} \mathbf{y}. \tag{16}$$

Now, the matrix on the left hand side remains of full rank even when λ approaches 0. We use this equation to determine the update $\delta \mathbf{v}$.

Another approach involves making an arbitrary $nr \times (nr - r^2)$ matrix $\mathtt{N}_\perp$ whose column space is orthogonal to that of $\mathtt{N}$ and then solving the following equation for $\delta \mathbf{w}$:

$$\mathtt{G}^\top \mathtt{Q_F G N_\perp} \delta \mathbf{w} = \mathtt{G}^\top \mathtt{Q_F} \mathbf{y}. \tag{17}$$

Then, the update $\delta \mathbf{v}$ is obtained by $\delta \mathbf{v} = \mathtt{N}_\perp \delta \mathbf{w}$. However, this method requires the generation of $\mathtt{N}_\perp$ and its multiplication to the existing matrices. These operations incur overhead cost; therefore Eq.(16) is adopted.

The overall algorithm is summarized as Algorithm 1.

3.3. Other remarks on efficient computation

In each iteration, the minimization (7a) and the computation of the matrix on the left hand side of Eq.(17) require a certain amount of computational time. It is essential to use the sparseness of $\mathtt{F}$ and $\mathtt{G}$.

Algorithm 1 Damped Wiberg method with a constraint on the factorization ambiguity

```
1. Set some small value (e.g., 0.01) to $\lambda$.
2. Solve the minimization (7a) to determine $\mathbf{u}$.
3. Check for convergence. Exit if converged. Otherwise,
   go to Step 4.
4. Solve Eq.(16) for $\delta\mathbf{v}$ by performing the Cholesky de-
   composition followed by back substitution.
5. If the value of $f(\mathbf{u}, \mathbf{v} + \delta\mathbf{v})$ increases from the previous
   step, set $\lambda \leftarrow 10\lambda$ and go to Step 4. Otherwise, update
   $\mathbf{v} \leftarrow \mathbf{v} + \delta\mathbf{v}$ and $\lambda \leftarrow 0.1\lambda$ and go to Step 2.
```

Since $\mathtt{F}$ is a block diagonal matrix, its QR decomposition

$$\mathtt{F} = \mathtt{F}_Q\mathtt{F}_R \tag{18}$$

can be efficiently obtained by computing the QR decomposition of each subblock matrix; $\mathtt{F}_Q$ and $\mathtt{F}_R$ have the same block structure as $\mathtt{F}$. Once this decomposition is obtained, it can be used to convert the minimization (7a) into

$$\mathtt{F}_R\mathbf{u} = \mathtt{F}_Q^\top\mathbf{y}, \tag{19}$$

and then, $\mathbf{u}$ can be efficiently computed by back substitution. $\mathtt{F}_Q$ can also be used to efficiently compute the matrix $\mathtt{G}^\top\mathtt{Q}_\mathtt{F}\mathtt{G}$ in Eq.(16) as follows.

$$\begin{aligned}\mathtt{G}^\top\mathtt{Q}_\mathtt{F}\mathtt{G} &= \mathtt{G}^\top(\mathtt{I} - \mathtt{F}(\mathtt{F}^\top\mathtt{F})^{-1}\mathtt{F}^\top)\mathtt{G}\\ &= \mathtt{G}^\top\mathtt{G} - (\mathtt{F}_Q^\top\mathtt{G})^\top(\mathtt{F}_Q^\top\mathtt{G}).\end{aligned} \tag{20}$$

The sparseness of the matrices should be used in the multiplications.

The first matrix $\mathtt{G}^\top\mathtt{Q}_\mathtt{F}\mathtt{G}$ in Eq.(16) can be sparse or sometimes very sparse. The second matrix $\mathtt{N}\mathtt{N}^\top$ is also a sparse matrix with $(r-1)/r$ sparsity. However, the non-zero components are distributed over the entire matrix; thus, using a dense Cholesky decomposition to solve Eq.(16) is faster than using a sparse Cholesky decomposition in the experiments shown below. A more efficient solution of the equation is a topic to be considered in future research.

4. Experimental results

4.1. Compared methods

We first summarize the six methods compared in the experiments.

Damped Newton (DN) method proposed by Buchanan et al. This method [1] minimizes the following cost function that regularizes f with respect to $\mathtt{U}$ and $\mathtt{V}$.

$$f_\mathrm{R}(\mathtt{U},\mathtt{V}) = \|\mathtt{W}\odot(\mathtt{Y} - \mathtt{U}\mathtt{V}^\top)\|_F^2 + \mu_1\|\mathtt{U}\|_F^2 + \mu_2\|\mathtt{V}\|_F^2. \tag{21}$$

The regularization terms constrain the factorization ambiguity (3). The minimization is performed using the standard Newton algorithm that includes a damping factor. One drawback of the DN method is that the number of parameters is larger than the methods below; the Hessian matrix is $(m+n)\times(m+n)$. In DN, μ_1 and μ_2 are required to be specified. In our experiments, we used the Buchanan et al.'s MATLAB implementation, which is publicly available[1]. The implementation is partially optimized using a MEX file.

Chen's LM_S and LM_M LM_S and LM_M [3] are both damped Newton methods similar to DN, but they solve the reduced problem in a similar manner to the Wiberg method. Although the derivation is different, the parameter elimination step is expected to be essentially the same as the Wiberg method, since there is no (effective) step other than Eq.(4) to do so. LM_S does not constrain the factorization ambiguity at all, whereas LM_M constrains it by confining the search space to the Grassmann manifold, i.e., $\{\mathtt{V}\,|\,\mathtt{V}^\top\mathtt{V} = \mathtt{I}\}$ [8]. Because of the lack of a constraint, LM_S has the same problem as the updating scheme of Eq.(10); the linear equation for the Newton step becomes indeterminate if λ approaches 0. The prominent differences between LM_S/LM_M and our method are as follows: LM_S and LM_M compute the full Hessian matrix whereas our method computes its Gauss-Newton approximation, and LM_M and our method constrain the factorization ambiguity in a different manner. We used the Chen's MATLAB implementation, which is publicly available[2].

Damped Wiberg (DW) method We refer to the algorithm presented in the previous section the damped Wiberg (DW) method. We implement it in the native MATLAB code (i.e., no optimized MEX file) and use it in the experiments[3].

Matrix factorization using low-rank semidefinite program (MFLRSDP) based method Recently, Mitra et al. showed [10] that the matrix factorization problem can be formulated as a low-rank semidefinite program (LRSDP) [2]. The problem is then solved using the augmented Lagrangian method, a standard method for LRSDP, which uses the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, one of the quasi Newton methods, for the computation. This matrix factorization method, called the MFLRSDP, minimizes the regularized cost function $f_\mathrm{R}(\mathtt{U},\mathtt{V})$ of Eq.(21), and therefore, it requires the specification of μ_1 and μ_2. Note that in [10], MFLRSDP is not compared with any Newton-family methods other than DN. In the experiments below, we used the Mitra et al.'s MATLAB implementation, which is publicly available[4].

Successively alternate least square (SALS) In [15], Zhao and Zhang presented a method that assumes a "weak"

[1]http://www.robots.ox.ac.uk/ amb/
[2]http://sist.sysu.edu.cn/ chenpei/
[3]Available from http://www.fractal.is.tohoku.ac.jp/okatani/DW.html.
[4]http://ttic.uchicago.edu/ ssameer/#code

prior knowledge that all components of $\mathtt{Y}$ including the missing ones lie within a certain range, i.e., $a \leq y_{ij} \leq b$. The method, called SALS, iterates the following procedures: for a given reconstruction $\mathtt{X}$ of $\mathtt{Y}$, $\mathtt{U}$ and $\mathtt{V}$ are optimized by alternated least squares (ALS) (e.g., [1]), so that $\mathtt{UV}^\top$ will approach $\mathtt{X}$, followed by the reconstruction of $\mathtt{Y}$ by computing $\mathtt{X} = \mathtt{UV}^\top$ and imposing the inequality constraint to it.

Although there can be a substantial amount of prior knowledge that is potentially effective for the matrix factorization, we avoid considering it in this paper because it is problem-specific and, moreover, optimization with many constraints often shows only a limited global convergence. Nevertheless, we include SALS for performance comparison, since the inequality constraint may be generally available owing to its weakness; the results shown in [15] are promising. Besides $[a, b]$, SALS further requires the specification of several control parameters, K, λ, and δ: Refer to [15] for details.

It should be noted that in [15], SALS has been compared to LM_M and LM_S, but there is no experimental result using real data that shows its global convergence ability starting from random initial values. We used our MATLAB implementation of SALS in the experiments.

4.2. Results for real data

We first conducted experiments employing the real data used in [1]; these data can be downloaded from http://www.robots.ox.ac.uk/ amb/. Their specifications are shown in Tbl.1.

Table 1. The details of three real data.

	dinosaur	*giraffe*	*face*
Matrix size	319×72	240×166	2596×20
Rank	4	6	4
Missing rate	76.9%	30.2%	35.1%

We run each method for these data starting from random initial values. The initial value of $\mathtt{V}$ is generated according to a normal distribution $N(0, 1)$; that is, $\mathtt{V}$ is initialized by a MATLAB code `randn(n,r)`. The least squares solution to (7a) for the initialized $\mathtt{V}$ is used in the method requiring the initial value of $\mathtt{U}$. For each run, the identical initial value of $\mathtt{V}$ thus generated is supplied to all the methods. We also use an identical termination condition if possible. For methods other than MFLRSDP, we terminate the computation either when $|f_{k-1} - f_k| < 10^{-9} f_k$ is satisfied (f_k is the value of f at the k-th iteration), or when the iteration count exceeds a predetermined number. For MFLRSDP, we follow [10] and set the upper limit of execution time at ten minutes for *dinosaur* and *face* and at one hour for *giraffe*. A few methods require the specifications of other control parameters, which were obtained considering the recommendations in the literature. For DN and MFLRSDP, we set the regularization parameters $\mu_1 = \mu_2 = 10^{-3}$. For SALS, we set the range $[a, b]$ to coincide with the minimum and maximum values of the existing components; we set $K = 60$, $\lambda = 0.01$, and $\delta = (b - a)/100$.

Figure 1 shows the results. We performed 100 runs for both *dinosaur* and *face* and 25 runs for giraffe. DN was not performed for *face*, since it was very slow. For each run, we recorded the residual error, iteration counts, and elapsed CPU time measured by MATLAB. The residual error is defined as

$$\mathrm{RMS} = \sqrt{\|\mathtt{W} \odot (\mathtt{Y} - \mathtt{UV}^\top)\|_F^2 / p}, \tag{22}$$

where p is the number of existing components. All the experiments were conducted on a PC with a 3.16 GHz Xeon CPU, 32 GB memory, and 64-bit Windows OS running 64-bit MATLAB 7.9.0.529/R2009b.

The histograms in the left column of Fig.1 show the RMS errors. From these, we can evaluate the global convergence ability of each method starting from random initial values, i.e., the odds of finding the global minimum. It is observed that DW is the best; its convergence to the global minimum is nearly 100% in the case of *dinosaur* and *giraffe*. The second and third best are LM_M and LM_S, respectively; they are inferior to DW in the case of *dinosaur* but are comparable to DW in the case of the other two data. DN shows a moderate performance in the case of *giraffe* but only shows a poor performance in the case of *dinosaur*. MFLRSDP is better than DN in the case of *dinosaur*, as reported in [10], but its overall performance is considerably inferior to the top three methods. SALS shows similarly inferior performances but shows a moderate performance particularly in the case of *face*. This may be because the prior knowledge assumed in SALS fits the nature of the problem.

The histograms in the middle column of Fig.1 show the number of iterations carried out by each algorithm until convergence. From these, we can compare the convergence speed of the Newton-family methods (i.e., DN, LM_M, LM_S, and DW). MFLRSDP and SALS have iterative structures that are different from these methods, and thus, a direct comparison is not suitable. (Hence, MFLRSDP is omitted in the figure.) It is observed that DW shows superior performance again; it significantly outperforms other methods in the case of *dinosaur* and is the best in the case of *giraffe* and nearly the best for *face*. The second and the third best are, again, LM_M and LM_S, respectively. It is noteworthy that LM_S sometimes took a larger number of iterations than LM_M; this symptom may be attributable to the lack of a constraint on the factorization ambiguity in LM_S.

The histograms in the right column of Fig.1 show the elapsed time taken by each method. Although elapsed time might not be a good measure to estimate the theoretical efficiency of algorithms, it can still provide rough estimates. It is seen that DW significantly outperforms other methods; on average, it is 2 to 100 times faster than LM_M. SALS and MFLRSDP occasionally terminate in a shorter time than LM_M; however, these instances are ignored because the converged solutions are not the global minimum.

4.3. Results for synthetic data

We also conducted experiments using synthetic data to examine how the performances of the methods are affected

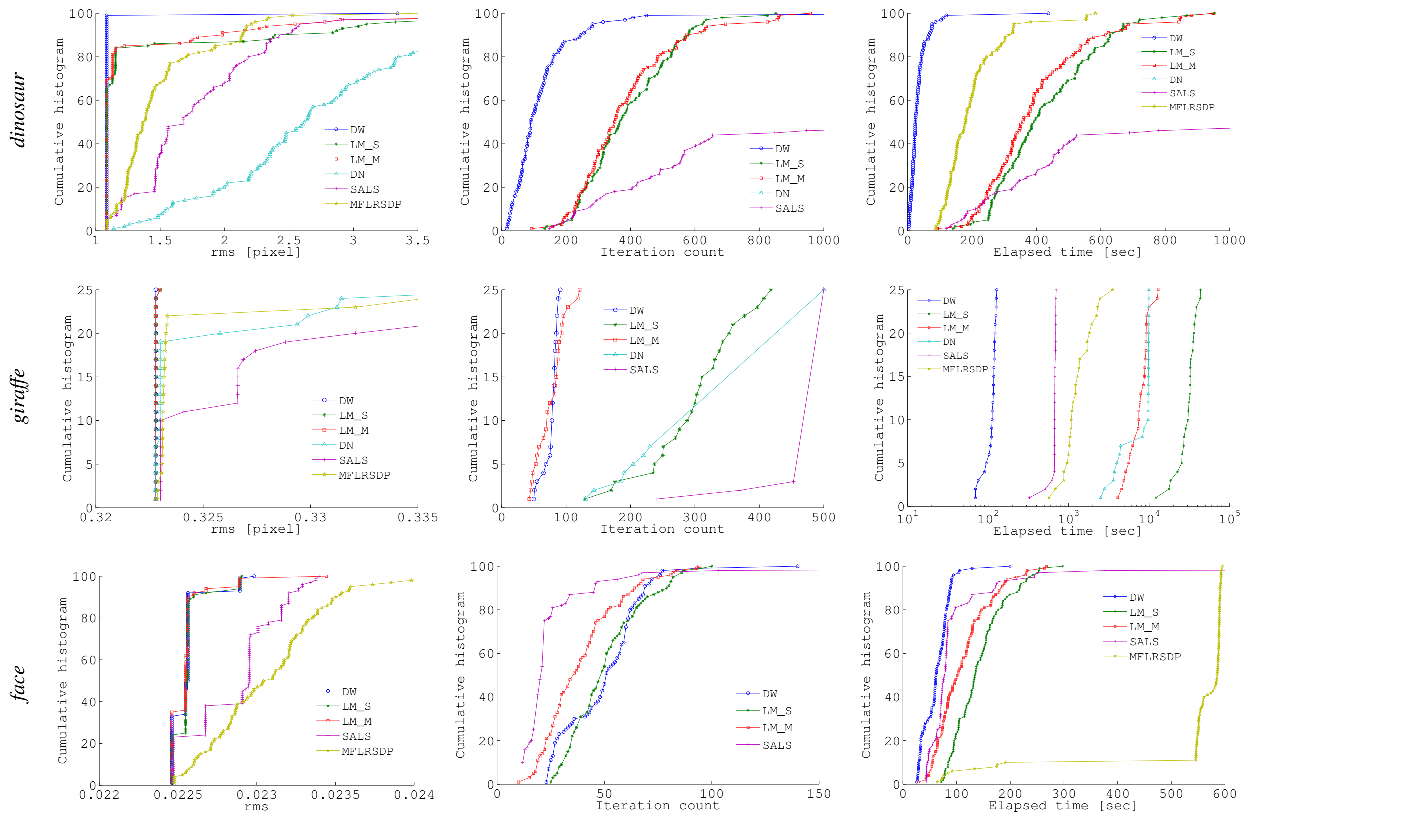

Figure 1. Results for real data. From upper to lower row: *dinosaur*, *giraffe*, and *face*. From left to right: cumulative histograms of residual errors, iteration counts used until convergence, and elapsed CPU time. Note that the figures are in color.

by the nature of the data, such as the population of missing components.

To generate data, we assume two scenarios of SfM. In the first scenario, an orthographic camera moves around M points in space and captures n images so that a sequence similar to *dinosaur* is obtained. Henceforth, we refer to the data thus generated as *rotation*. In the second scenario, the same camera undergoes a translational move along the y axis, in front of M points, and captures n images. Henceforth, we refer to this data as *translation*. The points are randomly generated as per uniform distributions with the xyz range $[-100, 100] \times [-100, 100] \times [0, 200]$ for *rotation* and $[-100, 100] \times [0, 720] \times [0, 100]$ for *translation*. Setting the image size to 300×300, we add Gaussian random noise with standard deviation $\sigma \in [0.5 : 3.0]$ to the image points. For both data, $\mathtt{Y}$ has the size $2M \times n$.

We also generate the indicator $\mathtt{W}$ of the existing components for $\mathtt{Y}$ thus obtained. We consider two types of patterns for $\mathtt{W}$. One is a *random pattern* which is randomly generated while fixing the number of 1's in each row of $\mathtt{W}$. The other is a *band-diagonal pattern*, which can occur in real SfM problems, as in *dinosaur*. We denote the number of existing components per row by ω in the following discussions. For each pattern, the effects of ω on the convergence were examined by varying ω.

We first set the matrix size as 200×60. Fig.2 shows the results of the rotation/band-diagonal sequence over 20 runs. The results for other sequences are omitted here, since they are similar. The upper and lower rows show the results when $\sigma = 0.5$ and 3.0, respectively. results display the same tendency as the results obtained for real data. DW, LM_M, and LM_S perform significantly better than the others, and DW performs the best among the three. This trend is almost the same for different noise strengths, although it can be seen that the superiority of DW to LM_M/LM_S slightly increases with the increase in noise. It is also noted that DW consistently achieves 100% global convergence rate for from around $\omega = 10$ (83.3% missing); however, it takes less computational time as compared to other methods.

The performance of these methods for matrices of large size was examined by experiments using 500×500 matrices. In [10], it is reported that DN is very slow for matrices of this size; therefore, MFLRSDP is compared against methods other than DN, such as Optspace [7], with the conclusion that MFLRSDP outperforms them. In fact, our experiments confirm that DN, LM_S, and LM_M require a significant amount of computational time. Because the speed of DW is comparable to MFLRSDP, we compared DW and MFLRSDP using the 500×500 matrices synthesized as above. We choose $\sigma = 0.5$. For MFLRSDP, we

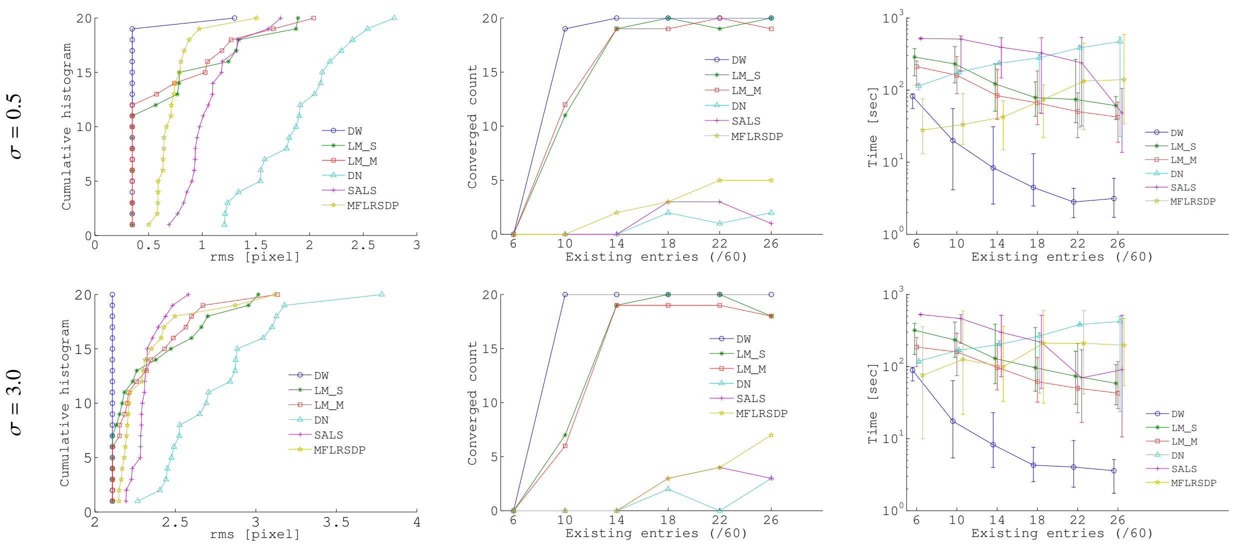

Figure 2. Results for synthetic data (matrix size 200×60). Upper row: $\sigma = 0.5$. Lower row: $\sigma = 3.0$. From left to right: the cumulative histograms of the RMS residual error for $\omega = 10$ out of 60 (i.e., 83.3% missing), the number of runs for which each method converges to the global minimum versus the number of the existing components per row, ω, and the elapsed time versus ω (the bars indicate the average, maximum, and minimum). Note that the figures are in color.

set $\mu_1 = \mu_2 = 10^{-3}$ and the upper limit of execution time to 1500 seconds for the random patterns and 3500 seconds for the band-diagonal patterns. By varying the number of existing components per row, ω, from 10 to 80 out of 500, we run both methods starting from random initial values as in the previous experiments.

Fig.3 shows the results. The results in the upper row (the rotation/random-pattern sequence) suggest that DW and MFLRSDP achieve similar performances, although DW is slightly better. The results in the middle row (rotation/band-diagonal) and lower row (translation/band-diagonal) show that DW continues to perform well although some performance deterioration can be observed, whereas MFLRSDP shows very poor performance. The drastic performance difference seen in case of MFLRSDP is attributable to the pattern of existing/missing components in Y; MFLRSDP performs well in the case of random patterns but performs poorly in the case of the band-diagonal patterns. Note that these results agree with the results shown in [10], in which only randomly generated patterns are used for W. It is seen that the performance of DW also deteriorates for the band-diagonal patterns; the global convergence rate becomes 100% for around $\omega = 22$ (95.6% missing) in the case of the random patterns and for around $\omega = 50$ (90.0% missing) in the case of the band-diagonal patterns, respectively. These observations imply that the band-diagonal patterns are more difficult than the random patterns.

5. Summary and discussion

We have studied the problem of factorizing a low-rank matrix with missing components into the product of two smaller matrices. We present a numerical method called the damped Wiberg (DW) method for this problem; this incorporates a damping factor into the Wiberg method. We then show the experimental comparisons of the latest six methods, including our method, by using synthetic as well as real data.

In all the experiments, the best performance is achieved by the group of Newton-family methods that have a damping factor and which reduce the problem by eliminating either of the two matrices, i.e., DW and Chen's LM_M and LM_S. On the other hand, the method retaining both matrices as parameters, i.e., the damped Newton (DN) method, shows a significantly inferior performance as compared to these methods. Besides the fact that DN does not eliminate parameters, it also differs from the three methods in that it uses regularization. However, the regularization is not considered to be the main reason for the limited performance of DN, since we experimentally confirmed that if we invalidate the regularization terms by setting $\mu_1 = \mu_2 = 0$, the results did not improve. In this case, the factorization ambiguity is left unconstrained, but the same applies to LM_S, which nevertheless shows fairly good performance. Considering that DW, LM_M, and LM_S have several differences (e.g., different constraints on the factorization ambiguity and the computation of the full Hessian or its Gauss-Newton approximation), it is natural to think that the significant difference in performance between these methods and DN is attributable to whether or not the parameter is eliminated. Thus, we conclude that the parameter elimination boosts global convergence ability.

Among the three methods, DW is rated the best in terms of both global convergence ability and convergence speed. It is particularly observed for SfM problems that DW con-

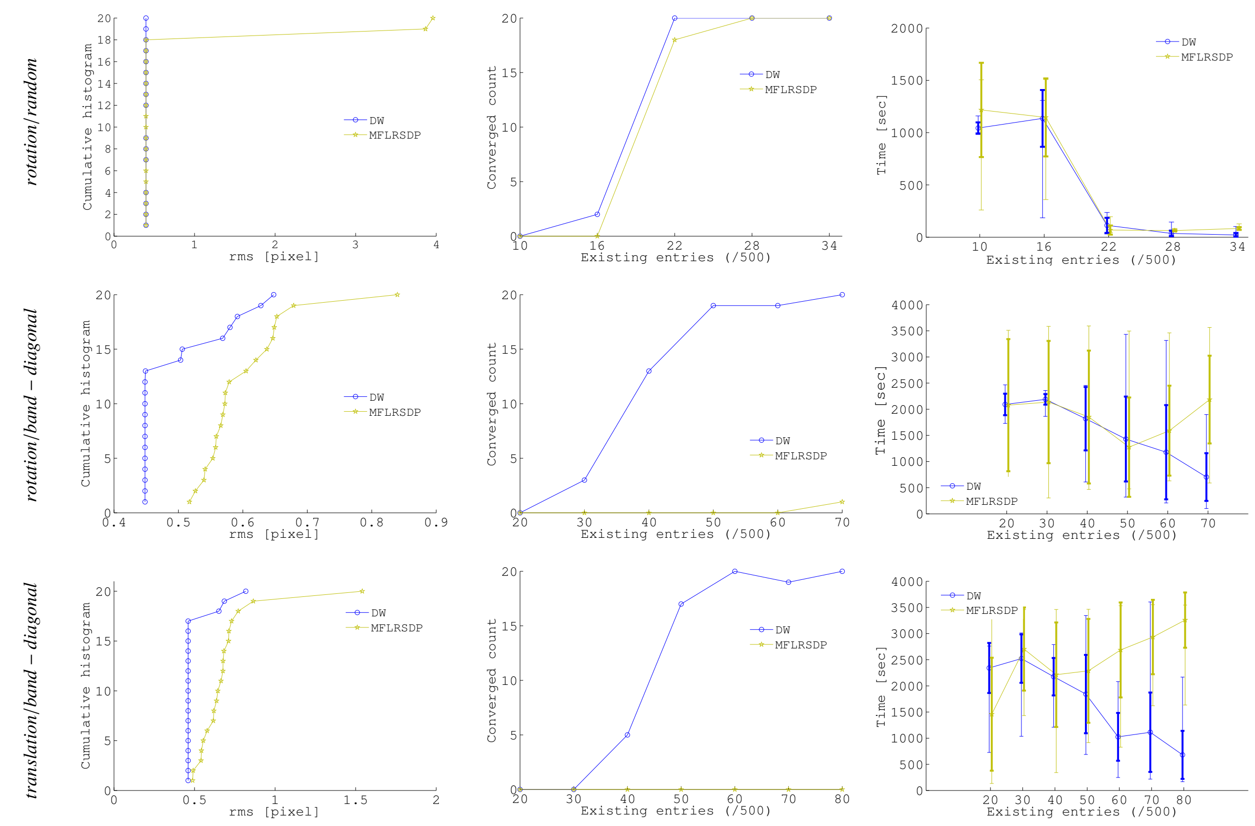

Figure 3. Results for synthetic data (matrix size 500×500). From upper to lower row: the results for rotation/random-pattern, rotation/band-diagonal, and translation/band-diagonal sequences. From left to right column: cumulative histograms of the RMS residual error for a selected ω (from upper to lower row, ω = 22, 40, 50), the number of runs for which the algorithm converges to the global minimum versus ω, and the elapsed time versus ω (the bars indicate the average, maximum, and minimum). Note that the figures are in color.

verges to the global minimum even if the population of missing components is very large; DW achieves a success rate of 100% if the population of the missing components is smaller than a certain number. Although more experiments are required, we can say that the SfM problems are such that the global minimum can be found with high probability, independent of initial values. They can be regarded as bundle adjustment assuming an affine camera. Although the linearity of the problem undoubtedly contributes to it, this empirical finding is contrasted with the ordinary bundle adjustment for perspective cameras, in which such global convergence, or equivalently, the independence of initial values, has never been an issue.

References

[1] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Proc. CVPR*, 2005. 1, 2, 4, 5

[2] S. Burer and C. Choi. Computational enhancements in low-rank semidefinite programming. *Optimization Methods and Software*, 21(3):493–512, 2006. 4

[3] P. Chen. Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix. *Int. J. Comput. Vision*, 80:125–142, October 2008. 1, 2, 3, 4

[4] A. Eriksson and A. V. D. Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l1 norm. In *Proc. CVPR*, 2010. 1

[5] N. Guilbert, A. Bartoli, and A. Heyden. Affine approximation for direct batch recovery of euclidian structure and motion from sparse data. *Int. J. Comput. Vision*, 69:317–333, September 2006. 1

[6] D. W. Jacobs. Linear fitting with missing data for structure-from-motion. *Comput. Vis. Image Underst.*, 82(1):57–81, 2001. 1

[7] R. H. Keshavan, A. Montanari, and S. Oh. Matrix Completion From a Few Entries. *IEEE Trans. on Information Theory*, 56(6):2980–2998, June 2010. 6

[8] J. H. Manton, R. Mahony, and Y. Hua. The geometry of weighted low-rank approximations. *IEEE Trans. on Signal Processing*, 51(2):500–514, 2003. 4

[9] D. Martinec and D. Martinec. 3d reconstruction by fitting low-rank matrices with missing data. In *Proc. CVPR*, pages 198–205. IEEE Computer Society, 2005. 1

[10] K. Mitra, S. Sheorey, and R. Chellappa. Large-scale matrix factorization with missing data under additional constraints. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2010. 1, 2, 4, 5, 6, 7

[11] T. Okatani and K. Deguchi. On the wiberg algorithm for matrix factorization in the presence of missing components. *Int. J. Comput. Vision*, 72:329–337, May 2007. 1, 2, 3

[12] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(9):855–867, 1995. 1

[13] J.-P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy. Algorithms for batch matrix factorization with application to structure-from-motion. In *Proc. CVPR*, pages 1–8, 2007. 1

[14] T. Wiberg. Computation of principal components when data are missing. In *Proceedings Symposium of Comp. Stat.*, pages 229–326, 1976. 1

[15] K. Zhao and Z. Zhang. Successively alternate least square for low-rank matrix factorization with bounded missing data. *Comput. Vis. Image Underst.*, 114:1084–1096, October 2010. 1, 2, 4, 5