Recovering the Missing Components in a Large Noisy Low-Rank Matrix: Application to SFM

Pei Chen and David Suter, *Senior Member*, *IEEE*

Abstract—In computer vision, it is common to require operations on matrices with "missing data," for example, because of occlusion or tracking failures in the Structure from Motion (SFM) problem. Such a problem can be tackled, allowing the recovery of the missing values, if the matrix should be of low rank (when noise free). The filling in of missing values is known as imputation. Imputation can also be applied in the various subspace techniques for face and shape classification, online "recommender" systems, and a wide variety of other applications. However, iterative imputation can lead to the "recovery" of data that is seriously in error. In this paper, we provide a method to recover the most *reliable* imputation, in terms of deciding when the inclusion of extra rows or columns, containing significant numbers of missing entries, is likely to lead to poor recovery of the missing parts. Although the proposed approach can be equally applied to a wide range of imputation methods, this paper addresses only the SFM problem. The performance of the proposed method is compared with Jacobs' and Shum's methods for SFM.

Index Terms—Imputation, missing-data problem, rank constraint, singular value decomposition, denoising capacity, structure from motion, affine SFM, linear subspace.

1 Introduction

Several problems in computer vision (and beyond) can be reduced to fitting a large matrix to its closest low-rank approximation: The factorization method under affine models of Structure from Motion (SFM) [16], [17], [20], [26], optical flow estimation in multiframe video [12], [13], subspace constraints in face recognition and indexing, pose determination, data mining, and a plethora of related problems (e.g., customer modeling and recommender systems [3], [22]).

In this paper, we restrict our application to the structure from motion in an affine camera setting, although this is to make the problem concrete rather than to exploit any special structure of that problem. Indeed, we do *not* use any features of the problem formulation that is specific to the particular application (see Section 1.1), so we will generically say that the matrix $\mathbf{M}$ (of dimension $m \times n$ and with real number entries) should be (without noise) of rank $r << \min\{m, n\}$. A consequence of the matrix being of rank r is that it can be factored into $\mathbf{RS}$ for real rank-r matrices $\mathbf{R}$ of size $m \times r$ and $\mathbf{S}$ of size $r \times n$, and vice versa. For the SFM problem, we are, of course, interested in *particular* factors (the factorization is not unique because for any invertible matrix $\mathbf{G}$ of size $r \times r$ we have $\mathbf{RS} = (\mathbf{RG})(\mathbf{G}^{-1}\mathbf{S})$). However, for other problems, we are not interested in any of the factors per se, but we are interested in the projection onto a low rank matrix to reduce noise, to fill in missing data, or extrapolate to as yet uncollected data. For example, we may wish to exploit the low rank constraint to assist in the feature point-matching problem (predicted search ranges) or to extrapolate tracks.

• *The authors are with the Department of Electrical and Computer Science, Egineering Monash University, PO Box 35, Clayton Victoria 3800, Australia. E-mail: {pei.chen, d.suter}@eng.monash.edu.au.*

Manuscript received 7 Apr. 2003; revised 9 Oct. 2003; accepted 9 Jan. 2004. Recommended for acceptance by Z. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0019-0403.

In most real-world problems, noise is inevitably introduced in the data. In the presence of noise, the measurement matrix quickly becomes full-rank. Thus, the matrix has to be projected upon its low-rank approximation $\mathbf{M}^r$ minimizing mean squared error (using Frobenius norm):

$$\|\mathbf{M} - \mathbf{M}^r\|_F^2. \tag{1}$$

The singular value decomposition (SVD) gives the best solution to this problem [8]: $\mathbf{M} = \mathbf{UDV}^T$, $\mathbf{M}^r = \mathbf{UD}^r\mathbf{V}^T$, where $\mathbf{D}^r$ is obtained by setting to 0 all of the singular values except the r largest ones. This is classical and is the starting point of the original factorization method for SFM and, hence, for many of its variants.

We could equivalently seek the rank-r factors explicitly in the formulation. That is, finding $\mathbf{R}$ of size $m \times r$ and $\mathbf{S}$ of size $r \times n$, that minimize

$$\|\mathbf{M} - \mathbf{RS}\|_F^2. \tag{2}$$

In such cases, one can side-step directly computing the "clean" $\mathbf{M}^r$ (the reprojected points in SFM terminology).

The issues to solve, other than computational efficiency issues, include: how to deal with missing values and how to deal with large amounts of data or data that is arriving sequentially. We will focus here on the first problem and an algorithm is presented in Section 4.3.

1.1 Missing-Data Problem in SFM

In SFM, one starts from the mathematical relationship between the measurement matrix $\mathbf{M}$ (coordinates of features tracked through frames), the object-camera motion matrix $\mathbf{R}$, and the structure/shape matrix $\mathbf{S}$. In the nondegenerate cases, and assuming an affine camera, the measurement matrix, should be exactly of rank 4. However, one can exploit the special structure: the "registered" measurement matrix, formed by subtracting the center of mass of the image points from their coordinates, which should be of rank 3 [16], [17],

[20], [26], and one can even reduce the problem to a rank-1 problem [1].

Regardless of what formulation, in terms of rank, the SVD cannot be directly used if some of the data are unavailable. This issue has been regarded [14], [15], [21] as the major drawback of the factorization method. Attempts to apply a subspace projection approach, in the presence of missing data, can be divided into two categories:

1. Those that attempt to "fill in" (or *impute*) the missing values:
   a. The seminal approach of Tomasi and Kanade [26] where the filling in is called "*hallucination*." In their somewhat heuristic approach to the missing data subproblem, a full submatrix (no missing entries) is first decomposed by the factorization method and then the initial solution grows by one row or by one column at a time, hallucinating missing data. The final estimate is then refined by employing a steepest descent minimization method on a least squares fitting criterion (2) $\|\mathbf{M} - \mathbf{RS} - \mathbf{C}\|_F^2$, where the inclusion of $\mathbf{C}$ makes the adjustment for the registration.
   b. Jacobs' method [14], [15] treated each column, with some missing entries, as an affine subspace and solved the problem by obtaining the intersection of all the quadruple (in practice, a large selection of) affine subspaces. Unknown entries are recovered by finding, for each column, the least squares regression onto this subspace.
2. Methods that directly obtain the factors—thus not imputing the measurement matrix (directly), e.g., Shum's method [23] and Guerreiro and Aguiar's work [9]. Though Shum's method was not originally formulated for SFM (see Section 1.2), Jacobs [14], [15] suggested that it could be applied to the SFM problem. We note that Shum's formulation uses data weighting to incorporate confidence measures, an elaboration not essential to our exposition. In essence, the method iteratively solves coupled least squares problems for the factors starting from the formulation of (2), but modifying the Frobenius norm so that only entries for measured data are involved, and adding the weights as mentioned previously. Since the formulation is bilinear in the factors, one can hold one factor constant and solve a linear least square problem for the other factor. Thus, the missing data are only indirectly imputed (one can "reproject" the recovered structure onto the images).

Tomasi and Kanade's approach to the problem of occlusion [26] has the following disadvantages: needing to start from a complete submatrix (it is an NP-hard problem of finding the largest complete submatrix), asymmetric usage of the data, and error propagation, as pointed out by Jacobs [14], [15].

The greatest advantage of Jacobs' method lies in the fact that it does not need to start from a complete submatrix. Ideally, for a generic problem, all the quadruple affine subspaces should be utilized in order to obtain a good result. In practice, a selection of the affine subspaces is needed. However, in the severe noise case, using only a small portion of the affine subspaces may produce unsatisfactory results. Intrinsically, Jacobs' linear approach can be employed in any missing-data problems under low-rank constraint; however, better performance for the SFM problem can be obtained because some "outlier" detection strategies are used, by incorporating the specialty of the SMF problem; while, for a general low-rank problem, the performance of the generic algorithm proved to be far away from the optimal solution, especially when there is a lot of missing data.

One drawback of Shum's approach is its dependence on an initial matrix, although a random initial matrix works when the percentage of the missing data is low and the data is not highly corrupted by noise. Even taking Jacobs' result as its initial point, Shum's approach still tends to diverge when there is a lot of missing data, especially for the generic low-rank problems.

Recently, by combining Jacobs' method [14], [15] with the projective factorization method of Sturm and Triggs [24], Martinec and Pajdla [18] solved the missing-data problem under the perspective model. Various geometric constraints [4], [11], [16] have also been employed to cope with the missing-data problem. For example, Heyden and Kahl [11], [16] proposed to use "closure constraints" for affine construction, where the missing-data problem can be naturally handled. They noted that Jacobs' method could be regarded to be "dual" to the closure constraints. It should also be noted that the missing-data problem in SFM could be efficiently solved by an incremental SVD [2]. Our own method for solving this problem is to be found in Section 4.3.

1.2 Other Missing Data Problems under Low Rank Constraint

Low rank-based imputation is so commonly useful that it is not surprising that many variations have appeared in the literature. Many applications are quite far removed from SFM: e.g., DNA prediction [27], or in a recommender system [3], [22]. Yet, these studies share the same intrinsic nature: missing-data problem under low-rank constraints.

The approach used in DNA prediction [27] employs an "SVDimpute" algorithm that bears a superficial similarity to our approach. The starting point of that approach is to fill in the missing values with row averages, then to use the SVD to rank r-project, then regress the missing values against the spanning vectors of the SVD, the process then being reiterated until convergence. The first potential drawback of these imputation methods is that the initial values for the starting point are rather arbitrary. Such limits its application to the cases where only a few components are missing [3], [22]. Second [27], only one missing component is updated at a time—an inefficiency. More importantly, as will be covered in the Appendix, such a strategy does not impute with minimal distance to the "current" subspace. Thus, convergence cannot be ensured. Indeed, the same criticisms as have been leveled at Tomasi and Kanade apply: strong dependence on the starting matrix and the imputation order [2], [3]. In addition, the iterative imputation method has the possibility of exhibiting "bad behavior" (see the Appendix), i.e., the estimate goes further from the underlying optimal solution as the iteration proceeds. However, such an important issue was overlooked in [27].

In a recommender system, the low rank constraint is supposed to capture customer preferences and it needs to be continually updated. However, it would be very computationally expensive to update the system online by a traditional

SVD. Brand [2], [3] proposed an incremental SVD to efficiently do this work, making the online updating possible. In what Brand calls bootstrapping [3], he reorders the matrix to have a dense submatrix in the top left corner and incrementally adds rows and columns using incremental SVD updating routines. Incremental update is also desirable in SFM problems [2], but it is beyond the scope of the present paper.

1.3 Contributions of this Paper

The main contribution of this paper is that we provide a means of determining which parts of the matrix should be used in the iterative imputation/recovery process. In the SFM context, this corresponds to deciding which tracks and/or which frames (typically the former) should be exploited in the iterative recovery process. Intuitively, the gain, on the one hand, of using more data (rows and/or columns) is balanced by the fact that extra rows and cols carry more missing entries. Rows or columns that have almost all entries missing are not likely to bring much extra information and the extra degrees of freedom can make the recovery less stable. Incorporation of data with more missing values can cause the solution to "wander" away from the true solution.

As a second contribution, we present an iterative imputation strategy and prove its weak convergence. Although falling short of a theoretical guarantee, the weak convergence, together with our mechanism of precluding the "wandering" of the iterative approach, ensures the iteration to the optimal solution in almost every case. This will be demonstrated by experiments.

1.4 Overview of the Paper

In Section 2, we first state the general missing-data problem under the low-rank constraint, using an objective function that is subtly different from the one in Shum's method. In Section 3, we analyze the central idea, used in the imputation approach [3], [22], [27], i.e., to fill in the missing data so that the complete vector has a minimal distance to a known low-rank subspace. Then, we propose a new iterative method of recovering the missing data in a large low-rank matrix and prove its weak convergence. In Section 4, we propose a criterion determining whether it is worth incorporating the incomplete vectors in the iteration. In Section 5, we experimentally compare the algorithm with Jacobs' and Shum's methods. In the Appendix, we discuss some aspects of the iterative method, including its convergence, the "wandering" issue, a bootstrapping strategy that provides a partial solution to the "wandering issue" (hinting at a more complete solution), and the relation to other approaches.

2 The Definition of the Problem and Its Nonlinear Nature

2.1 The Problem

A large matrix $\mathbf{M} \in R^{m,n}$, which should have a low rank r, is corrupted with noise (assumed to be i.i.d. Gaussian), and has missing entries. The problem is to recover these missing entries and to minimize the approximation error between the recovered matrix, $\hat{\mathbf{M}}$ and its closest rank-r approximation, $\hat{\mathbf{M}}^r$:

$$\min \quad ||\hat{\mathbf{M}} - \hat{\mathbf{M}}^r||_F^2 \tag{3}$$

subject to $\hat{M}_{i,j} = M_{i,j}$ if $M_{i,j}$ is observed. In other words, we seek to minimize the difference between the imputed matrix $\hat{\mathbf{M}}$ (where the missing values have been recovered but the matrix *has not been denoised*) and the closest rank-r approximation of the imputed matrix $\hat{\mathbf{M}}^r$ (now imputed and denoised).

Note: The minimization objective is different from that in Shum's approach [23], where the objective is to *recover the matrix factors* that minimize *the reprojection error of the "nonmissing" data*, i.e., the sum of the square of the difference between known elements in the incomplete matrix and the corresponding elements in the new recovered matrix, which is exactly of low-rank. Moreover, Shum's formulation incorporates *weighted* errors—an elaboration that can be extremely effective if one has error covariance estimates that can be exploited. Weighted error norms are beyond the scope of this paper and, so, we express Shum's formulation as:

$$\min \quad ||\mathbf{M} - \hat{\mathbf{R}}\hat{\mathbf{S}}||^2_{F\_\text{nonmissing.}} \tag{4}$$

In essence, (4) predisposes one to directly seek the factors, and to perform imputation and denoising together. This suggests different implementation strategies, but the solutions to both formulations should be equivalent. Of course, given different implementation strategies, the stability and convergence properties can differ.

2.2 Nonlinearity of the Problem

Obviously, Shum's formulation (4) is nonlinear: In fact, it is bilinear in the factors $\mathbf{R}$ and $\mathbf{S}$. Here, we show the intrinsic nonlinearity of our formulation (3).

Suppose $\mathbf{M} \in R^{m,n}$. Its closest rank-r matrix, measured by the Frobenius norm, is $\mathbf{M}^r = \mathbf{U}^r\mathbf{\Sigma}^r(\mathbf{V}^r)^T = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, with $\|\mathbf{M} - \mathbf{M}^r\|_F^2 = \sum_{i=r+1}^p \sigma_i^2$ [8], where $p = \min(m,n)$ and $\{\sigma_i^2\}$ are the nondescending eigenvalues of $\mathbf{M}^T\mathbf{M}$.

Suppose $\mathbf{M}$ has some missing entries $\{M_{i,j}|(i,j) \in \Xi\}$, where $\Xi = \{(i,j)|M_{i,j} \text{ is unknown}, 1 \le i \le m, 1 \le j \le n\}$. $\mathbf{E}_{i,j} \in R^{m,n}$, has all zero entries, except a one at (i,j). Let the recovered matrix be $\hat{\mathbf{M}}$, $\hat{\mathbf{M}} = \bar{\mathbf{M}} + \sum_{(i,j)\in\Xi} k_{i,j}\mathbf{E}_{i,j}$, where

$$\bar{M}_{i,j} = \begin{cases} M_{i,j} & (i,j) \notin \Xi \\ 0 & (i,j) \in \Xi. \end{cases}$$

The characteristic polynomial of $\hat{\mathbf{M}}^T\hat{\mathbf{M}}$, $p(\lambda)$, is a high-order polynomial of λ and $k_{i,j}$. The equation, $p(\lambda) = 0$, has n nonnegative roots for any $\{k_{i,j}\}$, because $\hat{\mathbf{M}}^T\hat{\mathbf{M}}$ is positive semidefinite. The problem reduces to finding $\{\hat{k}_{i,j}\}$, which minimizes the sum of the least $n - r$ roots of the equation, $p(\lambda) = 0$. This is a nonlinear problem.

Consider a simple case, $\mathbf{M} \in R^{10,10}$ with a missing entry $M_{1,1}$. Suppose $\mathbf{M}$ should be of rank 4, if it were noise free and had no missing entries. Its characteristic polynomial, $p(\lambda, t)$, where t denotes the missing entry, is of the form: $p(\lambda,t) = \lambda^{10} + f_2(\lambda)t^2 + f_1(\lambda)t + f_0(\lambda) = \lambda^{10} + \sum_{i=0}^9 \lambda^i g_i(t)$, where $f_i(\lambda) = \sum_{j=0}^{j=9} f_{i,j}\lambda^j$ and $g_i(t) = \sum_{j=0}^{j=2} g_{i,j}t^j$, and $f_{i,j}$ and $g_{i,j}$ are determined by $\mathbf{M}$. This equation is nonlinear and the problem of minimizing the sum of the least *six* roots is very complicated. If there are many missing entries in the matrix, the problem appears intractable from this point of view.

3 An Iterative Imputation Method

In this section, an iterative algorithm, based on the imputation principle, is proposed, and we prove a weak convergence of the iterative algorithm.

3.1 Minimization of the Distance of a Vector with Missing Entries to a Known Subspace

The key starting point is to "grow" a complete matrix by adding rows or columns, filling in those missing entries in the new rows or columns. Without loss of generality, we consider only the case of column-wise growth of the complete matrix. Thus, suppose we have a complete matrix, $\mathbf{M} \in R^{m,n}$, which should be of rank r ($r \leq m, n$) if it were noise-free; and another vector $\mathbf{x} \in R^m$, with missing components. Ideally, $[\mathbf{M}, \mathbf{x}]$ should be also of rank r if both of them were noise-free and complete. Suppose the first k ($k \leq m - r$) components of $\mathbf{x}$ (i.e., $\mathbf{x}_{1:k}$) are missing (swapping rows if necessary). The imputation method finds a linear combination of column vectors in $\mathbf{M}$, fitting $\mathbf{x}$ the best [2], [27]:

$$\hat{\mathbf{x}}_{1:k} = \mathbf{U}_1(\mathbf{U}_2^T\mathbf{U}_2)^{-1}\mathbf{U}_2^T\mathbf{x}_{k+1:m}, \tag{5}$$

where, by SVD, the rank-r projection of $\mathbf{M}$ is

$$\mathbf{M}^r = \mathbf{U}diag(\mathbf{s})\mathbf{V}^T = \begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{bmatrix}diag(\mathbf{s})\mathbf{V}^T,$$

and $\mathbf{U}_1$ is the upper k rows of $\mathbf{U}$ and $\mathbf{U}_2$ is the rest of $\mathbf{U}$.

Intuitively: $\hat{\mathbf{x}}$ is the closest point to the subspace $Span\mathbf{U}$. Because this property is crucial in proving the convergence in Section 3.3, we give a formal proof here.

Theorem 1. *The estimate $\hat{\mathbf{x}}$, obtained from (5), is the closest point to the subspace $Span\mathbf{U}$.*

Proof. For any estimate, $\tilde{\mathbf{x}}$, suppose $\mathbf{U}^T\tilde{\mathbf{x}} = \tilde{\mathbf{c}}$:

$$\left\|\tilde{\mathbf{x}} - \mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}\right\|_F^2 = \|\tilde{\mathbf{x}} - \mathbf{U}\tilde{\mathbf{c}}\|_F^2 = \|\tilde{\mathbf{x}}_{1:k} - \mathbf{U}_1\tilde{\mathbf{c}}\|_F^2 + \|\mathbf{x}_{k+1:m} - \mathbf{U}_2\tilde{\mathbf{c}}\|_F^2 \geq \|\mathbf{x}_{k+1:m} - \mathbf{U}_2\hat{\mathbf{c}}\|_F^2,$$

where the equality holds *iff* $\tilde{\mathbf{c}}$ is the LS solution $\hat{\mathbf{c}}$ for $\mathbf{U}_2\mathbf{c} = \mathbf{x}_{k+1:m}$ and $\tilde{\mathbf{x}}_{1:k} = \mathbf{U}_1\hat{\mathbf{c}}$. □

Note: Although the solution by (5) is optimal in terms of the distance between the vector with missing data and the known subspace, it is not true for the new subspace of $[\mathbf{M}, \hat{\mathbf{x}}]$; because the new subspace depends not only on M, but also on $\hat{\mathbf{x}}$.

3.2 An Iterative Algorithm for the Problem (*Iter*)

In this section, we present an iterative algorithm (called *Iter*) to solve the nonlinear problem defined in Section 2.1. Though Iter performs well in the vast majority of cases, it does not *always* converge to a good solution. Hence, this core algorithm will be improved in Section 4.

Algorithm (*Iter*)

1. ***Starting from a complete submatrix***: Suppose, without loss of generality, that $\mathbf{M}$, after some row and column exchanges, has a block representation:

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix},$$

where all entries in $\mathbf{A}$ are known, and some entries in $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are missing. For example, permute columns so that columns with least missing values are on the left and permute rows so that rows with least missing values are toward the top. We do not need the largest submatrix—any $\mathbf{A}$ of size $2r \times 2r$ or larger will do.

2. ***Initialization—growing a complete submatrix***:

   a. *Column-wise filling.* First, consider the submatrix $[\mathbf{A} \quad \mathbf{B}]$. Recover $\hat{\mathbf{B}}$ from $\mathbf{A}$ by (5) and obtain

$$\begin{bmatrix}\mathbf{A} & \hat{\mathbf{B}}_1 & \mathbf{B}_2\\ \mathbf{C} & \mathbf{D}_1 & \mathbf{D}_2\end{bmatrix},$$

   where the missing entries in $\hat{\mathbf{B}}_1$ have been recovered and the missing entries in $\mathbf{B}_2$ cannot be recovered. **Note**: This induces a split of submatrix $\mathbf{D}$.

   b. *Row-wise filling.* Similarly, recover $[\mathbf{C} \quad \mathbf{D}_1]$ from $[\mathbf{A} \quad \hat{\mathbf{B}}_1]$, and obtain

$$\begin{bmatrix}\mathbf{A} & \hat{\mathbf{B}}_1 & \mathbf{B}_2\\ \hat{\mathbf{C}}_1 & \hat{\mathbf{D}}_{11} & \mathbf{D}_{12}\\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{D}_{22}\end{bmatrix}.$$

   Note: After Step (b),

$$\begin{bmatrix}\mathbf{A} & \hat{\mathbf{B}}_1\\ \hat{\mathbf{C}}_1 & \hat{\mathbf{D}}_{11}\end{bmatrix}$$

   is now $\mathbf{A}$, the complete submatrix, in the block representation of

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}; \begin{bmatrix}\mathbf{B}_2\\ \mathbf{D}_{12}\end{bmatrix},$$

   $[\mathbf{C}_2 \quad \mathbf{D}_{21}]$ and $\mathbf{D}_{22}$ now are $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, respectively.

After Step (a), check whether all the missing entries have been recovered. If so, terminate the initialization step and go to the iteration step; if not, go to Step (b). After Step (b), check for completion again. If all the entries have been recovered, go to the iteration step. If not, check the following condition: Is the number of the nonrecovered entries before Step (a) the same number as after Step (b)? If so, the missing entries in $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ cannot be recovered. If the number of nonrecovered entries decreases, continue the initialization Step (a) by regarding the recovered entries as "nonmissing." (**Note**: Although growing the complete submatrix to obtain the initial complete matrix, as described here, is somewhat iterative, we prefer to view this as an initialization step to the refinement iterations that follow in the next step.)

After this initialization procedure, we obtain a recovered matrix $\hat{\mathbf{M}}_1$, which is complete; and we prepare for the iterative stage by setting a convergence measure $d_0 = \infty$.

3. ***Iteration-refining the complete matrix***: From $\hat{\mathbf{M}}_i$, obtain its closest rank-r approximation by SVD: $\hat{\mathbf{M}}_i^r = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^T$. Compute the rank-$r$ approximation error $d_i = \|\hat{\mathbf{M}}_i^r - \hat{\mathbf{M}}_i\|_F$. If

$$d_{i-1} - d_i < \varepsilon, \tag{6}$$

terminate the iteration; else, from $\mathbf{U}_i$, recover the missing entries in $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ by (5), and obtain $\hat{\mathbf{B}}_{i+1}$, $\hat{\mathbf{C}}_{i+1}$, and $\hat{\mathbf{D}}_{i+1}$. Set

$$\hat{\mathbf{M}}_{i+1} = \begin{bmatrix} \mathbf{A} & \hat{\mathbf{B}}_{i+1} \\ \hat{\mathbf{C}}_{i+1} & \hat{\mathbf{D}}_{i+1} \end{bmatrix}.$$

3.3 The Convergence of the Iterative Algorithm

In this section, we prove a weak convergence of the iterative imputation algorithm above (thus, the algorithm is independent of the initial matrix when the matrix has not been badly corrupted by the noise or by the missing data—as experimentally verified).

Theorem. *The iterative algorithm above converges to a local minimum.*

Proof. Suppose $\mathbf{m}$ is an arbitrary column of $\mathbf{M}$, and its estimates are $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{m}}_{i+1}$ at the ith and the $(i+1)$th iteration steps, respectively.

$$\begin{aligned} \left\|\hat{\mathbf{M}}_i^r - \hat{\mathbf{M}}_i\right\|_F^2 &= \sum_{all\ \mathbf{m}} \left\|\hat{\mathbf{m}}_i - \mathbf{U}_i\mathbf{U}_i^T\hat{\mathbf{m}}_i\right\|^2 \\ &\geq \sum_{all\ \mathbf{m}} \left\|\hat{\mathbf{m}}_{i+1} - \mathbf{U}_i\mathbf{U}_i^T\hat{\mathbf{m}}_{i+1}\right\|^2 \\ &\geq \sum_{all\ \mathbf{m}} \left\|\hat{\mathbf{m}}_{i+1} - \mathbf{U}_{i+1}\mathbf{U}_{i+1}^T\hat{\mathbf{m}}_{i+1}\right\|^2 \\ &= \left\|\hat{\mathbf{M}}_{i+1}^r - \hat{\mathbf{M}}_{i+1}\right\|_F^2. \end{aligned}$$

The first inequality is from Theorem 1, and the second from the SVD theorem [8]. □

Note: There are many ways to detect/characterize convergence. Another condition for the convergence, not so rigorous as (6), is to check the variation of the missing entries, i.e.,

$$||\hat{\mathbf{M}}_{i+1} - \hat{\mathbf{M}}_i||_F < \varepsilon'. \tag{7}$$

Condition (7) is easier to check. However, (7) is stronger than (6), and it may happen that (7) fails to indicate convergence. The cases, nonconvergent measured by (7), are described as ***divergent*** in Section 5.1 and Section 5.2.

4 SVD's Denoising Capacity versus Missing Data

Vectors, with only a few "nonmissing" components, may cause the iteration to "wander away" from the true solution. Moreover, even if the optimal solution, defined as in (3), can be obtained,[1] we experimentally find that these recovered vectors may degrade the accuracy that might have been gained from the other reliable data, alone. We have experimented, with some success with various strategies to detect and rectify this (see the Appendix), however, the true solution will be found in a closer analysis of the denoising process. By analyzing the SVD's denoising capacity [5], we present a criterion to decide whether it is worth incorporating an incomplete vector into the iteration.

1. With synthetic data, or real data with artificial occlusion, it is, of course, easy to check for divergence and to assess how badly the solution has been degraded by the addition of one or more columns with large missing data and/or large amounts of noise.

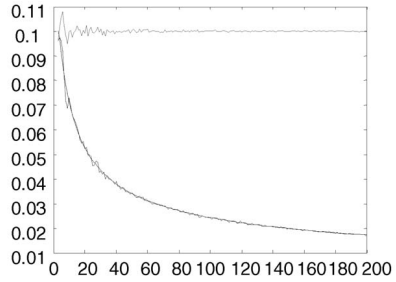


Fig. 1. SVD's denoising capacity. The abscissa is the number of columns of the square matrix, and the ordinate is the error in the rank-4 approximation matrix. Three curves are drawn: the upper one is the RMS error in the noise-corrupted matrix, the smooth dashed one is the expectation of the RMS error in the rank-4 approximation matrix, and the other nonsmooth one is the result, simulated by the computer. The latter two traces are so close as to be hard to distinguish from each other—confirming the theory.

4.1 SVD's Denoising Capacity and Its Extension to an Incomplete Matrix

In [5], with the tool of the matrix perturbation theory [28], the SVD's denoising capacity is analyzed, in terms of the size of the matrix, the noise level, and the underlying rank. More formally, it is depicted by the following fact [5].

Result 1 (Denoising capacity of SVD). Suppose a matrix $\mathbf{A} \in \mathbf{R}^{m,n}$ has a rank of $r (r << m, n)$. It is corrupted by i.i.d. Gaussian noise producing another matrix $\mathbf{B}$, which is directly observed. Then, the expected error that still resides in the rank-r approximation matrix, $\mathbf{B}^r$, is

$$E|B_{i,j}^r - A_{i,j}| = \sigma\sqrt{\frac{r(m+n) - r^2}{mn}} \tag{8}$$

if the noise level σ, compared with the signal level, is small enough. Especially, as $m, n \to \infty$, the rank-r approximation of $\mathbf{B}$ approaches $\mathbf{A}$, i.e., $\mathbf{B}^r \to \mathbf{A}$; and if $n \equiv k$ $(k \geq r)$ and $m \to \infty$,

$$E|B_{i,j}^r - A_{i,j}| \to \sigma\sqrt{\frac{r}{k}}. \tag{9}$$

The advantage of the SFM factorization method can be ascribed to the SVD's denoising capacity. From (8), we can see, as the size of the matrix increases, the low-rank approximation matrix approaches the noise-free matrix. That is, the underlying superiority of the factorization method when applied to a complete matrix: All the feature points are treated uniformly so that most of the noise can be suppressed if the size of the measurement matrix is large enough. Fig. 1 shows the SVD's denoising capacity.

However, SVD is not directly applicable when there is some missing data in the matrix. A possible solution is to first recover the missing data, using, for example, the iterative imputation method above; then to SVD the recovered matrix. However, when there are a lot of missing components, a vector with only a few "nonmissing" components, might degrade the accuracy obtainable from the other reliable data. Yet, using only a small complete submatrix may not achieve optimal denoising ability—clearly there is a trade off here. This is illustrated in Fig. 2: as the missing percentage increases, the performance deteriorates. *A natural question arises: Is it possible to find a submatrix, complete or incomplete, which is more reliable than the whole matrix?* From (8), the

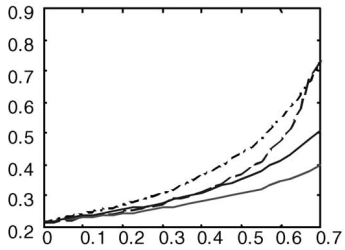


Fig. 2. The optimal performance of the iterative algorithm. The abscissa is the missing percentage, ρ and the ordinate is RMS error for the iterative algorithm. Four curves are drawn: the upper dotted one is $(1-\rho)^{-1}$, the lower solid one$(1-\rho)^{-1/2}$, the solid one in the middle is $(1-\rho)^{-0.7}$, and the dashed one in the middle is the optimal performance of the proposed iterative algorithm (Section 3.2).

denoising capacity of the SVD is dependent on the ratio between $(m+n-r)r$ and mn: The former is the number of the independent elements of the low-rank matrix [23] and the latter is the number of the variables in the matrix.[2] From this fact, we postulate that the incomplete matrix approximately has similar "denoising capacity."

Hypothesis 1 (the denoising capacity of the incomplete matrix). *Suppose there are p ($p \geq (m+n-r)r$) "nonmissing" components in a matrix $\mathbf{B}$, and each row (column) has at least r "nonmissing" components. The best estimate of $\mathbf{B}$, $\hat{\mathbf{B}}$, should have the following property:*

$$E|\hat{B}^r_{i,j} - A_{i,j}| = \sigma\sqrt{\frac{r(m+n)-r^2}{p}} = \sigma\sqrt{\frac{r(m+n)-r^2}{mn}}\sqrt{\frac{mn}{p}} = \sigma\sqrt{\frac{r(m+n)-r^2}{mn}}\sqrt{\frac{1}{1-\rho}}, \tag{10}$$

where ρ is the percentage of the missing data.

Compared with the denoising capacity of the complete matrix, the error in the incomplete matrix should increase by $\sqrt{\frac{1}{1-\rho}}$ as a function of the missing percentage. The RMS error index of the iterative algorithm approximately follows $(1-\rho)^{-0.7}$ (see Fig. 2), when the percentage is less than 0.5—not exact agreement but still useful.[3]

We employ (10) as a criterion as to whether it is worth incorporating a vector, with missing data, into the iteration. For an incomplete matrix with a rank of r, all of whose columns and rows have at least r "nonmissing" components, we define its ***unreliability*** as the ratio between the number of its independent variables and the number of nonmissing components:

$$c = \frac{r(m+n)-r^2}{p}. \tag{11}$$

Thus, we propose to use the following strategy: first, use the iterative algorithm in Section 3.2, to recover the most ***reliable*** incomplete submatrix, which has the minimal unreliability ratio; then, project other columns (rows) on it, if required, using the imputation method. Specifically for SFM, our strategy is: First, reconstruct the 3D scene and the cameras by the factorizing *the most reliable measurement matrix* (obtained by the algorithm in Section 4.3); then to estimate the positions of *other* feature points and *other* camera matrices, using the techniques in [26].

4.2 The Minimal Unreliability Ratio in SFM

It is an NP-hard problem to find the submatrix that has the minimal unreliability ratio. Here, we propose a simple approach: to iteratively exclude the vector(s), which has the least "nonmissing" components among the retained submatrix, until the unreliability ratio begins to increase. Obviously, only a local minimum can be obtained, in general. However, in many cases, such as the SFM problem and the recommender system, we usually have a ***thin*** matrix, i.e., it has a large width or height. In the following discussion, we suppose, without loss of generality, we have an incomplete matrix whose width is much larger than its height. We then *sort the columns* so that the columns with the least missing entries are toward the left. Now, we simply must find a "cut" point, beyond which to exclude unreliable columns. Indeed, if we restrict the exclusion to columns, the optimal property can be proven. Without loss of generality, suppose $n >> m >> r$, and the nonmissing number in the ith column, k_i, is descending, i.e., $k_i > k_{i+1}$ for $1 \leq i < n$. The unreliability ratio of the submatrix $\mathbf{M}_l$ (the left l columns of $\mathbf{M}$), is:

$$c_l = (m+l-r)r \Big/ \sum_{i=1}^{l} k_i. \tag{12}$$

We only need to prove: $c_l > c_{l+1} \Rightarrow c_{l-1} > c_l$ and $c_l < c_{l+1} \Rightarrow c_{l+1} < c_{l+2}$. That is, the curve c_l has one minimum. The first can be easily proven: $c_l > c_{l+1} \Leftrightarrow c_l > \frac{r}{k_{l+1}} \Rightarrow c_l > \frac{r}{k_l} \Leftrightarrow c_{l-1} > c_l$. Please note c_l, r, k_l are positive numbers. The second fact can be similarly proven.

4.3 Algorithm (*IterPart*)

In this section, we propose another algorithm, which still uses ***Iter***, in Section 3.2, at its core.

1. Use quick cull of cols(rows) that are not reasonable to iteratively impute (Section 4.2).
2. Use the "sweeping" initialization of the core algorithm (Section 3.2). This could be augmented with a bootsrapping strategy (Appendix), but such appears to be unnecessary in all of our experiments.
3. Use error norm monitored iteration of the core algorithm to convergence (i.e., the iteration step in ***Iter***, in Section 3.2).
4. Finally, recover the "hopeless" (the entries not recovered by 1-3), if one really must, with another approach—e.g., Tomasi-Kanade. These portions may not be recovered well, but at least they will be somewhat recovered and they will not pollute the accuracy of the previously recovered portion.

In short, ***IterPart*** is an improvement on ***Iter*** that benefits from our heuristic approach to deciding which of the entries are worth recovering directly. The difference is in Step 1 (to predetermine where the core of reliable information is likely to be) and Step 4 (optional recovery of those parts that are likely to be unreliable). This approach tends to converge more often than other methods.

2. Please note that in Shum's formulation [23], the mean is also considered so, in that case, there are $(m+n-r)r+n$ independent variables.

3. The exponent may vary in different settings: with different-size matrix or with different underlying rank. However, the optimal performance is generally better than $(1-\rho)^{-1}$.

4.4 Discussion

IterPart (Section 4.3) performs almost the same as ***Iter*** (Section 3.2) *when there are only a few missing components.* Suppose the matrix is very large: $n >> m >> r$. Then, the unreliability ratio for the complete matrix is about r/m. Thus, if each column (or a row) has less than r (or nr/m) missing components, the whole matrix is the most reliable one; i.e., ***IterPart*** is the same as ***Iter***. Moreover, if the missing percentage is comparatively low, both of them are expected to have similar performance, as will be validated by experiments.

When there are a lot of missing components, ***IterPart*** should perform better than the ***Iter***. Generally, each column (row) in the most reliable submatrix has more than *2r* nonmissing components; because the most reliable matrix would generally have an unreliability ratio less than *0.5*. If the matrix can be recovered, there should be $(m + n - r)r$ nonmissing components at least, i.e., the unreliability should be less than *1*. The unreliability ratio decreases as a result of the cutting processes. The vectors with only r nonmissing components are retained in the most reliable matrix *only* if the whole incomplete matrix has an unreliability ratio of 1. We also note that ***Iter*** has a risk of divergence, even when employing an additional "bootstrapping" strategy outlined in the Appendix. ***IterPart*** generally, does not have such problems, even without the aid of bootstrapping, as will be demonstrated by experiments.

5 Experiments

In this section, we compare the performance of eight approaches: ***Iter*** (proposed in Section 3.2), its variant: ***IterPart*** (proposed in Section 4.3), Jacobs's three methods: (***rankr*** "***Jacobs1***," ***rankrsfm*** "***Jacobs2***," and ***rankrsfm_tpose*** "***Jacobs3***"), and Shum's method, also with three variants: ***Shum1***, ***Shum2***, ***Shum3*** (starting from Jacobs' methods above). We use rank4 versions of Jacobs routines, sidestepping the erroneous centroid subtraction in the presence of missing data [11], [16]. We present three groups of experiments, one using synthetic data, another from the *box* sequence, which was also used by Jacobs [14], [15], and the other from the *dinosaur* sequence, which is somewhat more challenging.

We focus on stability since ***Iter***, ***Shum1***, ***Shum2***, and ***Shum3*** have almost the same performance *when they converge*. ***IterPart*** has a very small risk of divergence. It should be very stable because only the most reliable submatrix is used in the iteration, where each row (column) *generally* has more than *2r* nonmissing components and $r + 1$ at least. Indeed, no divergent case has been found in all 20,000 cases we examined (20-noise-level $\times$ 10-level-of-missing-percentage $\times$ 100-times repetition).

5.1 Synthetic Data in an 8-Frame-and-40-Point Sequence

As in [11], [16], all the synthetic image data is generated this way: The 3D feature points are uniformly distributed in a cube, within [-500, 500]*[-500, 500]*[-500, 500] units; the cameras are placed around 1,000 units away from the origin. Thus, the 2D image size is about 500*500. Then, different levels of Gaussian noise, standard deviation from 1 to 20, are added into the 2D feature points. Because the proposed algorithm has to start from a complete submatrix, we suppose that the first 8×8 submatrix is always nonmissing and the missing entries are then randomly distributed in the rest of the matrix. In addition, in order to have a recoverable incomplete matrix, we make sure that each row/column of the incomplete matrix has four nonmissing entries at least. The simulation repeats 100 times for each setting.

The experimental results under noise level of 1, 5, 10, 15, and 20 are shown in Fig. 3. Please note, we do not include those ***divergent*** cases for the approaches of ***Iter***, ***Shum1***, ***Shum2***, and ***Shum3*** (if the RMS of any iterative algorithms has a magnitude of three times or more than the noise level, the algorithm is regarded divergent); because the divergent cases would require a greatly expanded RMS axis. Fig. 4 depicts the convergence rate for the iterative algorithms. Since the convergence rate is strongly dependent on the missing percentage, we only compare the average convergence rates (over different noise levels) for the same missing percentage.

We can see, from Fig. 3, that the proposed iterative algorithm (***Iter***) has almost the same performance as Shum's, and that these four curves (***Iter*** and the three versions of Shum's) merging into the second lowest trace. Another conclusion is that the more stable variant of our method (***IterPart***) shows its superiority when there is a lot of missing data, performing much better than ***Iter*** and ***Shum***, as expected from Section 4. Of Jacobs' methods, the ***rankrsfm*** performs best, good enough to be the initial point for the iterative algorithms. Note, ***rankrsfm_tpose*** is much worse than ***rankr***. Though the three versions of Shum's algorithm (starting from the three versions of Jacobs as their initial matrix) perform identically with ***Iter*** when they converge, Fig. 4 shows that ***Iter*** generally converges at least as reliably. Note, the improved algorithm, ***IterPart*** converges in 100 percent of the experiments.

5.2 Box Sequence

Here, to test the algorithms on real data, we use the box video, which was used in [14], [15]. The sequence consists of 40 feature points across eight frames. One frame is shown in Fig. 5. As in Section 5.1, we suppose that eight points in four frames are available. This 8×8 submatrix is randomly selected. We then randomly occlude (consider as missing) the other feature points.

For this example, as shown in Fig. 6, the five methods have almost the same performance: ***Iter***, ***IterPart***, ***Shum1***, ***Shum2***, and ***Shum3***.

5.3 Dinosaur Sequence

Here, we present an example where some data is truly missing (i.e., not artificially occluded to simulate missing data). 4,983 feature points were tracked over the 36-frame "dinosaur" sequence [7] and the 20th frame is shown in Fig. 7, where the feature points are denoted by symbol "+". The feature points, extracted by the Harris interest operator [10], were obtained from Oxford (http://www.robots.ox.ac.uk/~vgg/data/). Over the dinosaur sequence, about 90.84 percent of the data is missing and the mask of the tracked feature points is shown in Fig. 8, where a black pixel in (i, j) means the ith feature point (in abscissa) is tracked in the jth frame (in ordinate) and a gray pixel denotes the occlusion/missing data. Under the assumption of the affine camera, the measurement matrix should lie in a four-dimension subspace. However, in this example, the perspectivity factor is not negligible and the four-dimensional subspace does not fit

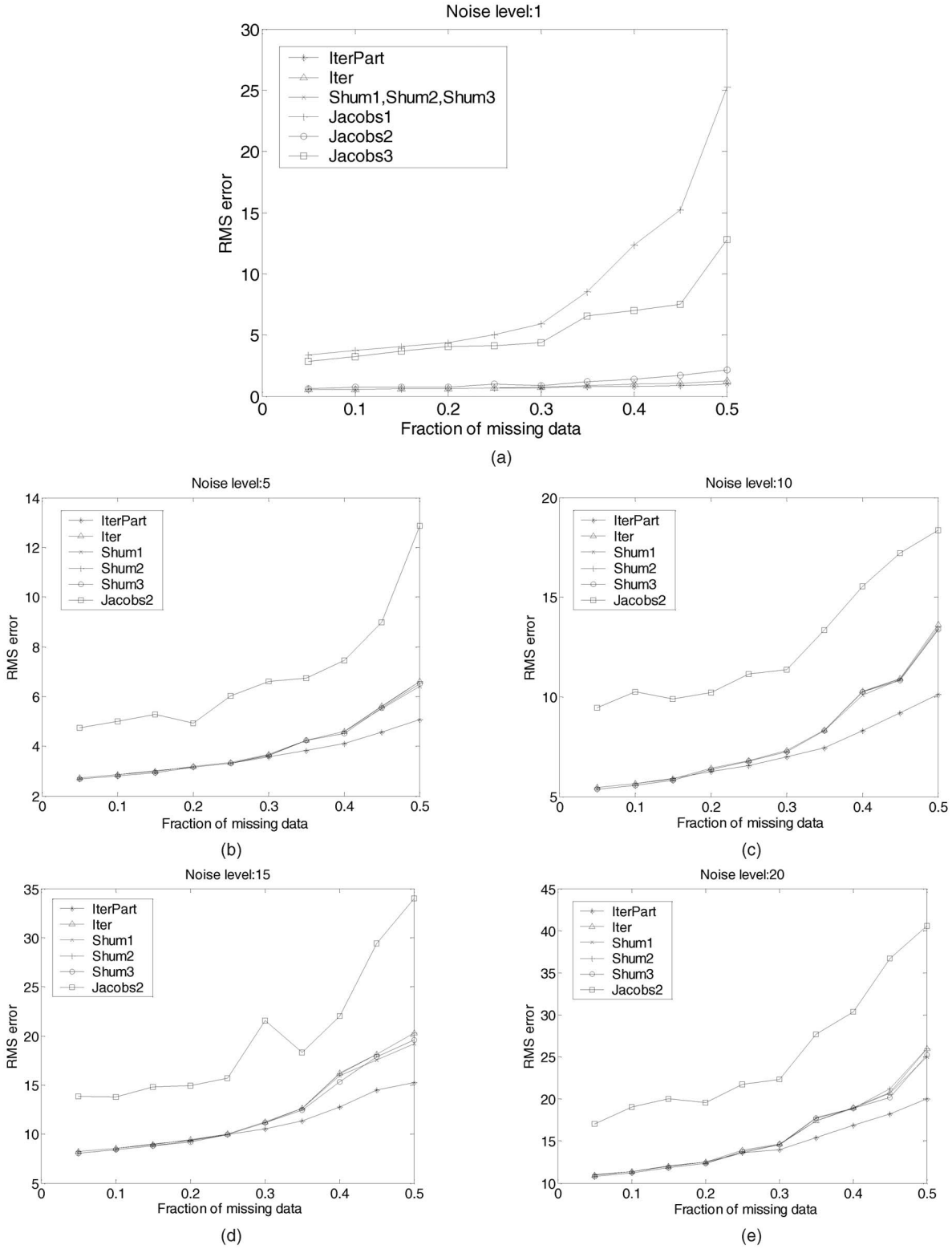


Fig. 3. The reprojection RMS error of the eight methods, as described in the beginning of Section 5. The abscissa is the missing percentage, and the ordinate is the reprojection RMS error. In (a), we depict all eight methods when the noise level is only 1 (from the best to the worst, they are ***IterPart***, ***Iter*** (and ***3 Shums***), ***rankrsfm***, ***rankr***, and ***rankrsfm_tpose***); while, in (b), (c), (d), and (e), with noise levels of 5, 10, 15, and 20, respectively, only six methods: three versions of Shum's method, ***Iter***, ***IterPart***, and the best Jacobs' method ("***rankrsfm***"), are depicted, in order to make the comparison visible. ***IterPart*** is the best, and ***rankrsfm*** is the worst one, and the other four have almost the same performance.

the feature points well even without other noise. Thus, the model error, as well as the error introduced in the feature extraction, makes it a challenging task to recover these missing feature points. We note that the projective model was used, by Martinec and Pajdla [18], to recover the dinosaur sequence. It is beyond the scope of this paper to tackle such a setting, but we find that our results, even in the inferior affine setting, are approximately same, at least as far as one can

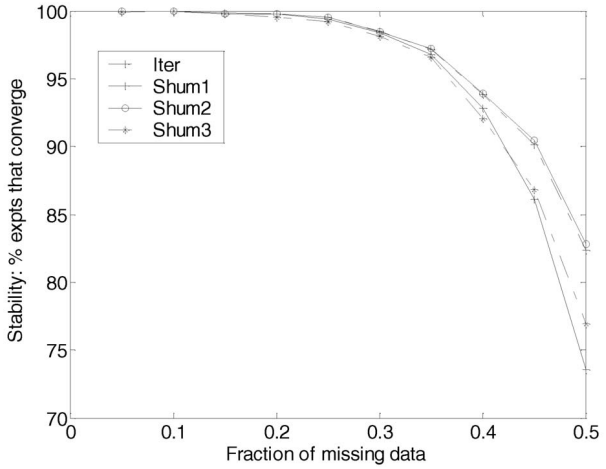


Fig. 4. The convergence rate of four iterative methods against the missing entry fraction. Dotted curve with plus (+): ***Iter***; solid curve with circle: Shum + rankrsfm; dotted curve with star (*): Shum + rankrsfm_tpose; and solid curve with plus (+): Shum + rankr.

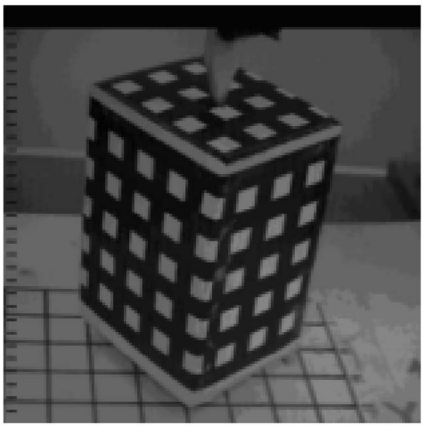


Fig. 5. One frame of the box sequence.

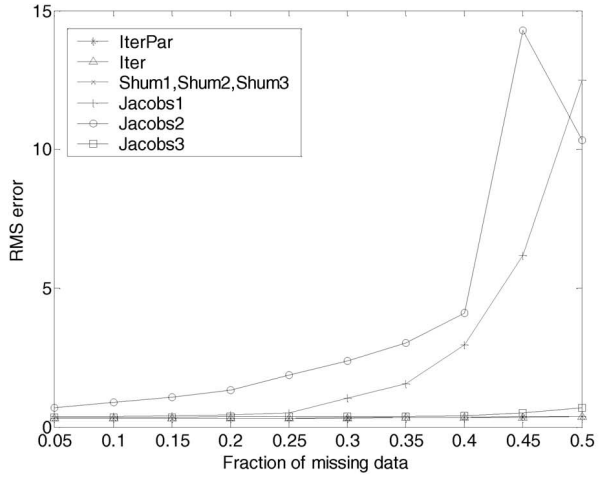
determine from gross statistics, as Martinec and Pajdla's results [18].

The core iterative algorithm (**Iter**) fails on the total sequence because of too much missing data and strong noise. By excluding the vectors with a few nonmissing components (***IterPart***), the most reliable matrix has 36 frames and 336 feature points, with an unreliability ratio of ***0.2892***, where each point has been tracked over more than six (> 6) frames, and each frame tracked more than 20 feature points. We compare all algorithms using this same subset of "reliable" data.

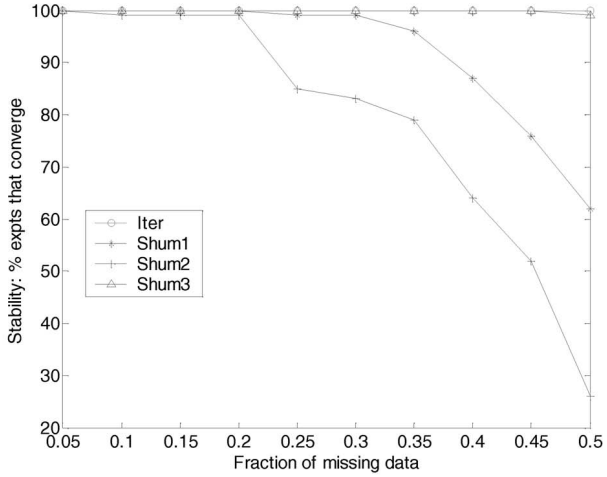
First, by the core iterative method in Section 3.2, we reconstruct the 336 ("most reliable") feature points, as shown in Fig. 9, where about 77 percent of the data is missing. The result by Jacobs' method under the affine camera, as shown in Fig. 10a, is unsatisfactory. When the initial result is not accurate enough, Shum's approach tends to diverge, or become trapped in a local minimum, as shown in Figs. 10b and 10c. The recovered tracks by the proposed method ***Iter*** are shown in Fig. 10d. (Which is, of course, the same as that by ***IterPart*** since we have pruned.)

By combining Jacobs' method [14], [15] and Sturm and Triggs' projective factorization method [24], a good result over the whole sequence was reported [18]: The mean reprojection error per image point, measured by pixels, was reported as 1.76 pixels and the maximal reprojection error was reported as 73.9 pixels. The mean error and maximal error were reported as 0.64 and 41.5 pixels (respectively) after bundle adjustment.

However, the above indexes (mean error and maximal error) may be sometimes misleading in assessing the performance of the algorithms as we demonstrate here. Using the stable variant of the proposed iterative method (***IterPart***), we conducted some experiments over two selections of the data: 1) the whole 4,983 feature points and 2) with only the 2,683 feature points that were tracked over more than two frames. (2,300 feature points were tracked only over two frames in the dinosaur sequence!) Our results of the



(a)



(b)

Fig. 6. The performance on the box sequence: (a) The RMS reprojection error of the eight methods are depicted: triangle (△) for five approaches (***Iter***, ***IterPart***, and three Shum approaches), circle (o) for rankrsfm_tpose, star (*) for rankr, and cross (+) for rankrsfm. Note: Five approaches (***Iter***, ***IterPart***, and three Shum approaches) have almost the same performance, so those five curves merged into one curve at the bottom. (b) The convergence rate of the four iterative methods are depicted: circle (o) for ***Iter***, triangle (△) for Shum + rankrsfm_tpose, star (*) for Shum + rankr, and cross (+) for Shum + rankrsfm.

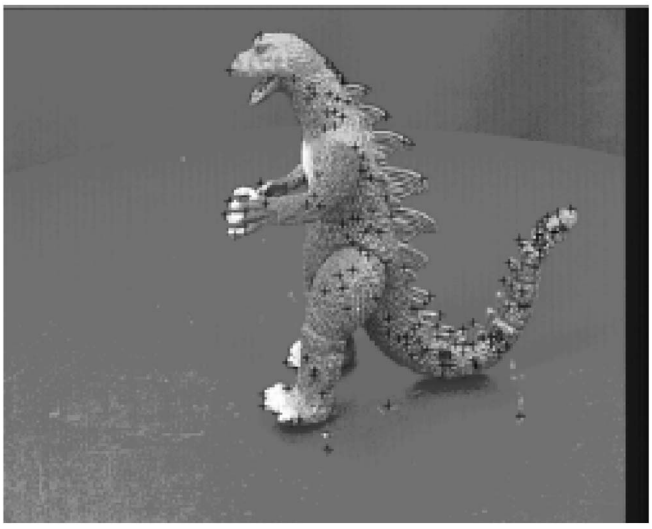


Fig. 7. The 20th frame of the dinosaur sequence.

Fig. 8. The missing data (gray) and measured data (black) for the dinosaur sequence.

reprojection tracks, for 4,983 and 2,683 feature points, respectively, are shown in Figs. 11a and 11b. Obviously, the result from 2,683 features is much better than that from 4,983 features. The recovered tracks should be approximately elliptical, because the sequence was taken while the dinosaur was on a rotating turntable [7]. Note: all the wild recovered tracks in the first experiment are from the 2,300 feature points which have been tracked over only two frames—thus, the likely reason for such sensitive behavior in Fig. 11a is that some feature points are tracked only over two frames (any noise in these features is likely to be influential). Contrast the visual quality with the impression conveyed by the mean/maximal error for 4,983 and 2,683 features, which are respectively 1.8438/72.4467 and 2.4017/72.4467 pixels; obviously, these measures alone are misleading since the reconstruction from the case with only 2,683 features scores worse although it has no wild recovered tracks. In fact, the mean/maximal error for the 2,300 feature points tracked over only two frames is only 0.4088/ 7.8093 pixels. Since we only have the measures, as reported by Martinec and Pajdla [18], it is not clear whether their results may have included such wild (and wrong) recovered tracks.

6 Conclusion

The main contribution of this paper is the development of a criterion one can use to recover the most *reliable* submatrix—i.e., to decide which parts of a matrix contain too many missing values to be included in the imputation. We also propose an iterative algorithm to employ the above criterion to the problem of missing data in a large low-rank matrix and we prove its convergence. In the cases, where the matrix has been badly corrupted by the missing data, the approach we propose is superior to other approaches. We avoid the NP-hard problem of finding the largest complete submatrix, as one does not need to start with a very large complete submatrix in our approach. Due to the convergence (toward the optimal solution, as demonstrated by the experiments), one can expect to arrive at the same solution even when starting from different complete submatrices.

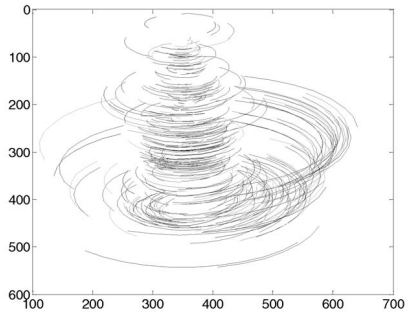


Fig. 9. The 336 tracked feature points over 36 frames.

As a result of our work, we also draw to the attention of the reader a salutary message regarding the use of simple error measures in making decisions about the superiority of one algorithm over another. It may be the case, as we demonstrated, that an approach with several very bad tracks, scores better than a method with generally very good tracks. Some care must be taken in assessing the contributions of studies that report only a single such measure (see also Appendix A.4).

Appendix

A.1 "Bad-Behavior" and a Bootstrapping Strategy

As noted in Section 1, for the proposed core iterative imputation method (***Iter***) only the convergence to a local minimum is proven. The worse case scenario is that, some components "wander away" from the underlying ground truth as the iteration proceeds. We call this phenomenon "bad behavior." Some vectors have polluted the first r components and the remaining data cannot "correct" the values that have "wandered." By an example [6], it has been shown that, if one data (an outlier) has 10 times the energy as the sum of the rest of the data, the outlier becomes the first principal component, and the first and the second original principal components become the other two principal components, approximately. Such a fact can be easily proven by the matrix perturbation theory [28], by regarding the outlier as the signal matrix and the original signal matrix as the perturbation.

If we followed the algorithm ***Iter***, outlined in Section 3.2, we may observe this *in a few cases* (although *very rare*, it *does* occur), when the percentage of the missing data is very high. The problem is with the initialization step. In the bad cases, the initialization step in the algorithm usually needs a few loops to obtain a complete initial matrix. However, no refinement is made on the newly increased submatrix before it continues to absorb other columns/rows.

In practice, such a phenomenon can be easily detected. From experiments, we found that the energy in $||\hat{\mathbf{M}}_{i+1} - \hat{\mathbf{M}}_i||_F^2$ concentrates in a few missing-values, mostly in one or two columns (rows). Having detected the likely "wandering," we can attempt to "purify" the matrix in the initialization phase or in the iteration phase. We can first regress these bad (one or two) columns (or rows) against the other columns (or rows) and then continue the iteration. Another is to restart the algorithm: In the initialization step, using those columns (or rows) that do not produce such bad behavior producing a

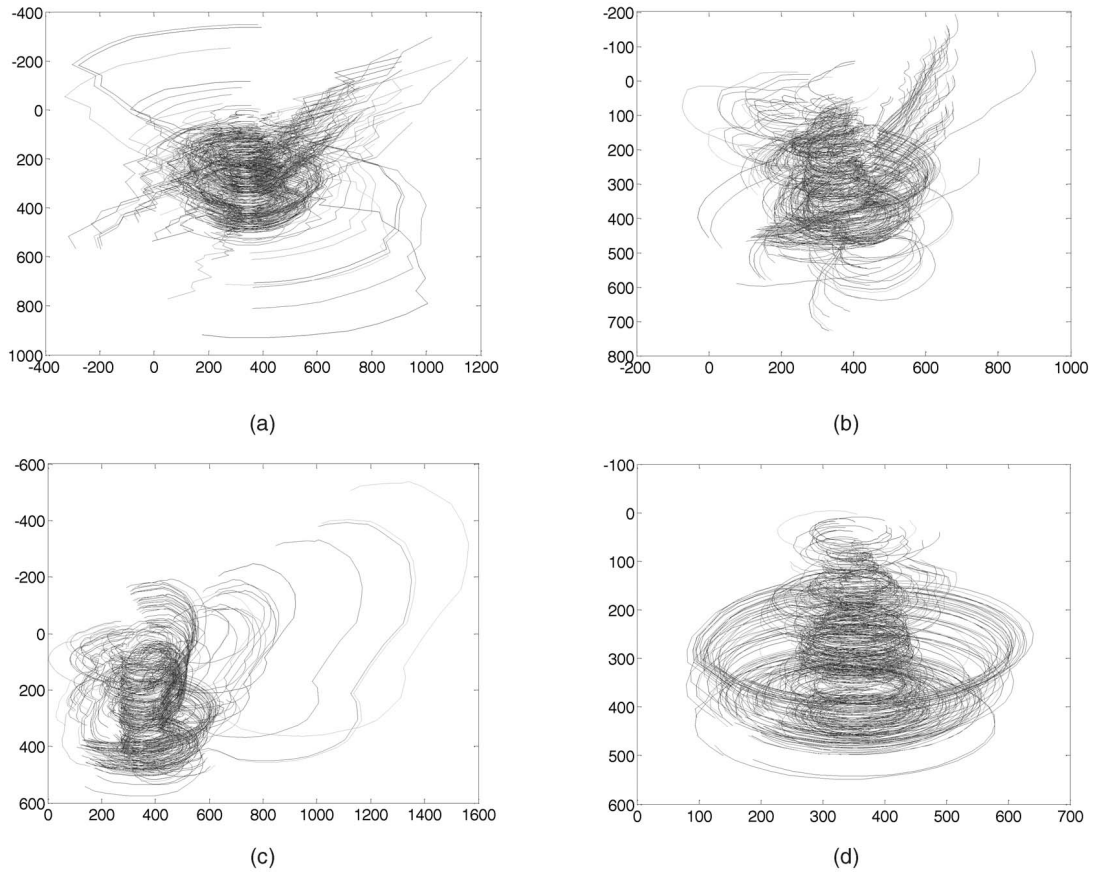


Fig. 10. The 336 recovered tracks by the Jacobs' and Shum's and the proposed methods: (a) Jacobs, (b) Shum's result after 100 iterations, (c) Shum's result after 400 iterations, and (d) the 336 recovered tracks by ***Iter***.

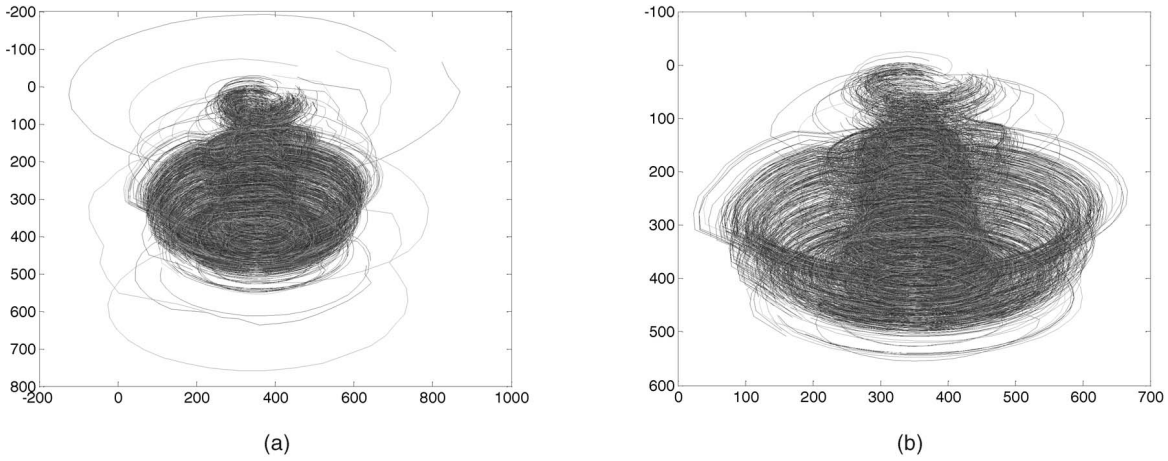


Fig. 11. The recovered tracks over 36 frames: (a) for 4,983 points and (b) for 2,683 points.

partially complete matrix; then regress the columns (rows) with bad behavior against the partially complete matrix before restarting the iteration. The second strategy, experimentally, performs better than the first.

The "afterward" bootstrapping strategy is not ideal because it is time consuming. Generally, the wandering-away behavior occurs with those columns, with only r (or slightly more than r) "nonmissing data," because the noise in such cases can be influential, especially when the subspace is ill-conditioned. For other columns, with only a few missing components, the imputation method of (5) is intrinsically an overdetermined system; therefore, it can resist noise to some extent and, consequently, it is unlikely that the wandering-away behavior occurs with these vectors.

Thus, we propose the following bootstrapping strategy[4] to overcome the wandering-away: To recover those columns (rows) with fewer missing values first, i.e., to recover the more stable vectors in the inner initialization loops. In order to reduce the computation loops in the initialization step, we

4. In [3], a similar bootstrapping strategy was employed to make the imputation method robust.

suggest that only those columns (rows), with more than (or equal to) $2r$ nonmissing data, should be incorporated into the complete submatrix, by using the imputation method (5).

Such a strategy raises another issue: in some cases, the complete submatrix stops increasing because no incomplete vector has more than (or equal to) $2r$ "nonmissing" values. In such cases, one can temporarily relax the constraint of requiring $2r$ "nonmissing values"—using columns (rows) with $2r-1$ "nonmissing values" (even as low as r if need be) to break the impasse and then resume with the more conservative demand of at least $2r$ "nonmissing values." This bootstrapping strategy can increase the robustness of the algorithm, especially when there are a lot of missing components; while it only incurs a little computation overhead—one or two more loops in the initialization step. However, we have found a similarly motivated procedure that makes this bootstrapping largely redundant (Section 4).

A.2 Revisiting the Objective Function in (3)

As stated in Section 2.1, our objective function is subtly different from that, used in Shum's approach [23]. However, under the strong convergence condition (7), the error index for the missing components, $\sum_{(i,j)\in\Xi}(\hat{M}_{i,j}-\hat{M}^r_{i,j})^2$, where $\Xi=\{(i,j)|M_{i,j}\text{ is unknown}\}$, approaches zero during the iterations. Thus, the objective function of (3), under the convergence condition of (7), is effectively same as Shum's objective function. It was proven by experiments that virtually the same solution is obtained by our method described in Section 3.2 and by Shum's method, *providing* both of them converge. In practice, our approach converges far more reliably.

A.3 Difference from the Imputation in [27]

The iterative algorithm in Section 3.2 (***Iter***) has been loosely anticipated by the method in [26]. Here, we describe the difference between the proposed algorithm in Section 3.2 and the algorithm in [26]. As noted in the introduction, the iterative imputation method in [27] cannot be shown to converge, although the iteration may stop after a few loops. The problem with the method in [27] lies in its updating procedure in the iteration. In [27], even if there is more than one missing component in one column, only one missing data is updated at a time; by regarding all other components known, including other missing data that has been estimated. Thus, k applications of updating are needed for a column, where k components are missing. **Note**: If every incomplete vector has only one missing entry (an entirely unlikely event), then the method is same as ***Iter***, outlined in Section 3.2. However, if there is more than one missing component the two are not equivalent and any method that can only recover one missing entry at a time raises the question: Which imputation order should be taken? After some components have been updated, should their old or new values be employed in the sequential estimation for other missing components? Generally, for any sequential updating, a different estimate from that, by (5), would be obtained, i.e., the estimate in [27] does not have the nice property that it is the closest point to the current subspace. Consequentially, no convergence can be promised in the iterative method in [27].

A.4 RMS and Reprojection Error

Generally, the root mean square (RMS) of the reprojection error is used to evaluate the performance of the reconstruction algorithm. However, the reprojection error index, in the real data sequence, may be misleading unless we have the ground truth. We illustrate the reason for our cautionary note here.

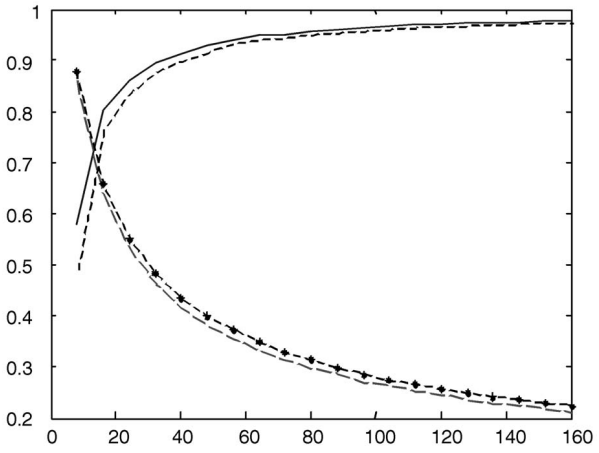


Fig. 12. The RMS errors in the low-rank approximation matrix and the reprojection RMS error: the abscissa is the size of the square measurement matrix and the ordinate is the RMS error. The dotted with star curve and the dotted one is the RMS errors of the rank-4 approximation matrix, compared with the noise-free matrix and the noise-corrupted matrices, respectively. The dashed one and the solid one denote the reprojection RMS errors, compared with the noise-free matrix and the noise-corrupted matrices, respectively.

In [5], it is proved that as the size of the matrix approaches infinite, its low-rank approximation approaches the underlying noise-free matrix. Consequently, for a very large matrix, if we compare its low-rank approximation with the noise-corrupted matrix, the residuals are approximately the added noise; yet if we compare with the ground truth (the uncorrupted matrix), the error should be around 0. From Fig. 12, we can observe this point: a series of synthetic measurement matrices ($\tilde{\mathbf{M}}$) are generated and i.i.d. Gaussian noise (0-mean-and-1-variance) is added, observing $\mathbf{M}$. The reprojection error, compared with $\tilde{\mathbf{M}}$ and $\mathbf{M}$, is depicted by the dashed curve and the solid curve, respectively. We also compare the rank-4 approximation of $\mathbf{M}$, $\mathbf{M}^4$, with $\tilde{\mathbf{M}}$ and $\mathbf{M}$, the error is depicted by the dot-with-star curve and the dotted curve, respectively. Obviously, the RMS indexes, against $\mathbf{M}$ (upper traces—"observed/noise corrupted" data), are misleading, in evaluating the performance. If we use the RMS error against the noise corrupted measurement matrix, the reconstruction error also increases as the size of the matrix increases (upper two traces); contrasting with an accepted fact that more frames produce more accurate reconstruction [19], [25]. In contrast, the lower two traces (using "ground truth") show the correct trend.

Because of this, we mainly rely on the synthetic data in evaluating the performance of the algorithms. In addition, please note that we use a different reprojection error index, from that in Jacobs' paper: In our work, the RMS error is obtained over the whole sequence, including those artificially occluded points. It makes little difference in most cases; however, the occluded points should be included in the evaluation, if possible, because in some pathological cases, we can find the reprojection error for the nonmissing data is comparatively small, while that for the whole data is very large. In Section 5.1, we can easily find such a case: with 50 percent data missing and a noise level of 10, where

the RMS error for the nonmissing entries is only 7.2098, while the RMS error for the artificially missing entries/all entries is 57.2773/38.2693, by the iterative algorithms.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for the suggestion of considering the "wandering behavior" issue in the iteration and their suggestions for improving the experiments and the manuscript.

REFERENCES

[1] P.M.Q. Aguiar and J.M.F. Moura, "Rank 1 Weighted Factorization for 3D Structure Recovery: Algorithms and Performance Analysis," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 25, no. 9, pp. 1134-1149, Sept. 2003.

[2] M. Brand, "Incremental Singular Value Decomposition of Uncertain Data with Missing Values," *Proc. European Conf. Computer Vision,* pp. 707-720, 2002.

[3] M. Brand, "Fast Online SVD Revisions for Lightweight Recommender Systems," *Proc. SIAM Third Int'l Conf. Data Mining,* 2003.

[4] S. Brandt, "Closed-Form Solutions for Affine Reconstruction under Missing Data," *Proc. Statistical Methods for Video Processing Workshop,* pp. 109-114, 2002.

[5] P. Chen and D. Suter, "An Analysis of Linear Subspace Approaches for Computer Vision and Pattern Recognition," Technical Report MECSE-6-2003, Monash Univ., Clayton 3800, Australia, http://www.ds.eng.monash.edu.au/techrep/reports/2003/MECSE-6-2003.pdf, 2003.

[6] F. DelaTorre and M.J. Black, "A Framework for Robust Subspace Learning" *Int'l J. Computer Vision,* vol. 54, pp. 117-142, 2003.

[7] A. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D Model Construction for Turn-Table Sequences, 3D Structure from Multiple Images of Large-Scale Environments," *Lecture Notes in Computer Science,* vol. 1506, pp. 155-170, 1998.

[8] G.H. Golub and C.F.V. Loan, *Matrix Computations,* second ed. Baltimore: Johns Hopkins Univ. Press, 1989.

[9] R.F.C. Guerreiro and P.M.Q. Aguiar, "Estimation of Rank Deficient Matrices from Partial Observations: Two-Step Iterative Algorithms," *Proc. Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition,* 2003.

[10] C.J. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Alvey Vision Conf.,* pp. 147-151, 1988.

[11] A. Heyden and F. Kahl, "Reconstruction from Affine Cameras Using Closure Constraints," *Proc. Int'l Conf. Pattern Recognition,* pp. 47-50, 1998.

[12] M. Irani, "Multi-Frame Optical Flow Estimation Using Subspace Constraints," *Proc. Int'l Conf. Computer Vision,* 1999.

[13] M. Irani, "Multi-Frame Correspondence Estimation Using Subspace Constraints," *Int'l J. Computer Vision,* vol. 48, no. 3, pp. 173-194, 2002.

[14] D. Jacobs, "Linear Fitting with Missing Data: Applications to Structure-from-Motion and to Characterizing Intensity Images," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 206-212, 1997.

[15] D. Jacobs, "Linear Fitting with Missing Data for Structure-from-Motion," *Computer Vision and Image Understanding,* vol. 82, pp. 57-81, 2001.

[16] F. Kahl and A. Heyden, "Affine Structure and Motion from Points, Lines, and Conics," *Int'l J. Computer Vision,* vol. 33, no. 3, pp. 163-180, 1999.

[17] K. Kanatani, "Motion Segmentation by Subspace Separation and Model Selection," *Proc. Int'l Conf. Computer Vision,* pp. 301-306, 2001.

[18] D. Martinec and T. Pajdla, "Structure from Many Perspective Images with Occlusion," *Proc. European Conf. Computer Vision,* pp. 355-369, 2002.

[19] T. Morita and T. Kanade, "A Sequential Factorization Method for Recovering Shape and Motion from Image Streams," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 8, pp. 858-867, Aug. 1997.

[20] C. Poelman and T. Kanade, "A Paraperspective Factorization Method for Shape and Motion Recovery," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 3, pp. 206-219, Mar. 1997.

[21] C. Rother and S. Carlsson, "Linear Multi View Reconstruction with Missing Data," *Proc. European Conf. Computer Vision,* 2002.

[22] B.M. Sarwar et al., "Application of Dimensionality Reduction in Recommender System-A Case Study," *Proc. ACM WebKDD 2000 Web Mining for E-Commerce Workshop,* pp. 309-324, 2000.

[23] H. Shum, K. Ikeuchi, and R. Reddy, "Principal Component Analysis with Missing Data and Its Applications to Polyhedral Object Modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 9, pp. 854-867, Sept. 1995.

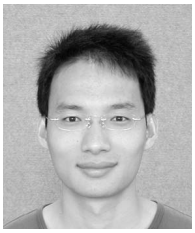
[24] P. Sturm and B. Triggs, "A Factorization Based Algorithm for Multi-Image Projective Structure and Motion," *Proc. European Conf. Computer Vision,* pp. 709-720, 1996.

[25] J.I. Thomas and J. Oliensis, "Dealing with Noise in Multiframe Structure from Motion," *Computer Vision and Image Understanding,* vol. 76, no. 2, pp. 109-124, 1999.

[26] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *Int'l J. Computer Vision,* vol. 9, no. 2, pp. 137-154, 1992.

[27] O. Troyanskaya et al., "Missing Value Estimation Methods for DNA Microarrays," *Bioinformatics,* vol. 17, pp. 520-525, 2001.

[28] J.H. Wilkinson, *The Algebraic Eigenvalue Problem.* Oxford: Clarendon Press, 1965.



Pei Chen received the BSc degree in mechanical engineering from the Jingdezhen Institute of Ceramic (1995), the masters degree in electrical engineering from North China University of Technology (1998), and the PhD degree on wavelet from Shanghai Jiaotong University (2001). Now, he is pursuing his second PhD degree in computer vision at Monash University. His main research interest includes subspace analysis in computer vision and wavelet application in image processing. He is a reviewer for *IEEE Transactions on Signal Processing*.

David Suter received the BSc degree in applied maths and physics from The Flinders University of South Australia (1977) and the PhD degree in computer vision from La Trobe University (1991). His main research interest is motion estimation from images and visual reconstruction. He is an associate professor in the Department of Electrical and Computer Systems Engineering, Monash University, Australia. He served as general cochair of the 2002 Asian Conference in Computer Vision and has been cochair of the Statistical Methods in Computer Vision workshops (2002 Copenhagen and 2004 Prague). He currently serves on the editorial board of two international journals: the *International Journal of Computer Vision* and the *International Journal of Image and Graphics*. He is a senior member of the IEEE and vice president of the Australian Pattern Recognition Society.