PowerFactorization : 3D reconstruction with missing or uncertain data

Richard Hartley and Frederik Schaffalitzky
Research School of Information Sciences,
Australian National University,
ACT 0200, AUSTRALIA

Abstract

Many problems in computer vision may be considered as low-rank approximation problems, in which a matrix of measured data must be approximated by a matrix of given low rank. If the matrix has no missing entries, then this is easily accomplished by a Singular Value Decomposition (SVD). If some measurements are missing however, and the matrix has holes, then the SVD method can not be applied. We present here a practical iterative method for approximating a data matrix, possibly with missing entries, with another matrix of small rank r. For a complete data matrix the method reduces to the well-known "Power Method" which is provably convergent to a unique global optimum. If the data is well approximated by a matrix of rank r the Power Method has rapid convergence. Our method for incomplete data is applied to several problems of 3D reconstruction, generalizing the Tomasi-Kanade method for orthographic cameras and the Sturm-Triggs method for projective cameras to missing and uncertain data.

1 Introduction

In the affine or orthogonal reconstruction problem, a set of points $\mathbf{X}_j$ are seen in several images, the measured coordinates of the points being denoted by $\mathbf{x}_{ij}$. The reconstruction task, given the image measurements $\mathbf{x}_{ij}$, is to find the positions of the points $\mathbf{X}_j$ and the camera matrices $\mathtt{P}_i$ of the cameras. It was shown by Tomasi and Kanade ([13]) that this problem may be reduced to a low-rank approximation problem as follows. A matrix $\mathtt{M}$ is formed, consisting of the measured point coordinates in a centred coordinate system: $\mathtt{M} = [\mathbf{x}_{ij}]$. One then finds a rank-3 approximation $\bar{\mathtt{M}}$ to $\mathtt{M}$, minimizing the Frobenius norm $||\mathtt{M} - \bar{\mathtt{M}}||$. Here $\bar{\mathtt{M}} = \mathtt{A}\mathtt{B}^\top$ for two matrices $\mathtt{A}$ and $\mathtt{B}$ both having three columns. The camera matrices $\mathtt{P}_i$ and the points $\mathbf{X}_j$ may then be read directly from $\mathtt{A}$ and $\mathtt{B}$ respectively. The rank-3 approximation $\bar{\mathtt{M}}$, and the two matrices $\mathtt{A}$ and $\mathtt{B}$ are conveniently computed by carrying out a Singular Value Decomposition (SVD) of matrix $\mathtt{M}$, and zeroing out all but the three largest singular values. The great success of this algorithm is due to its simplicity and reliability The main computational tool is the SVD, which (given a good implementation) is virtually fool-proof. Furthermore, it minimizes the correct geometric cost function.

Subsequently, low-rank approximation and factorization of measurement matrices has been applied to many problems. We name just a few such applications. The original algorithm was generalized to various other camera models ([9, 12]). Furthermore Irani and Anandan ([6]) have generalized it to the case of non-isotropic measurement covariance. Shashua ([10]) observed that the set of trifocal tensors from an image sequence lie in a low-rank subspace. Irani has made a similar observation concerning image-to-image homographies ([15]). Applications to Principal Component Analysis (PCA) are given in [11, 3]. In all these algorithms, low-rank reduction is used to mitigate noise effects and provide robust approaches to the respective estimation problems. The method used in this paper can be applied with advantage to all these problems.

The most persistent difficulty with the method of low-rank approximation is that it requires that all the points be visible in all views, or in a more abstract context, the measurement matrix $\mathtt{M}$ must have no missing entries. Various ways ([7, 13]) have been suggested for getting around this restriction. The best known is [13] in which a strategy for filling in missing data are discussed and suggested. The problem is that this data is effectively invented, and can cause the results to be biased to accommodate this fictitious data. [2] addresses this by dealing with the missing data using an EM algorithm. A recent paper [1] addresses the problem of incremental SVD with missing data by combining imputation (the filling-in of missing entries based on what has been seen so far) with a sparse reduction of the updated matrices to diagonal form. While the results are promising the method suffers from the same theoretical difficulty and the final result will depend on the order in which the data is encountered. In this paper we do not attempt to fill in missing data, but live with it (or rather, without it).

Comparison with previous work. Our paper is most closely related to work of Morita&Kanade [8] on factorization, and De la Torre&Black ([3]) and Shum et al. ([11])

on PCA. Morita and Kanade introduce orthogonal iteration as a method of incremental affine reconstruction. We observe that a small but important modification to their method yields a technique that may be applied with missing or uncertain data. In seemingly unrelated work [3, 11] a method of alternating between optimization over partial parameter sets is proposed for Principal Component Analysis. This method of alternation has been criticized as a general technique in [14] on the grounds of slow convergence. In this paper, however, we show that the techniques of [8] and [3, 11] are closely related, and that the guaranteed fast global convergence properties shown in [8] carry over to a wider range of problems, and provide theoretical and experimental justification of the process of alternation.

We use the developed technique to solve the problem of affine reconstruction with uncertain data and missing data (which is an extreme case of uncertainty). The algorithms are rapidly convergent from a random initial point and are robust to large amounts of noise. The paper of Irani and Anandan ([6]) is the most noteworthy prior paper on reconstruction with uncertainty. Their method does not give an exact solution (our method does), since they reduce the measurement matrix to rank 6 (instead of 3) and then use an inexact method for the final reduction to rank 3. In addition, they require that the covariance of a given point be the same in all images, which our method does not.

Overview of the PowerFactorization method. The standard method for finding a low-rank approximation to a matrix $\mathtt{M}$ is to use the SVD, but this is not the only method. The problem is mathematically equivalent to finding the eigenvalues and associated eigenvectors of $\mathtt{M}\mathtt{M}^\top$. The Power Method ([4]) is a useful method for finding the dominant eigenvector of a matrix $\mathtt{X}$, converging rapidly if the largest eigenvalue sufficiently dominates the next largest. Starting from a random vector $\mathbf{u}_0$ the method is to repeatedly apply $\mathtt{X}$ to $\mathbf{u}$ and normalize the result:

$$\mathbf{u}_{k+1} = \mathtt{X}\mathbf{u}_k / \|\mathtt{X}\mathbf{u}_k\|$$

The Power Method may be extended for finding the dominant subspace of dimension r (i.e. the subspace spanned by the first r eigenvectors). Applying it to a symmetric positive semi-definite matrix $\mathtt{M}\mathtt{M}^\top$, we starting with a random matrix $\mathtt{U}_0$ with r orthonormal columns and repeatedly multiply by $\mathtt{M}\mathtt{M}^\top$ and re-orthonormalize columns:

$$\mathtt{U}_k = (\mathtt{M}\mathtt{M}^\top)\mathtt{U}_{k-1}\mathtt{N}_k \tag{1}$$

where $\mathtt{N}_k$ is an upper triangular matrix that makes the columns of $\mathtt{U}_k$ orthonormal. This normalizing matrix $\mathtt{N}_k$ is found by the Gram-Schmidt process, equivalent to QR decomposition of $(\mathtt{M}\mathtt{M}^\top)\mathtt{U}_{k-1}$. The resulting algorithm is known as *orthogonal power iteration* [4].

Using this iteration for the dominant subspace of $\mathtt{X} = \mathtt{M}\mathtt{M}^\top$ turns out (see below) to be equivalent to the following iteration for factoring $\mathtt{M} = \mathtt{A}\mathtt{B}^\top$ where, starting from a random rank r factor $\mathtt{A}_0$, we define successive updates

$$\begin{aligned} \mathtt{B}_k &= (\mathtt{M}^\top\mathtt{A}_{k-1})(\mathtt{A}_{k-1}{}^\top\mathtt{A}_{k-1})^{-1} \\ \mathtt{A}_k &= (\mathtt{M}\mathtt{B}_k)(\mathtt{B}_k{}^\top\mathtt{B}_k)^{-1} \end{aligned} \tag{2}$$

Thus, the algorithm requires little more than matrix multiplication and inversion of small ($r \times r$) matrices, and will converge very rapidly if $\mathtt{M}$ is indeed close to a rank r matrix. Convergence to a *global* minimum of the cost $||\mathtt{M} - \mathtt{A}\mathtt{B}^\top||$ is guaranteed.

Matrix multiplication is not possible with missing entries, so it is not clear how this solves the missing data problem. The key observation of this paper is that each step in the iteration is *exactly equivalent* to solving a least-squares problem using normal equations: starting with a random $\mathtt{A}$ one solves alternately for $\mathtt{B}$ and $\mathtt{A}$ until convergence. The point is that this set of equations can be solved even if $\mathtt{M}$ has missing entries. There is one equation for each entry of $\mathtt{M}$, and this equation may be omitted when the entry of $\mathtt{M}$ is missing. Thus, in the case of missing data, we replace the normal equations used in the full data case by least-squares solution of a set of linear equations. The cost function minimized by this algorithm is

$$\sum_{i,j} (\mathtt{M}_{ij} - (\mathtt{A}\mathtt{B}^\top)_{ij})^2$$

where the sum is only over index pairs i, j such that $\mathtt{M}_{ij}$ is defined.

In the case of 3D affine reconstruction, the method just described consists of starting with a random set of initial points, and linearly solving alternately for the camera matrices and the 3D points until their product converges to the measurement matrix. This approach seems so naive that it is hard to see that it will be effective. However, as will be shown, in the case of full data and no noise the algorithm converges to the optimal solution in a single iteration. With added noise it may be proven to converge rapidly to the global optimum. For the case of missing data, it is possible for local minima to occur when the noise level or the percentage of missing points is high. It is possible for the algorithm to fall into a local minimum in such a case. Nevertheless for moderate amounts of missing points this problem does not seem to occur.

The advantage of the technique described in this paper is that it does not simply make local incremental improvements. Instead the cost function is optimized (globally) with respect to $\mathtt{A}$ and $\mathtt{B}$ alternately. Thus the algorithm is able to make large jumps towards the global minimum at each step.

2 Finding a rank r approximation

We consider again the affine reconstruction problem. In the affine camera model, a point represented by a non-homogeneous 3-vector $\mathbf{X}$ is mapped to an image point $\mathbf{x}$ according to $\mathbf{x} = \mathtt{A}\mathbf{X} + \mathbf{t}$, where $\mathtt{A}$ is a 2×3 matrix, and $\mathbf{t}$ is a 2-vector.

The affine reconstruction problem is to compute m affine cameras and n 3D points given only the image points $\mathbf{x}_{ij}$. In particular, we want to solve the equations $\mathbf{x}_{ij} = \mathtt{A}_i\mathbf{X}_j + \mathbf{t}_i$. For the present, we assume that the points are visible in all views, so that $\mathbf{x}_{ij}$ is defined for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

In the well known paper ([13]) it was observed that by expressing the image coordinates in each image with respect to an origin defined to the the centroid of the image points in each image, it may be assumed that each $\mathbf{t}_i$ is zero. The reconstruction problem is then to find matrices $\mathtt{A}_i$ and points $\mathbf{X}_j$ such that $\mathbf{x}_{ij} = \mathtt{A}_i\mathbf{X}_j$. This can be written as a single matrix equation

$$\begin{bmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \cdots & \mathbf{x}_{mn} \end{bmatrix} = \begin{bmatrix} \mathtt{A}_1 \\ \vdots \\ \mathtt{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \cdots & \mathbf{X}_n \end{bmatrix} .$$

This set of equations may be written as $\mathtt{M} = \mathtt{A}\mathtt{B}^\top$, where we note that $\mathtt{M}$ (the *measurement matrix*) is made up of the image coordinates of all the image points, matrix $\mathtt{A}$ is formed by stacking the individual camera matrices, and $\mathtt{B}$ consists of the coordinates of the 3D points. The affine reconstruction problem is thus exactly the problem of factoring the matrix $\mathtt{M}$ into two factors of rank 3. For more details, the reader is referred to the original paper [13] or to [5].

In the presence of noise, the exact factorization of $\mathtt{M}$ into factors of rank 3 will not be possible. Instead, one finds the closest rank 3 product $\bar{\mathtt{M}} = \mathtt{A}\mathtt{B}^\top$ (where $\mathtt{A}$ and $\mathtt{B}$ have just three columns), so as to minimize the Frobenius norm $||\mathtt{M} - \mathtt{A}\mathtt{B}^\top||$ of the difference between $\mathtt{M}$ and $\hat{\mathtt{M}}$. Assuming isotropic IID Gaussian image noise this will compute the Maximum Likelihood Estimate. The minimization can be carried out by computing a Singular Value Decomposition of $\mathtt{M}$ and truncating all but its first three singular values to zero.

2.1 The PowerFactorization Method

Let $\mathtt{M}$ be an $m \times n$ matrix and let $r < m, n$. We wish to find the best rank r approximation to $\mathtt{M}$, i.e. to compute matrix factors $\mathtt{A}$ and $\mathtt{B}$ of size $m \times r$ and $n \times r$ respectively such that $\|\mathtt{M} - \mathtt{A}\mathtt{B}^\top\|$ is minimized. As remarked above the solution to this problem is to compute the SVD of $\mathtt{M}$ and truncate all but the r largest singular values to zero.

An alternative method is the following generalization of the orthogonal power iteration method. Starting with an initial random $m \times r$ matrix $\mathtt{A}_0$, we iterate as follows.

$$\begin{array}{rcl} \mathtt{B}_k & = & \mathtt{M}^\top \mathtt{A}_{k-1} \mathtt{N}_k \\ \mathtt{A}_k & = & \mathtt{M}\mathtt{B}_k (\mathtt{B}_k{}^\top \mathtt{B}_k)^{-1} \end{array} \tag{3}$$

until the product $\mathtt{A}_k\mathtt{B}_k{}^\top$ converges. The matrix $\mathtt{N}_k$ is a non-singular $r \times r$ matrix used for normalization, important only for numerical stability of the problem. It is easily verified that the sequence of products $\mathtt{A}_k\mathtt{B}_k{}^\top$ does not depend on the normalization matrices $\mathtt{N}_k$, since all occurences of $\mathtt{N}_k$ cancel out when the product is formed.

In the case where $\mathtt{N}_k$ is chosen so that $\mathtt{B}_k$ has orthonormal columns, then $(\mathtt{B}^\top\mathtt{B})$ is the identity matrix, so the iteration becomes

$$\begin{array}{rcl} \mathtt{B}_k & = & \mathtt{M}^\top \mathtt{A}_{k-1} \mathtt{N}_k \\ \mathtt{A}_k & = & \mathtt{M}\mathtt{B}_k \end{array} \tag{4}$$

Combining the two steps gives $\mathtt{A}_k = (\mathtt{M}\mathtt{M}^\top)\mathtt{A}_{k-1}\mathtt{N}_k$, which is just the formula for orthogonal iteration given in (1) and [8]. The convergence properties of orthogonal power iteration are well known ([4, 8]). Those convergence results may be extended to the convergence of (3), as follows[1].

2.1. *Let s_j be the j-th largest singular value of $\mathtt{M}$ and suppose that $s_r > s_{r+1}$. If $\bar{\mathtt{M}}$ be the closest rank-r approximation to $\mathtt{M}$, then there exists a constant C (depending on $\mathtt{M}$ and $\mathtt{A}_0$) such that for all k*

$$||\bar{\mathtt{M}} - \mathtt{A}_k\mathtt{B}_k{}^\top|| \leq C(s_{r+1}/s_r)^{2k}$$

If the matrix $\mathtt{M}$ were actually of rank r then s_{r+1} would of course be zero and the algorithm would converge in one iteration. For matrices that are merely close to having rank r the gap between s_r and s_{r+1} will be large and so convergence will be rapid.

Actually, the method given above provides a slight extension of the algorithms in [4] and [8], in that it computes both $\mathtt{A}$ and $\mathtt{B}$. In [8] orthogonal iteration is applied to the symmetric matrix $\mathtt{M}\mathtt{M}^\top$ to compute $\mathtt{A}$. By contrast, we carry out the iteration in two steps, computing $\mathtt{A}$ and $\mathtt{B}$. The difference is crucial for various reasons. Forming the product $\mathtt{M}\mathtt{M}^\top$ explicitly is a bad idea from the point of view of conditioning ([4]), secondly proceeding in two steps in this way is more efficient in terms of operations for small rank r, and most importantly, the two-step iteration may be generalized to missing data by observing a link with least-squares problems in a way that the standard orthogonal iteration method cannot. This will be explained in the next section.

[1] Lack of space precludes inclusion of a proof of this, but the reader may see [8] or [4] for the general idea.

2.2 PowerFactorization as Least-squares

A different form of the normalization matrix $\mathtt{N}$ in (3) can be made in which the two steps are symmetric, namely

$$\begin{aligned} \mathtt{B}_k &= (\mathtt{M}^\top \mathtt{A}_{k-1})({\mathtt{A}_{k-1}}^\top \mathtt{A}_{k-1})^{-1} \\ \mathtt{A}_k &= (\mathtt{M}\mathtt{B}_k)({\mathtt{B}_k}^\top \mathtt{B}_k)^{-1} \end{aligned} \tag{5}$$

As observed above, this sequence will generate the same sequence of rank r products $\mathtt{A}_k{\mathtt{B}_k}^\top$ as the orthogonal power iteration method (3) and (4). There are two interesting things to note about this iteration.

Firstly, each $\mathtt{B}_k$ and $\mathtt{A}_k$ is nothing other than the least-squares solution to a set of equations of the form $\mathtt{M} = \mathtt{A}\mathtt{B}^\top$. At each step, we minimize the squared norm $||\mathtt{M} - \mathtt{A}\mathtt{B}^\top||$, solving first for $\mathtt{B}$ and then for $\mathtt{A}$. This least-squares property is easily seen by comparing it with the least-squares equation $\mathbf{m} = \mathtt{A}\mathbf{b}$. The "normal equation" solution to this problem is $\mathbf{b} = (\mathtt{A}^\top\mathtt{A})^{-1}\mathtt{A}^\top\mathbf{m}$, which when transposed is $\mathbf{b}^\top = (\mathbf{m}^\top\mathtt{A})(\mathtt{A}^\top\mathtt{A})^{-1}$ which looks very like (5). Applying this to each row $\mathbf{m}$ of $\mathtt{M}$ gives the desired equivalence. The equations 5 are just matrix versions of the normal equations.

Secondly, since each update involves nothing other than solving a linear least squares problem – minimize $\|\mathtt{A}\mathtt{B}^\top - \mathtt{M}\|$ over $\mathtt{B}$ given $\mathtt{A}$ or vice versa – it readily generalizes to the case where some entries of $\mathtt{M}$ are missing, even though the tool of matrix multiplication (used in (3)) can not be used in this setting. This is the PowerFactorization algorithm:

Algorithm 2.2.

1. Given $m \times n$ matrix $\mathtt{M}$, start with a random matrix factor $\mathtt{A}_0$ of dimension $m \times r$. Then, repeat the next two steps until the product $\mathtt{A}_k{\mathtt{B}_k}^\top$ converges.

2. Given $\mathtt{A}_{k-1}$, find the $n \times r$ matrix $\mathtt{B}_k$ that minimizes $\sum_{ij} |\mathtt{M}_{ij} - (\mathtt{A}_{k-1}{\mathtt{B}_k}^\top)_{ij}|^2$ where the sum is over those index pairs i, j such that the entry $\mathtt{M}_{ij}$ is available.

3. Given $\mathtt{B}_k$, find the matrix $\mathtt{A}_k$ that minimizes $\sum_{ij} |\mathtt{M}_{ij} - (\mathtt{A}_k{\mathtt{B}_k}^\top)_{ij}|^2$ where the sum is over the same index pairs as before.

Thus, we see that the algorithm adopts the seemingly naive strategy of starting at a random starting point $\mathtt{A}_0$ and iteratively solving (by linear least squares) for $\mathtt{B}_k$ and $\mathtt{A}_k$ until convergence. Despite the simple-minded nature of this algorithm, we have shown in (2.1) that it converges very rapidly to the **global** minimum solution (at least in the case of complete data).

Computationally, the updates of $\mathtt{B}$ given $\mathtt{A}$ (the update of $\mathtt{A}$ given $\mathtt{B}$ is similar) can be done by solving a least squares problem for each row of $\mathtt{B}$ independently. This is because each column of ${\mathtt{B}}^\top$ contributes only to the entries in the corresponding column of $\mathtt{M}$. Missing entries in $\mathtt{M}$ correspond to omitted equations.

Normalization. At any point in the iteration at one can choose to condition the problem by replacing and $\mathtt{B}_k$ with $\mathtt{B}_k\mathtt{N}$ where $\mathtt{N}$ is some normalization matrix. This does not affect the sequence of products $\mathtt{A}{\mathtt{B}}^\top$ but may be helpful for numerical stability. In our implementation we normalize $\mathtt{B}_k$ at each iteration.

To summarize: the PowerFactorization method is a two-step form of the orthogonal power iteration applied to $\mathtt{M}$ which produce the same sequence of rank r factorizations and therefore has the same rate of convergence. For complete data matrices $\mathtt{M}$ it provably converges to the unique local (hence global) minimum. If the data matrix is of rank r then it converges in one iteration.

Timing experiments. Tests with showed that on a 500×500 matrix PowerFactorization found a rank-4 approximation in 5% of the time of SVD. More details on timing are found in [8].

2.3 Convergence

It was shown that in the case where all the entries of $\mathtt{M}$ are known, this algorithm converges quickly to a global minimum. We use this fact to argue that the omission of a number of equations, corresponding to missing entries in $\mathtt{M}$ will not greatly effect the convergence, or indeed the ultimate solution to the set of equations. The set of equations used to solve for the $\mathtt{A}_i$ or $\mathtt{B}_i$ at each step is highly redundant. In particular with no missing data, there are either mn equations in mr or nr unknowns. Omission of a small percentage of these equations will not be expected to materially effect the cost surface. Certainly, search for minima holding either $\mathtt{A}$ or $\mathtt{B}$ fixed is still a quadratic problem for which we reach a directional minimum in one step of linear-least squares. Since the algorithm converges in the complete data case, it may be expected to converge in the case of missing data, as long as the amount of missing data is not too great. This expectation was borne out by experiments, which showed that convergence from a random viewpoint will occur reliably with large amounts of missing data (see Fig 1).

Naturally, deletion of a large fraction of the entries of $\mathtt{M}$, and hence a corresponding fraction of the equations used to solve for successive $\mathtt{A}_i$ and $\mathtt{B}_i$ will cause the solution to diverge increasingly from the estimate for the complete data case. With too few data points, the successive estimates may fail to converge, or converge very slowly, but this happens only with large amounts of missing data, and increasingly infrequently as the size of the matrix $\mathtt{M}$ grows.

If the solution evidently fails to converge, as indicated by too high a residual value, then starting again from a different random initial $\mathtt{A}_0$ will often solve the problem.

Since (even in the missing data case) the residual error $||\mathtt{M} - \mathtt{A}_i{\mathtt{B}_i}^\top||$ decreases at each iteration of the algorithm, the

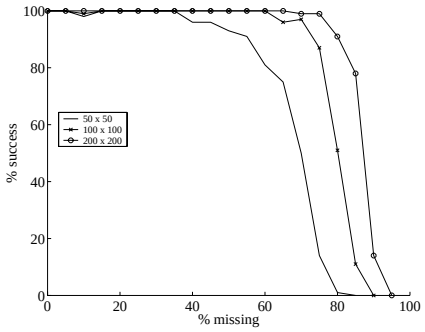

Figure 1: *Breaking point experiments on synthetic data: for three sizes of matrix and various proportions of missing entries the empirical chance of successful convergence was evaluated. As expected the risk of breakdown increases as the number of missing entries increases but the tolerance to incomplete data is still considerable. Moreover, as the size of the problem increases, the chance of success increases. In fact for a problem of size* 1000×2000 *the algorithm converged perfectly with up to 95% missing data (not shown on the graph). The noise level was 5% but the noise level seems to have little effect on the success rate.*

products $\mathtt{A}_i{\mathtt{B}_i}^\top$ must converge to a limit (or at least $||\mathtt{M} - \mathtt{A}_i{\mathtt{B}_i}^\top||$ must converge). In the case of complete data, this minimum must be a **global** minimum. In the case of missing data, however, it is possible to converge to a local minimum. An example of this is the matrix

$$\mathtt{M} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & -7 \\ -2 & 3 & \times \end{bmatrix}$$

where $\times$ indicates a missing entry. We attempted to approximate this matrix by its closest rank-1 matrix, $\bar{\mathtt{M}}$. It was determined empirically that there are two local minima with residuals $||\mathtt{M} - \bar{\mathtt{M}}||$ of 1.515 and 2.091 respectively. The algorithm falls into one or either of these minima depending on the starting point. It should be emphasized, however that this is exceptional behaviour, and does not occur very often in practice, unless the amount of missing data is too great.

Detecting successful convergence With noise, the algorithm will not of course converge to the true value of the noise-free points, but rather to the minimum of cost solution. If $\bar{\mathbf{X}}$ represents the noise-free data, $\mathbf{X}$ the noisy data and $\hat{\mathbf{X}}$ the final estimate, then the condition for convergence to a global minimum is that $\hat{\mathbf{X}} - \mathbf{X}$ is perpendicular to $\hat{\mathbf{X}} - \bar{\mathbf{X}}$ (see [5]). Note that this condition is not satisfied at a local minimum. The (somewhat stringent) condition we used to claim correct convergence is that the Pythagorean equality was satisfied within 0.1%.

Banded data problems The results given in Fig 1 were for data sets with randomly omitted points. In real image sequences, the missing data often has a banded form, since tracks are of finite length over consecutive frames. Experiments with real data showed that the PowerFactorization algorithm did not perform on such sequences as well as it does for randomly missing data. To solve this problem we adopted a bootstrapping procedure in which structure and motion estimates are built up over an expanding window of frames. This method was used with success on the real examples shown later. Details of this strategy will be in a longer version of this paper.

3 Why not bundle-adjustment

The idea of solving least-squares minimization problems by alternately carrying out minimization steps with respect to subsets of the variable parameters has been suggested many times before for different optimization problems, such as alternately solving for structure and motion. However, it has often been thought of as a sort of poor-man's bundle adjustment. The usual criticism has been that it will converge slowly because it may zig-zag slowly down narrow valleys instead of heading directly towards the minimum. This criticism is unfounded in the present case, however, since we have shown that progress towards a minimum is rapid when we solve alternately for affine structure and motion. The reason for this fast convergence is that in solving for either $\mathtt{A}$ or $\mathtt{B}$, while holding the other fixed one finds the absolute minimum in each direction in a single step.

By contrast, at each step of bundle adjustment with respect to the full set of parameters, one chooses an incremental parameter direction and does a one-dimensional search for the minimum of the cost function in that direction. This is true of methods such as conjugate-gradient, steepest descent, or Powell's method. Let $\mathtt{A}_0$ and $\mathtt{B}_0$ be some current values of the factor matrices, and let (Δ_A, Δ_B) represent an incremental search direction. The next value of the parameters will be of the form $(\mathtt{A}_0 + \lambda\Delta_A, \mathtt{B}_0 + \lambda\Delta_B)$. The cost for a given value of λ is $||\mathtt{X} - (\mathtt{A}_0 + \lambda\Delta_A)(\mathtt{B}_0 + \lambda\Delta B)^\top||^2$. It is easily seen that this is a fourth-degree polynomial in λ, which in general will not have a unique minimum. Figure 2 shows the variation of cost along some random linear cross-sections of parameter space, clearly showing double minima of the cost function. Methods such as Levenberg-Marquardt and Newton attempt to deduce the position of the minimum from local gradient information, assuming a quadratic cost function. As seen in Fig 2, the position of the minimum bears little relation to the local gradient measured far from the absolute minimum. In fact in such methods, convergence will be slow as the path to the minimum attempts to weave around local cross-sectional minima.

By contrast, if the incremental search direction is over

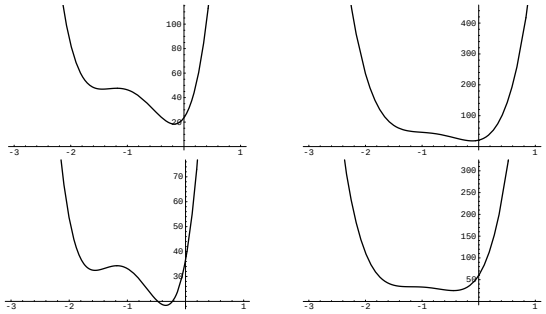

Figure 2: *Cross sectional cost-functions showing double minima, which slow convergence of full bundle-adjustment techniques. PowerFactorization avoids such local minima by alternating structure and motion optimization.*

one of the subspaces represented by $\mathtt{A}$ and $\mathtt{B}$, then the same calculation as before shows that the cost function is only a quadratic function of λ, and hence least-squares methods find the directional minimum in one step. Experiments with bundle adjustment have shown that it usually converges, but in 3 to 4 times as many iterations as PowerFactorization and as much as 100 times more slowly in terms of time, on moderate problems.

4 Affine Reconstruction

The PowerFactorization algorithm may be used to carry out affine reconstruction with full data by subtracting out the centroid of the image points in each image. This reduces the problem to one of rank-3 factorization, as discussed previously. In the case of missing data, this approach has the difficulty that it is not possible to compute the centroid of all the points in a given image, since all the image points are not known. The approach of simply using the centroid of the visible points to start with, and later filling in the missing points by reprojecting the computed points has been suggested ([13]). We prefer not to use this method, since hallucinating the missing image measurements has the potential to create a bias towards an incorrect solution. Instead, we take an approach in line with the PowerFactorization algorithm. First of all, the method will be described for the full-data case.

The affine projection mapping may be expressed in terms of a 2×4 projection matrix $\mathtt{A}$ acting on a 4-vector $\mathbf{X} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1)^\top$ representing the points. The image point (in non-homogeneous coordinates) is simply the product $\mathbf{x} = \mathtt{A}\mathbf{X}$. Note that for this to work, the last entry of the vector $\mathbf{X}$ must equal 1.

In this setting, the problem of affine reconstruction becomes one of factoring the measurement matrix $\mathtt{M} = [\mathbf{x}_{ij}]$, where the $\mathbf{x}_{ij}$ are the image points in non-homogeneous coordinates. The required factorization is $\mathtt{M} = \mathtt{A}\mathtt{B}^\top$, where each of $\mathtt{A}$ and $\mathtt{B}$ has 4 columns, subject to the additional constraint that the final column of $\mathtt{B}$ consists of all 1s.

As usual, the problem is solved by alternately finding the least-squares solution for $\mathtt{A}$ and $\mathtt{B}$ given the other. We start with $\mathtt{B}_0$, random except that the final column consists of all 1s. The estimate of $\mathtt{A}$ is no different from usual, consisting of the least-squares solution of the equations $\mathtt{M} = \mathtt{A}_i{\mathtt{B}_i}^\top$ for $\mathtt{A}_i$.

To solve for $\mathtt{B}_i$ is a little different. We write $\mathtt{A}_{i-1} = [\tilde{\mathtt{A}}_{i-1}|\mathbf{t}_{i-1}]$, and $\mathtt{B}_i = [\tilde{\mathtt{B}}_i|\mathbf{1}]$ where $\mathbf{1}$ represents a vector of 1s. The the equation $\mathtt{M} = \mathtt{A}_{i-1}{\mathtt{B}_i}^\top$ becomes $\mathtt{M} = \tilde{\mathtt{A}}_{i-1}\tilde{\mathtt{B}}_i + \mathbf{t}_{i-1}\mathbf{1}^\top$, or $\tilde{\mathtt{A}}_{i-1}\tilde{\mathtt{B}}_i = \mathtt{M} - \mathbf{t}_{i-1}\mathbf{1}^\top$, which we solve for $\tilde{\mathtt{B}}_i$.

The complete algorithm is as follows:

Algorithm 4.3.

1. Given $\mathtt{M}_{m\times n}$, initialize the $m \times r$ matrix $\mathtt{A}_0 = [\tilde{\mathtt{A}}_0|\mathbf{t}_0]$ with random entries.
2. Iteratively carry out the following steps until convergence of the product $\mathtt{A}_i{\mathtt{B}_i}^\top$.
3. Given $\mathtt{A}_{i-1} = [\tilde{\mathtt{A}}_{i-1}|\mathbf{t}_{i-1}]$, solve (one row at a time) for $\tilde{\mathtt{B}}_i$ as the least-squares solution to the set of equations $\tilde{\mathtt{A}}_{i-1}\tilde{\mathtt{B}}_i = \mathtt{M} - \mathbf{t}_{i-1}\mathbf{1}^\top$.
4. Given $\mathtt{B}_i$, solve (one row at a time) for $\mathtt{A}_i$ as the least squares solution to the set of equations $\mathtt{M} = \mathtt{A}_i{\mathtt{B}_i}^\top$.

We make some remarks here.

1. With missing data, those equations corresponding to missing entries in $\mathtt{M}$ are omitted. Naturally, it is possible to start with a random $\mathtt{B}_0$ instead of $\mathtt{A}_0$.
2. Normalization of each $\mathtt{A}_k$ or $\mathtt{B}_k$ is possible (and used in our implementation). Only the first three columns of $\mathtt{A}$ can be orthogonalized however, since the last column of $\mathtt{B}$ must remain equal to $\mathbf{1}$.

We tried the affine reconstruction method on a set of images taken from a plane. First and last images from the sequence are shown in Fig 3

5 Factorization with Uncertainty

We now suppose that each point measurement $\mathbf{x}_{ij}$ in an image comes with a specified uncertainty, represented by an inverse covariance matrix $\mathtt{C}_{ij}^{-1}$. In terms of this inverse covariance matrix, one may define a Mahalanobis distance

$$||\mathbf{x} - \mathbf{y}||^2_{\mathtt{C}_{ij}} = (\mathbf{x} - \mathbf{y})^\top \mathtt{C}_{ij}^{-1} (\mathbf{x} - \mathbf{y}) \ .$$

Note: In this section, subscripts on $\mathtt{A}_i$ and $\mathbf{X}_j$ indicate camera and point number, and not iteration number as elsewhere.

Figure 3: *Images used in the affine reconstruction experiment. The sequence consisted of 20 images, with 1415 tracks (3D points) and 18893 image points. This represents a missing data rate of 33%. The image sequence is not strictly affine, but is close enough to obtain good results. The Power-Factorization algorithm converged after 43 iterations, using the banded technique, to a residual of 0.365 pixels.*

For affine reconstruction, the estimation task becomes that of finding 2×4 affine projection matrices $\mathtt{A}_i$ and points $\mathbf{X}_j = (\mathtt{X}_j, \mathtt{Y}_j, \mathtt{Z}_j, 1)^\top$ that minimize the total Mahalanobis distance

$$\sum_{i,j} ||\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j||^2_{\mathtt{C}_{ij}} \quad .$$

This task may be carried out by a minor modification of the PowerFactorization algorithm as follows.

As with affine PowerFactorization, the two tasks of solving for the matrices $\mathtt{A}$ and $\mathtt{B}^\top$ are slightly different. However, as before, in the case where the covariance matrices are all equal to the identity the algorithm is identical with simple low-rank factorization.

Solving for the matrices $\mathtt{A}_i$. Suppose that all the points $\mathbf{X}_j = (\mathtt{X}_j, \mathtt{Y}_j, \mathtt{Z}_j, 1)^\top$ are known. Since matrix $\mathtt{C}_{ij}^{-1}$ is symmetric and positive semi-definite, we may factorize it as $\mathtt{C}_{ij}^{-1} = \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij}$, though this factorization does not need to be computed explicitly, as we shall see below. The contribution of point $\mathbf{x}_{ij}$ to the total cost is given by

$$\begin{aligned} & (\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j)^\top \mathtt{C}_{ij}^{-1} (\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j) \\ = \; & (\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j)^\top \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij} (\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j) \\ = \; & ||\mathtt{K}_{ij}(\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j)||^2_{\mathrm{E}} \end{aligned}$$

The final norm is simply the Euclidean norm of the vector. This cost is to be minimized over choices of $\mathtt{A}_i$. We may write

$$\mathtt{A}_i \mathbf{X}_j = \begin{bmatrix} \mathbf{X}_j{}^\top & \mathbf{0}^\top \\ \mathbf{0}^\top & \mathbf{X}_j{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{a}_i^1 \\ \mathbf{a}_i^2 \end{pmatrix}$$

where $\mathbf{a}_i^{j\top}$ represents the j-th row of $\mathtt{A}_i$. The matrix on the right-hand side of this expression we represent by $\bar{\mathtt{X}}_j$; it is a 2×8 matrix. In addition the vector $(\mathbf{a}_i^{1\top}, \mathbf{a}_i^{2\top})^\top$ on the right-hand side will be denoted by $\bar{\mathbf{a}}_i$. It contains the 8 entries of $\mathtt{A}_i$ to be computed. With this notation, the cost term to be minimized is

$$||\mathtt{K}_{ij}\mathbf{x}_{ij} - \mathtt{K}_{ij}\bar{\mathtt{X}}_j \bar{\mathbf{a}}_i||^2_{\mathrm{E}}$$

This is the cost minimized by a linear least-squares problem $(\mathtt{K}_{ij}\bar{\mathtt{X}}_j)\bar{\mathbf{a}}_i = \mathtt{K}_{ij}\mathbf{x}_{ij}$. For a given value of i, each measurement $\mathbf{x}_{ij}$ contributes such a equation, and the task is to minimize the squared sum of them, namely

$$\sum_j ||\mathtt{K}_{ij}\mathbf{x}_{ij} - \mathtt{K}_{ij}\bar{\mathtt{X}}_j \bar{\mathbf{a}}_i||^2_{\mathrm{E}} \quad .$$

The required $\mathtt{A}_i$ is found by minimizing this expression over all $\bar{\mathbf{a}}_i$. The solution is found in the usual manner by solving the *normal equations*

$$\begin{aligned} \bar{\mathbf{a}}_i &= (\sum_j \bar{\mathtt{X}}_j{}^\top \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij} \bar{\mathtt{X}}_j)^{-1} \sum_j \bar{\mathtt{X}}_j{}^\top \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij} \mathbf{x}_{ij} \\ &= (\sum_j \bar{\mathtt{X}}_j{}^\top \mathtt{C}_{ij}^{-1} \bar{\mathtt{X}}_j)^{-1} \sum_j \bar{\mathtt{X}}_j{}^\top \mathtt{C}_{ij}^{-1} \mathbf{x}_{ij} \quad . \end{aligned} \tag{6}$$

Solving for $\mathbf{X}_j$. Solving for the $\mathbf{X}_j$ is more simple; as before we need only make sure that the final coordinate of each $\mathbf{X}_j$ is 1. The contribution of $\mathbf{x}_{ij}$ to the total cost is

$$||\mathtt{K}_{ij}(\mathbf{x}_{ij} - \mathtt{A}_i \mathbf{X}_j)||^2_{\mathrm{E}} = ||\mathtt{K}_{ij}(\mathbf{x}_{ij} - \mathbf{t}_i) - \mathtt{K}_{ij}\tilde{\mathtt{A}}_i \tilde{\mathbf{X}}_j||^2_{\mathrm{E}}$$

where $\tilde{\mathtt{A}}_i$ is the 2×3 matrix consisting of the first three columns of $\mathtt{A}_i$, and $\tilde{\mathbf{X}}_j$ is the first three entries of $\mathbf{X}_j$. This is the cost minimized in a least-squares problem

$$(\mathtt{K}_{ij}\tilde{\mathtt{A}}_i)\tilde{\mathbf{X}}_j = \mathtt{K}_{ij}(\mathbf{x}_{ij} - \mathbf{t}_i) \quad .$$

Fixing j, and summing over all i, we need to find the required $\tilde{\mathbf{X}}_j$ that minimizes the sum of the squared residuals. The normal equation solution is obtained as

$$\begin{aligned} \tilde{\mathbf{X}}_j &= (\sum_i \tilde{\mathtt{A}}_i{}^\top \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij} \tilde{\mathtt{A}}_i)^{-1} \sum_i \tilde{\mathtt{A}}_i{}^\top \mathtt{K}_{ij}{}^\top \mathtt{K}_{ij} (\mathbf{x}_{ij} - \mathbf{t}_i) \\ &= (\sum_i \tilde{\mathtt{A}}_i{}^\top \mathtt{C}_{ij}^{-1} \tilde{\mathtt{A}}_i)^{-1} \sum_i \tilde{\mathtt{A}}_i{}^\top \mathtt{C}_{ij}^{-1} (\mathbf{x}_{ij} - \mathbf{t}_i) \quad . \end{aligned} \tag{7}$$

By alternating steps (6) and (7) as in Algorithm 4.3 convergence is reached in a small number of iterations (usually less than 5) from a random starting point.

5.1 Evaluation

Tests were carried out on synthetic data with varying numbers of points and views. Each image point was perturbed by non-isotropic Gaussian noise according to a randomly chosen covariance matrix. The square-roots of the eigenvalues of the covariance matrix (major and minor axes of the uncertainty ellipse) were chosen to be σ, and $r\sigma$, where r was chosen randomly per point, varying between 1 and $\max(r)$, either 20 or 100 in the experiments. The orientation of the uncertainty region was randomly chosen. Each experiment was run 100 times to give a percentage failure rate. The results are given in table 1.

# points	# views	σ	max(r)	% failure
10	10	0.01	20	47
20	20	0.01	20	6
10	100	0.1	100	8
10	100	0.01	100	0
10	100	0.01	20	0
50	50	0.01	20	0
100	100	0.01	20	0

Table 1: *Failure rates for PowerFactorization with uncertainty. The algorithm has 100% success (converging to the guaranteed global minimum) except on extreme cases. Failure occurs with too few (10) points (first row), occasionally with 20 points (row 2), or with extreme noise and a very elongated uncertainty region (third row). Note that the noise level is quoted in fraction of image half-width. Thus for row three, the average uncertainly region (one standard deviation) is of the order of 50 by 2500 pixels for an image size of* 1000×1000 *– totally unrealistic for real data.*

Figure 4: *Three images used in the projective reconstruction experiment. The sequence consisted of 61 images, with 1801 tracks (3D points) and 20605 image points. This represents a missing data rate of* 81.3% *– that is only* 18.7% *of the measurement matrix was populated with data. The strategy for banded data was used to reconstruct this sequence. After a final bundle adjustment the RMS reprojection error was* 0.368 *pixels.*

Projective factorization. The PowerFactorization method may be extended to projective reconstruction as well, by a straight-forward extension of the Sturm-Triggs factorization method. Details are omitted, but some results are shown in Fig 4.

6 Conclusion

The PowerFactorization algorithm is fast, reliable and relatively impervious to noise. Unlike most iterative algorithms it is stand-alone, not requiring an initialization, since it converges from a random starting point. It may be applied to a variety of 3D reconstruction problems.

In the case of affine reconstruction of complete data it is provably rapidly convergent to the global optimal solution. Since the cost surface should not change substantially by omission of some data points, it is expected to converge reliably with missing data, and this is borne out by experiment. Performance degrades gracefully from the certainty of convergence on complete data. It provides a particularly effective method for affine factorization with uncertainty for the case of arbitrary and independent uncertainty regions, for which no algorithm was previously known.

References

[1] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proc. 7th European Conference on Computer Vision, Part I, LNCS 2350, Copenhagen, Denmark*, pages 707–720. Springer-Verlag, 2002.

[2] S. Brandt. Closed-form solutions for affine reconstruction under missing data. In *Proc. 7th European Conference on Computer Vision, Part I, LNCS 2350, Copenhagen, Denmark*, pages 109–114. Springer-Verlag, 2002.

[3] F. De la Torre and M. J. Black. Robust principal component analysis for computer vision. In *Proc. Eighth International Conference on Computer Vision, Vancouver*, pages 362–369, 2001.

[4] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins University Press, 1983.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[6] Michal Irani and P. Anandan. Factorization with uncertainty. In *Computer Vision - ECCV 2000, Volume I, LNCS-Series Vol. 1842, Springer-Verlag*, pages 539 – 553, 2000.

[7] D. W. Jacobs. Linear fitting with missing data for structure-from-motion. *Computer Vision and Image Understanding*, 82:57–81, 2001.

[8] T. Morita and T Kanade. A sequential factorization method for recovering shape and motion from image streams. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(8):858–867, 1997.

[9] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *Proc. 3rd European Conference on Computer Vision, Stockholm*, volume 2, pages 97–108, 1994.

[10] A. Shashua and S. Avidan. The rank-4 constraint in multiple (≥ 3) view geometry. In *Proc. 4th European Conference on Computer Vision, Cambridge*, pages 196–206. Springer-Verlag, 1996.

[11] H. Y. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(9):854–867, 1995.

[12] P. Sturm and W. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. 4th European Conference on Computer Vision, Cambridge*, pages 709–720, 1996.

[13] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[14] W. Triggs, P. F. McLauchlan, R. I. Hartley, and A. Fitzgibbon. Bundle adjustment for structure from motion. In *Vison Algorithms: Theory and Practice*, pages 298–372. Springer-Verlag, 2000.

[15] LŻelnik-Manor and M. Irani. Multiview constraints on homographies. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 214–223, 2002.