Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-25-2003

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

P. Chen and D. Suter

MONASH
UNIVERSITY

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

Pei Chen and David Suter
Dept ECSE, P. O. Box 35, Monash University, Australia, 3800
{pei.chen, d.suter}@eng.monash.edu.au

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, on-line "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) [15, 16, 19, 25], optical flow estimation in multi-frame video [11, 12], subspace constraints in face recognition and indexing, pose determination, data mining and a plethora of related problems (e.g., customer modelling and recommender systems [3, 21]).

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 **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 RS for real rank-r matrices **R** of size $m \times r$ and **S** of size $r \times n$, and visa versa. For the SFM problem, we are of course interested in *particular* factors (the factorization is not unique because for any invertible matrix **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 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.

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 to its low-rank approximation $\mathbf{M}^r$ minimising mean squared error (using Frobenius norm):

$$\left\| \mathbf{M} - \mathbf{M}^r \right\|_F^2 \qquad (1)$$

The singular value decomposition (SVD) gives the best solution to this problem [8]: $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, $\mathbf{M}^r = \mathbf{U}\mathbf{D}^r\mathbf{V}^T$ 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 **R** of size $m \times r$ and **S** of size $r \times n$, that minimize

$$\left\|\mathbf{M} - \mathbf{R}\mathbf{S}\right\|_F^2 \qquad (2)$$

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

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 **M** (coordinates of features tracked through frames), the object-camera motion matrix **R**, and the structure/shape matrix **S**. In the non-degenerate 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 centre of mass of the image points from their coordinates, which should be of rank 3 [15, 16, 19, 25], and even one can 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 [13, 14, 20] 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 [25] where the filling in is called "*hallucination*". In their somewhat heuristic approach to the missing data sub-problem, 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 (equation 2) $\|M - RS - C\|_F^2$ where the inclusion of C makes the adjustment for the registration.

b. Jacobs' method [13, 14] 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 [22]. Though this was not originally formulated for SFM (see section 1.2) Jacobs [13, 14] suggested that it can 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 equation 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 [25] has the following disadvantages: needing to start from a complete submatrix (it is a NP-hard problem of finding the largest complete submatrix), asymmetric usage of the data, and error propagation, as pointed out by Jacobs [13, 14].

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 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 problems, 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 [13, 14] with the projective factorization method of Sturm & Triggs [23], Martinec et al. [17] solved the missing-data problem under the perspective model. Various geometric constraints [4, 10, 15], have also been employed to cope with the missing-data problem. For example, Heyden and Kahl [10, 15] proposed to use "closure constraints" for affine construction, where the missing-data problem can be naturally handled. They noted that Jacobs' method can be regarded to be "dual" to the closure constraints. 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 [26], or in recommender system [3, 21]. Yet these studies share the same intrinsic nature: missing-data problem under low-rank constraints.

The approach used in DNA prediction [26], employs "SVDimpute" algorithms 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 re-iterated 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 data are missing [3, 21]. Secondly [26], only one missing component is updated at a time – an inefficiency. More importantly, as will be covered in the Appendix, such a strategy doesn't impute with minimal distance to the "current" subspace. Thus convergence cannot be promised. Indeed, the same criticisms as have been levelled at Tomasi and Kanade: strong dependence on the starting matrix and the imputation order [2, 3], apply. In addition, the iterative imputation

method has the possibility of exhibiting “bad behaviour” (see Appendix), i.e. the estimate goes further from the underlying optimal solution as the iteration proceeds. However, such an important issue was overlooked in [26].

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 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 re-orders 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 cols) 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 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, 21, 26], 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, $\hat{\mathbf{M}}^r$:

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

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

Note: The minimization objective is different from that in Shum's approach [22], where the objective is to *recover the matrix factors* that minimize *the re-projection error of the "non-missing" 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:

$$\min \quad \| \mathbf{M} - \hat{\mathbf{R}}\hat{\mathbf{S}} \|^2_{F\_\text{non-missing}} \qquad (4)$$

In essence, (4) predisposes one to directly seek the factors, and to perform imputation and de-noising 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 Non-linearity of the problem

Obviously, Shum's formulation (equation 4) is non-linear: in fact it is bi-linear in the factors **R** and **S**. Here, we show the intrinsic non-linearity of our formulation (equation 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 $\left\| \mathbf{M} - \mathbf{M}^r \right\|_F = \sum_{i=r+1}^{p} \sigma_i^2$ [8], where $p = \min(m,n)$ and $\{\sigma_i^2\}$ are the non-descending eigenvalues of $\mathbf{M}^T\mathbf{M}$.

Suppose $\mathbf{M}$ has some missing entries $\{M_{i,j} \mid (i,j) \in \Xi\}$, where $\Xi = \{(i,j) \mid M_{i,j} \text{ is unknown}, 1 \le i \le m, 1 \le j \le n\}$. $\mathbf{E}_{i,j} \in R^{m,n}$, has all zero entries, except a one at (i,j). Let the recovered matrix be $\hat{\mathbf{M}}$, $\hat{\mathbf{M}} = \overline{\mathbf{M}} + \sum_{(i,j)\in\Xi} k_{i,j}\mathbf{E}_{i,j}$, where $\overline{M}_{i,j} = \begin{cases} M_{i,j} & (i,j) \notin \Xi \\ 0 & (i,j) \in \Xi \end{cases}$. 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 non-negative roots for any $\{k_{i,j}\}$, because $\hat{\mathbf{M}}^T\hat{\mathbf{M}}$ is positive semi-definite. 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 6 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 approach, 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 row or column. 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 ($r \le m,n$) if it were noise-free; and another vector $\mathbf{x} \in R^m$, with missing components. Ideally, [**M**, **x**] should be also of rank r if both of them are noise-free and complete. Suppose the first k ($k \le m-r$) components of **x** (i.e., $\mathbf{x}_{1:k}$) are missing (swapping rows if necessary). The imputation method finds a linear combination of column vectors in **M**, fitting **x** the best [2, 26]:

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

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 **U** and $\mathbf{U}_2$ is the rest of **U**.

Intuitively: $\hat{\mathbf{x}}$ is the closest point to the subspace *Span*(**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*(**U**).

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

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

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

QED

Note: Although the solution by (5) is optimal in terms of the distance between the vector with missing data and the known subspace, it isn't 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 subsection, 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

***(i) Starting from a complete submatrix*:** Suppose, w.l.o.g., 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 towards the top. We don't need the largest submatrix – any $\mathbf{A}$ of size $2r \times 2r$ or larger will do.

***(ii) Initialization*:** *(a) Column-wise filling*. First consider the submatrix $[\mathbf{A} \quad \mathbf{B}]$. Recover $\hat{\mathbf{B}}$ from $\mathbf{A}$ by equation 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 sub-step (b), $\begin{bmatrix} \mathbf{A} & \hat{\mathbf{B}}_1 \\ \hat{\mathbf{C}}_1 & \hat{\mathbf{D}}_{11} \end{bmatrix}$ is now **A,** the complete submatrix, in the block representation of $\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$; $\begin{bmatrix} \mathbf{B}_2 \\ \mathbf{D}_{12} \end{bmatrix}$, $\begin{bmatrix} \mathbf{C}_2 & \mathbf{D}_{21} \end{bmatrix}$ and $\mathbf{D}_{22}$ now are **B**, **C** and **D** respectively.

After sub-step (a), check whether all the missing entries have been recovered. If so, terminate the initialisation step and go to the iteration step; if not, go to sub-step (b). After sub-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 non-recovered entries before sub-step (a) the same number as after sub-step (b)? If so, the missing entries in $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ cannot be recovered. If the number of non-recovered entries decreases, continue the initialisation step (a) by regarding the recovered entries as "non-missing".

After this initialisation procedure, we obtain a recovered matrix $\hat{\mathbf{M}}_1$, and we prepare for the iterative stage by setting $d_0 = \infty$.

***(iii) Iteration*:** 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 $d_i = \left\| \hat{\mathbf{M}}_i^r - \hat{\mathbf{M}}_i \right\|_F$. 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 hasn't 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 **m** is an arbitrary column of **M**, and its estimates are $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{m}}_{i+1}$ at the i^{th} and the $i+1^{th}$ iteration step, respectively.

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

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

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

$$\| \hat{\mathbf{M}}_{i+1} - \hat{\mathbf{M}}_i \|_F < \varepsilon' \qquad (7)$$

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

4 SVD's denoising capacity vs. missing data

Vectors, with only a few "non-missing" 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*, we experimentally find that these recovered vectors might 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 Appendix), however the true solution will be found in a closer analysis of the de-noising process. By analyzing the SVD's

* 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.

denoising capacity [5], we present a criterion to decide whether it is worth incorporating an incomplete vector into the iteration.

4.1 SVD's denoising capacity and its extension to an incomplete matrix

In [5], with the tool of the matrix perturbation theory [27], 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 **B,** which is directly observed. Then, the error that still resides in the rank-r approximation matrix, $\mathbf{B}^r$, is

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

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

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

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. Figure 1 shows the SVD's denoising capacity.

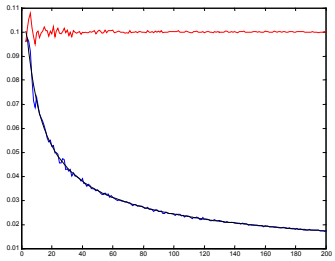


Figure 1: SVD's denoising capacity. The abscissa is the size of the square matrix, and the ordinate is the error in the rank-4 approximation matrix. Three curves are drawn: the upper one is the RMS error in the noise-corrupted matrix, the smooth dashed one is the expectation of the RMS error in the rank-4 approximation matrix, and the other non-smooth 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.

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 "non-missing" components, might degrade the accuracy obtainable from the other reliable data. Yet, using only a small complete submatrix may not achieve optimal de-noising ability – clearly there is a trade-off here. This is illustrated in Figure 2: as the missing percentage increases, the performance deteriorates.

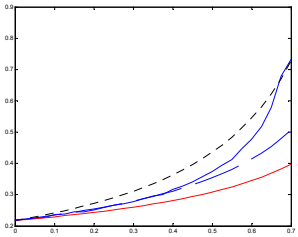


Figure 2: The optimal performance of the iterative algorithm. The abscissa is the missing percentage, ρ ; and the ordinate is RMS error for the iterative algorithm. Four curves are drawn: the upper dotted one is $(1-\rho)^{-1}$, the lower solid one $(1-\rho)^{-1/2}$, the dashed one in the middle is $(1-\rho)^{-0.7}$,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

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 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

[*] Please note that, in Shum's formulation [22], the mean is also considered, so in that case there are $(m+n-r)r+n$ independent variables.

the low-rank matrix [22], and the latter is the number of the variables in the matrix. 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 ($p \geq (m+n-r)r$) "non-missing" components in a matrix **B**, and each row (column) has at least r "non-missing" components. The best estimate of **B**, $\hat{\mathbf{B}}$, should have the following property:

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

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 missing percentage. The RMS error index of the iterative algorithm approximately follows $(1-\rho)^{-0.7}$ * (see Figure 2), when the percentage is less than 0.5 - not exact agreement but still useful.

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 "non-missing" components, define its ***unreliability*** as the ratio between the number of its independent variables and the number of non-missing components:

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

Thus, we propose to use the following strategy: first, using the iterative algorithm in section 3.2, to recover the most ***reliable*** incomplete sub-matrix, which has the minimal unreliability ratio; then, project other columns (rows) on it, if required, using the imputation

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

method. Specifically for SFM, our strategy is: first reconstruct the 3D scene and the cameras by the factorizing *the most reliable measurement matrix* (algorithm in section 4.3); then to estimate the positions of *other* feature points and *other* camera matrices, using the techniques in [25].

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 "non-missing" components among the retained submatrix, until the unreliability ratio beings 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 w.l.o.g. 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 towards 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 proved. Without loss of generality, suppose $n>>m>>r$, and the non-missing number in the i^{th} column, k_i, is descending, i.e., $k_i > k_{i+1}$ [*] for $1 \le i < n$. The unreliability ratio of the submatrix $\mathbf{M}_l$ (the left l columns of **M**), is:

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

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 proved: $c_l > c_{l+1} \Leftrightarrow c_l > \frac{r}{k_{l+1}} \Rightarrow c_l > \frac{r}{k_l} \Leftrightarrow c_{l-1} > c_l$.

Please note c_l, r, k_l are positive numbers. The second fact can be similarly proved.

[*] If $k_i = k_{i+1}$, both of the two columns would be pruned off or retained in the cutting process.

4.3 Algorithm (*IterPart*)

1 Use quick cull of cols(rows) that are not reasonable to iteratively impute (section 4.2).

2 Use the "sweeping" initialisation of the core algorithm (section 3.2) – generally avoiding the bootsrapping (Appendix), as it appears to no longer be necessary.

3 Use error norm monitored iteration of the core algorithm to convergence.

4 Finally, recover the "hopeless", 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.

This approach tends to converge more often then other methods.

4.4 Discussion

IterPart (section 4.3) has almost the same performance 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 is a lot of missing components, ***IterPart*** should perform better than the ***Iter***. Generally, each column (row) in the most reliable submatrix has more than $2r$ non-missing 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 $(m+n-r)r$ non-missing 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 non-missing 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 employing the "bootstrapping" strategy in the Appendix. ***IterPart*** generally, does not have such problems, as will be proved by experiments.

5 Experiments

In this section, we compare the performance of eight approaches: the proposed approach in section 3.2, ***Iter***; its variant, ***IterPart***, proposed in section 4.3; Jacobs's three methods: ***rankr***, ***rankrsfm***, and ***rankrsfm_tpose***; Shum's method, also with three variants: ***Shum+Jacobs1***, ***Shum+Jacobs2***, ***Shum+Jacobs3***, respectively starting from Jacobs' methods above. We use rank4 versions of Jacobs routines, sidestepping the erroneous centroid subtraction in the presence of missing data [10, 15] . We present 3 groups of experiments, one using synthetic data, another from the *box* sequence, which was also used by Jacobs [13, 14], and the other from the *dinosaur* sequence, which is a somewhat challenging.

We focus on stability since ***Iter***, ***Shum+Jacobs1***, ***Shum+Jacobs2***, and ***Shum+Jacobs3*** 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* non-missing components and *r+1 at least*. Indeed, no divergence case has been found in all 20000 cases we examined (20-noise-level×10-level-of-missing-percentage×100-times repetition).

5.1 Synthetic data in a 8-frame-and-40-point sequence

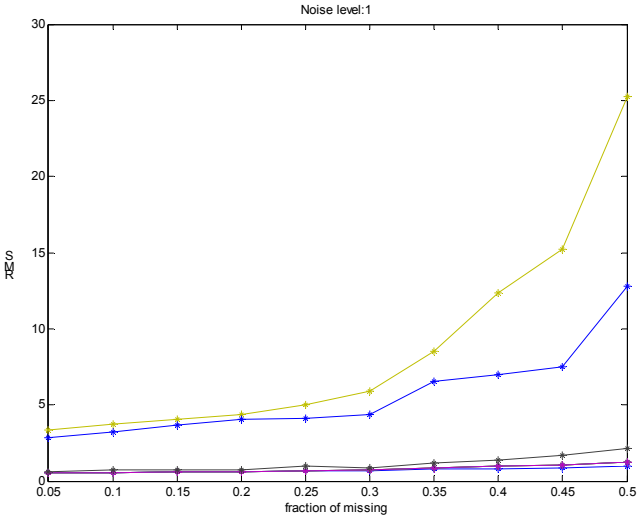
As in [10, 15], all the synthetic image data is generated this way: the 3D feature points are uniformly distribute in a cube, within [–500,500]*[–500,500]*[–500,500] units; the cameras are placed around 1000 units far away from the origin. Thus, the 2D image size is about 500*500. Then, different levels of Gaussian noise, from 1 to 20, are added into the 2D feature

points. Because the proposed algorithm has to start from a complete sub-matrix, we suppose that the first 8×8 sub-matrix is always non-missing and the missing entries randomly distribute in the other part of the matrix. In addition, in order to have a recoverable incomplete matrix, we make sure that each row/column of the incomplete has 4 non-missing entries at least. The simulation repeats 100 times for each setting.

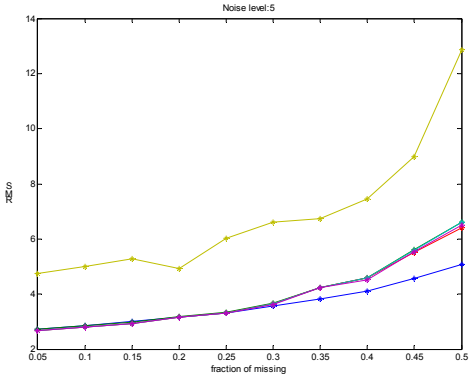
The experimental results under noise level of 1, 5, 10, 15 and 20, are shown in Figure 3. Please note, we don't include those ***divergent*** cases*, for the approaches of ***Iter***, ***Shum+Jacobs1***, ***shum+Jacobs2***, and ***shum+Jacobs3***; because the divergent cases would require a greatly expanded RMS axis. Figure 4 depicts the convergence rate for the iterative algorithms. Since the convergence rate is 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 Figure 3, that the proposed iterative algorithm (***Iter***) has almost the same performance as Shum's, and that these four curves (***Iter*** and 3 version 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,* Figure 4 shows that ***Iter*** generally converges at least as reliably. Note, the improved algorithm, ***IterPart*** converges 100% of the experiments.

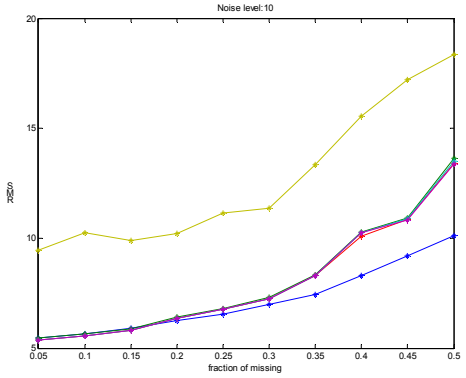
* If the RMS of any iterative algorithms has a magnitude of 3 times or more than the noise level, the algorithm is regarded divergent.



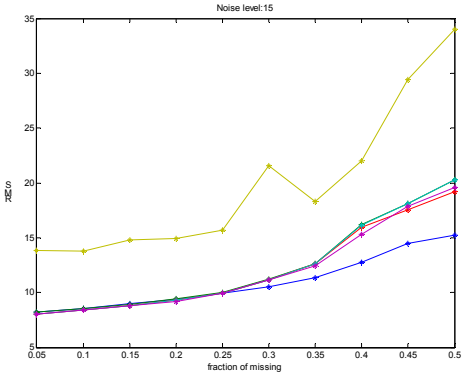
(a)



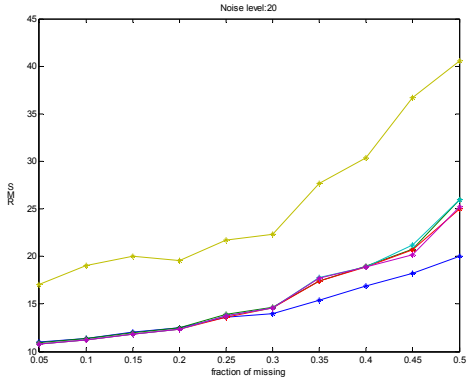
(b)



(c)



(d)



(e)

*Figure 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-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.*

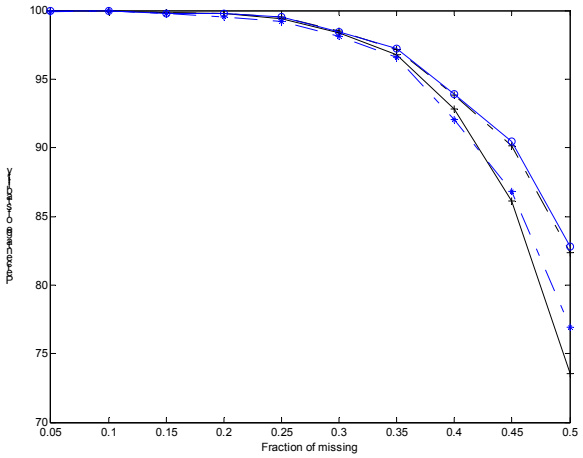


*Figure 4:The e convergence rate of four iterative methods against the missing entry fraction. Dotted curve with plus (+): **Iter**; solid curve with circle: Shum+rankrsfm; dotted curve with star (*): Shum+rankrsfm_tpose; and solid curve with plus (+):Shum+rankr.*

5.2 Box sequence

Here, to test the algorithms on real data, we use the box video, which was used in [13, 14]. The sequence consists of 40 feature points across 8 frames. One frame is shown in Figure 5. As in section 5.1, we suppose that 8 points in 4 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 Figure 6, the five methods have almost the same performance: ***Iter***, ***IterPart***, ***Shum+Jacobs1***, ***Shum+Jacobs2***, and ***Shum+Jacobs3***.

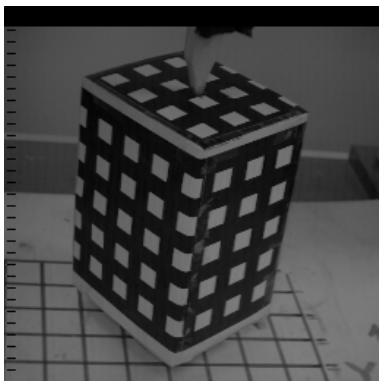


Figure 5: One frame of the box sequence.

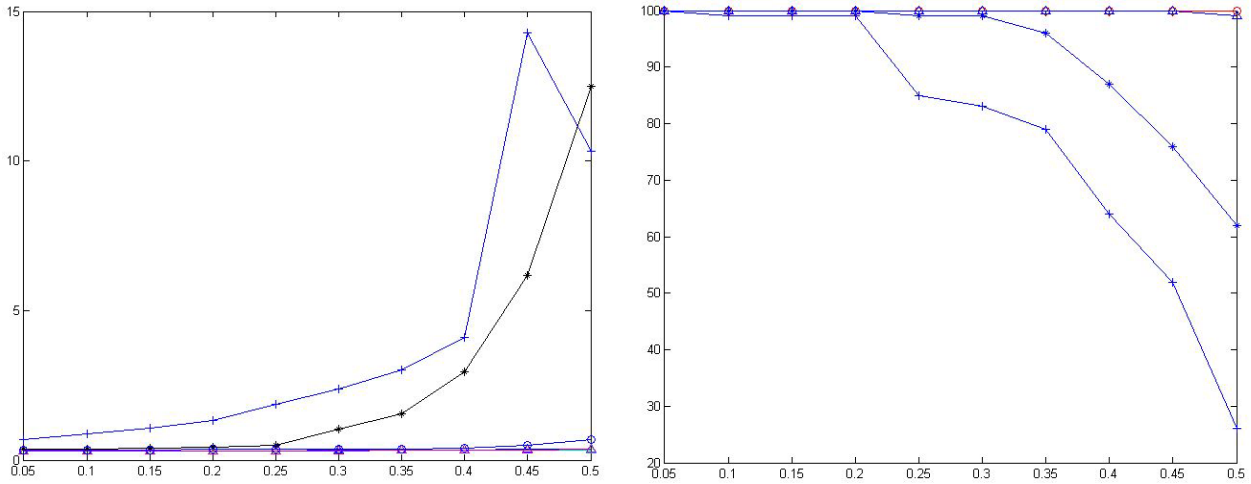


*Figure 6: The performance on the box sequence: (left) the RMS reprojection error of the eight methods are depicted: triangle (*Δ*) for five approaches (**Iter**, **IterPart**, and 3 Shum approaches), circle* (o) *for rankrsfm_tpose, star* (*) *for rankr and cross*(+) *for rankrsfm. Note: Five approaches (**Iter**, **IterPart**, and 3 Shum approaches) have almost the same performance so those five curves merged into one curve at the bottom. (right) The convergence rate of the four iterative methods are depicted: circle* (o) *for **Iter**, triangle (*Δ*) for Shum+rankrsfm_tpose, star* (*) *for Shum+ rankr and cross*(+) *for Shum+ rankrsfm.*

5.3 Dinosaur sequence

Here, we present an example where some data is truly missing (i.e., not artificially occluded to simulate missing data). 4983 feature points were tracked over the 36-frame "dinosaur" sequence [7], and the 20^{th} frame is shown in Figure 7, where the feature points are denoted by symbol "+". The feature points, extracted by the Harris interest operator [9], were obtained from Oxford [*]. Over the dinosaur sequence, about 90.84% data is missing; and the mask of the tracked feature points is shown in Figure 8, where a black pixel in (i, j) means the i^{th} feature point (in abscissa) is tracked in the j^{th} frame (in ordinate) and a grey pixel denotes the occlusion/missing data. Under the assumption of the affine camera, the measure matrix should lie in a four-dimension subspace. However, in this example, the perspectivity factor isn't negligible, and the four-dimension subspace doesn't fit 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

[*] http://www.robots.ox.ac.uk/~vgg/data/

used, by Martinec [17], 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 determine from gross statistics, as Martinec's results [17].

Figure 7: The 20th frame of the dinosaur sequence

Figure 8: The missing data (grey) and measured data (black) for the dinosaur sequence.

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 non-missing 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 6 (>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 Figure 9, where about 77% data is missing. The result by Jacobs' method under affine camera, as shown in Figure 10 (a), is unsatisfactory[*]. When the initial result is not accurate enough, Shum's approach tends to diverge, or become trapped in a local

[*] In Jacobs' software package (http://www.cs.umd.edu/~djacobs/), there are three routines for miss-data problems: one is for the generic problems, the other two are for SFM problems (one works on the measurement matrix and the other on its transpose). Here, we present the best result among these three routines.

minimum, as shown in Figure 10 (b-c). The recovered tracks by the proposed method ***Iter*** are shown in Figure 10 (d). (Which is, of course, the same as that by ***IterPart*** since we have pruned.)

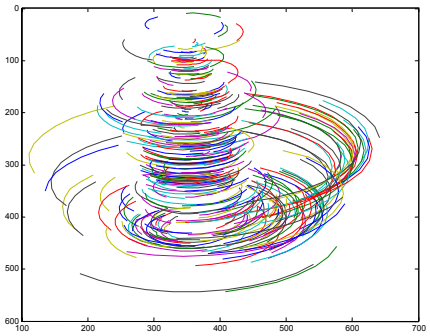
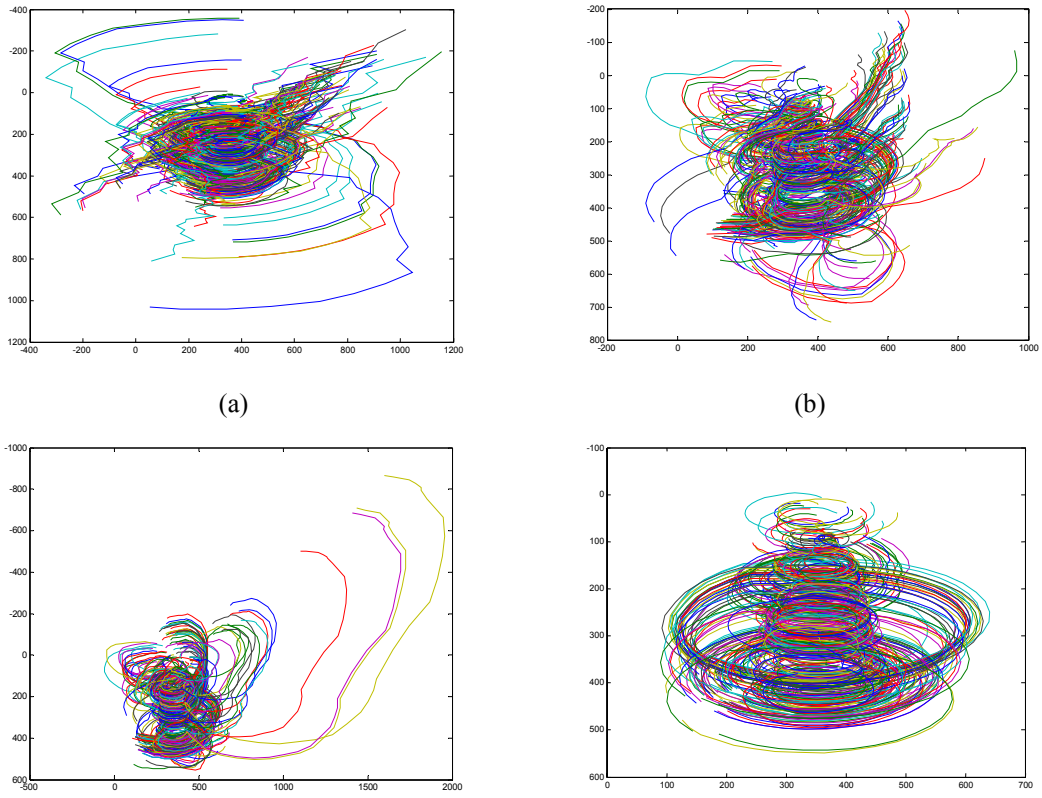


Figure 9: The 336 tracked feature points over 36 frames



(a) (b)

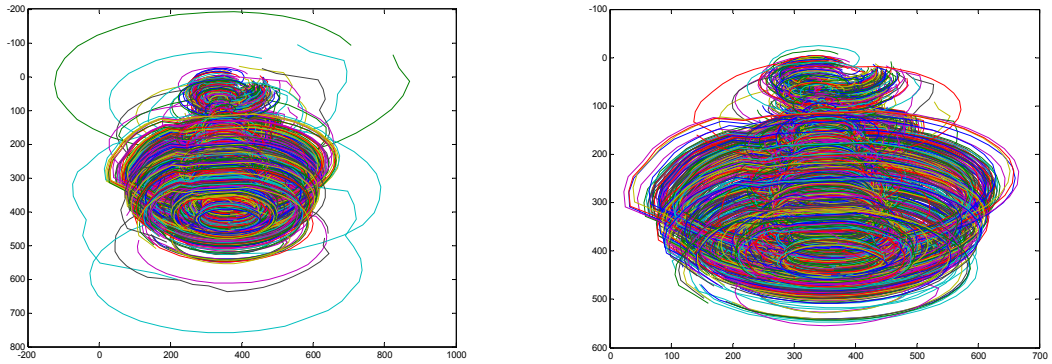
(c) (d)

*Figure 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, (d) The 336 recovered tracks by **Iter**.*

By combining Jacobs' method [13, 14] and Sturm and Triggs' projective factorization method [23], a good result over the whole sequence was reported [17]: the mean reprojection error per image point, measured by pixels, was reported as 1.76, 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: a) the whole 4983 feature points, and b) with only the 2683 feature points that were tracked over more than 2 frames. (2300 feature points were tracked only over 2 frames in the dinosaur sequence!) Our results of the reprojection tracks, for 4983 and 2683 feature points respectively, are shown in Figure 11 (a-b). Obviously, the result from 2683 features is much better than that from 4983 features. The recovered tracks should be approximately elliptical, because the sequence was taken while the dinosaur was on a rotating turn-table [7]. Note: all the wild recovered tracks in the first experiment are from the 2300 feature points which have been tracked over only 2 frames – thus the likely reason for such sensitive behaviour in Figure 11 (a) is that some feature points are tracked only over 2 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 4983 and 2683 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 2683 features scores worse although it has no wild recovered tracks. In fact, the mean/maximal error for the 2300 feature points tracked over only 2 frames is only 0.4088/ 7.8093 pixels. Since we only have the measures, as

reported by Martinec [17], it is not clear whether his results may have included such wild (and wrong) recovered tracks.



(a) (b)

Figure 11: The recovered tracks over 36 frames, (a) for 4983 points, and (b) for 2683 points.

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 proved by the experiments), one can expect to arrive at the same solution even when starting from different complete sub-matrices.

As a result of our work, we also draw to the attention of the reader a salutary message regarding using simple error measures to make 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.

Appendices

Appendix A

A.1 "Bad-behaviour" and a bootstrapping strategy

As noted above, for the proposed core iterative imputation method (***Iter***) only the convergence to a local minimum is proved. The worse case scenario is that, some components "wander away" from the underlying ground truth as the iteration proceeeds. We call this phenomenon "bad behaviour". Some vectors have polluted the first r components and the remaining data can't "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 proved by the matrix perturbation theory [27], 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 initialisation phase or in the iteration phase. We can first to regress these bad (one or two) columns (or rows) against the other columns (or rows), then to continue the iteration. Another is to restart the algorithm: in the initialization step, using those columns (or rows) that don't produce such bad behaviour producing a partially complete matrix; then, to regress the columns (rows) with bad behaviour against the partially complete matrix before re-starting the iteration. The second strategy, experimentally, performs better than the first.

The "afterward" bootstrapping strategy isn't ideal because of its time consuming. Generally, the wandering-away behaviour occurs with those columns, with only r (or slightly more than r) "non-missing 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 behaviour occurs with these vectors.

Thus, we propose the following bootstrapping strategy[*] 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 suggest that only those columns (rows), with more than (or equal to) $2r$ non-missing data, should be incorporated into the complete submatrix, by using the imputation method (equation 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) $2r$ "non-missing" values. In such cases, one can temporarily relax the constraint of requiring $2r$ "non-missing values" – using

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

columns (rows) with *2r-1* "non-missing 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* "non-missing 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 [22]. 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 will also be proved by experiments that almost the same solution is obtained by our method in section 3.2 and Shum's method, *providing* both of them converge. In practice our approach converges far more reliably.

A.3 Difference from the imputation in [26]

As noted in the introduction, the iterative imputation method in [26] can not be shown to converge, although the iteration may stop after a few loops. The problem with the method in [26] lies in its updating procedure in the iteration. In [26], 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 [26] doesn't 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 [26].

A.4 RMS and Re-projection 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.

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 Figure 12, we can observe this point: a series of synthetic measurement matrices ($\widetilde{\mathbf{M}}$) are generated and i.i.d. Gaussian noise (0-mean-and-1-variance) is added, observing **M**. The reprojection error, compared with $\widetilde{\mathbf{M}}$ and **M,** is depicted by the dashed curve and the solid curve, respectively. We also compare the rank-*4* approximation of **M**, $\mathbf{M}^4$, with $\widetilde{\mathbf{M}}$ and **M**, the error is depicted by the dot-with-star curve and the dotted curve, respectively. Obviously, the RMS indexes, against **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 [18, 24]. In contrast, the lower two traces (using "ground truth") show the correct trend.

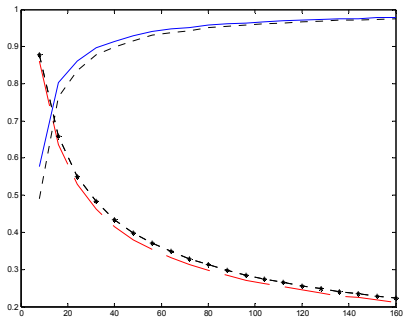


Figure 12: The RMS errors in the low-rank approximation matrix and the reprojection RMS error: the abscissa is the size of the square measurement matrix and the ordinate is the RMS error. The dotted with star curve and the dotted one is the RMS errors of the rank-4 approximation matrix, compared with 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

.

Thus, 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 non-missing 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% data missing and a noise level of 10, where the RMS error for the non-missing 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.

Acknowledgements: Thanks to 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 PAMI, vol. **25**, no. 9, pp. 1134-1149, 2003.

[2] M. Brand. *Incremental Singular Value Decomposition of Uncertain Data with Missing Values*. in *ECCV*, pp. 707-720, 2002.

[3] M. Brand. *Fast online SVD revisions for lightweight recommender systems*. in *SIAM 3rd International Conference on Data Mining*, pp., 2003.

[4] S. Brandt. *Closed-Form Solutions for Affine Reconstruction under Missing Data*. in *Statistical Methods for Video Processing Workshop, in conjunction with ECCV02*, 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 University, Clayton 3800, Australia, 2003. http://www.ds.eng.monash.edu.au/techrep/reports/2003/MECSE-6-2003.pdf

[6] F. De la Torre and M.J. Black, *A framework for robust subspace learning.* IJCV, vol. **54**, no., 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*. in *LNCS 1506*, pp. 155-170, 1998.

[8] G.H. Golub and C.F.V. Loan, *Matrix Computations,*. 2nd ed. 1989, Baltimore: Johns Hopkins University Press.

[9] C.J. Harris and M. Stephens. *A combined corner and edge detector*. in *Proc. Alvey Vision Conference*, pp. 147-151, 1988.

[10] A. Heyden and F. Kahl. *Reconstruction from Affine Cameras using Closure Constraints*. in *ICPR*, pp. 47-50, 1998.

[11] M. Irani. *Multi-frame optical flow estimation using subspace constraints*. in *ICCV*, pp. 626-633, 1999.

[12] M. Irani, *Multi-frame correspondence estimation using subspace constraints.* IJCV, vol. **48**, no. 3, pp. 173-194, 2002.

[13] D. Jacobs. *Linear fitting with missing data: Applications to structure-from-motion and to characterizing intensity images*. in *CVPR*, pp. 206-212, 1997.

[14] D. Jacobs, *Linear fitting with missing data for structure-from-motion.* CVIU, vol. **82**, no., pp. 57-81, 2001.

[15] F. Kahl and A. Heyden, *Affine structure and motion from points, lines and conics.* IJCV, vol. **33**, no. 3, pp. 163-180, 1999.

[16] K. Kanatani. *Motion segmentation by subspace separation and model selection*. in *ICCV*, pp. 301-306, 2001.

[17] D. Martinec and T. Pajdla. *Structure from many perspective images with occlusion*. in *ECCV*, pp. 355-369, 2002.

[18] T. Morita and T. Kanade, *A sequential factorization method for recovering shape and motion from image streams.* IEEE PAMI, vol. **19**, no. 8, pp. 858-867, 1997.

[19] C. Poelman and T. Kanade, *A paraperspective factorization method for shape and motion recovery.* IEEE PAMI, vol. **19**, no. 3, pp. 206-219, 1997.

[20] C. Rother and S. Carlsson. *Linear multi view reconstruction with missing data*. in *ECCV*, pp. 309-324, 2002.

[21] B.M. Sarwar*, et al. Application of dimensionality reduction in recommender system -- A case study*. in *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, pp., 2000.

[22] H. Shum, K. Ikeuchi, and R. Reddy, *Principal component analysis with missing data and its applications to polyhedral object modeling.* IEEE PAMI, vol. **17**, no. 9, pp. 854-867, 1995.

[23] P. Sturm and B. Triggs. *A factorization based algorithm for multi-image projective structure and motion*. in *ECCV*, pp. 709-720, 1996.

[24] J.I. Thomas and J. Oliensis, *Dealing with noise in multiframe structure from motion.* CVIU, vol. **76**, no. 2, pp. 109-124, 1999.

[25] C. Tomasi and T. Kanade, *Shape and motion from image streams under orthography: A factorization method.* IJCV, vol. **9**, no. 2, pp. 137-154, 1992.

[26] O. Troyanskaya*, et al.*, *Missing value estimation methods for DNA microarrays.* Bioinformatics, vol. **17**, no., pp. 520-525, 2001.

[27] J.H. Wilkinson, *The algebraic eigenvalue problem*. 1965, Oxford: Clarendon Press.