Linear Fitting with Missing Data for Structure-from-Motion[*]

David W. Jacobs
NEC Research Institute
4 Independence Way
Princeton, NJ 08540, USA
`dwj@research.nj.nec.com`
phone: (609) 951-2752
fax: (609) 951-2483

Running title: Linear Fitting with Missing Data

[*]A more preliminary version of this work has appeared as [8]

Abstract

Several vision problems can be reduced to the problem of fitting a linear surface of low dimension to data. These include determining affine structure from motion or from intensity images. These methods must deal with missing data; for example in structure from motion, missing data will occur if some point features are not visible in the image throughout the motion sequence. Once data is missing, linear fitting becomes a nonlinear optimization problem. Techniques such as gradient descent require a good initial estimate of the solution to ensure convergence to the correct answer. We propose a novel method for fitting a low rank matrix to a matrix with missing elements. This method produces a good starting point for descent type algorithms, and can produce an accurate solution without further refinement. We then focus on applying this method to the problem of structure-from-motion. We show that our method has desirable theoretical properties compared to previously proposed methods, because it can find solutions when there is less data present. We also show experimentally that our method provides good results compared to previously proposed methods.

1 Introduction

Recently, several important vision problems have been addressed by using affine imaging models, which reduce the problems to ones of finding a low-dimensional linear surface that represents some important scene structure. For example, in the linear combinations work of Basri and Ullman[21, 2], the set of images produced by any 3-D structure from different viewpoints is represented as a low-dimensional surface in the space of all possible images. This structure is then used for object recognition. Tomasi and Kanade[20] use a related insight to efficiently perform structure-from-motion in long motion sequences. Since the noise-free images should all lie in a low-dimensional linear surface, such a surface can be fit to the data and then used to derive structure and motion. Moses[12] and Shashua[16] show that the set of intensity images produced by a 3-D Lambertian structure from a single viewpoint but with a point light source of varying position and intensity can also be described as a 3-D linear surface in a space of possible images. Belhumeur and Kriegman[3] use this result to describe the set of intensity images produced by a convex object, allowing for multiple light sources and self-shadowing. In all of this work, the problem is greatly simplified by finding a reasonable set of assumptions to make it linear.

All of these methods require fitting a matrix of low rank, r, to a data set. The linear space spanned by this matrix corresponds to scene structure. This can be done by acquiring only the minimal amount of data needed to construct a $r \times n$ matrix ([21, 12, 16, 3]), which has exactly rank r even in the presence of noise. This method is simple, but can be quite sensitive to noise. Ullman and Basri also propose using singular value decomposition (SVD) to fit a low rank matrix to a larger matrix built when extra images are used to overdetermine the problem. Tomasi and Kanade use a long motion sequence to build a large matrix, and then find the nearest low rank matrix using SVD. Hayakawa[7] applies SVD to find the linear surface produced by varying illumination. SVD finds the low rank matrix that minimizes the sum of square differences between each element in the new matrix and the data matrix, reducing the effects of noise.

A significant limitation of these methods is that they require a data matrix with no missing elements. In the case of structure-from-motion this means that every point feature must be visible in every image. This is a strong assumption, because often points enter or leave the field of view during a motion sequence, or may be blocked from view by other objects in the middle of a sequence. While this problem can be addressed by working only with subsets of the points and frames that together have no missing elements, this is undesirable for two reasons. First, one will obtain suboptimal solutions that do not take account of all available data. Unless large subsets of points are mutually visible over large camera motions, results can be unstable. Second, the resulting solutions will not have a common reference frame.

The methods that describe the intensity manifold produced by a scene use a set of images in which illumination in each image comes from a single point light source. The data matrix will have missing elements when scene points are not illuminated by the light source; therefore these methods require that every visible scene point be illuminated in every image. However, for scenes with smooth objects, this assumption will be met only when the light source direction is the same as the viewing direction, ie. for one possible image. More generally, most visible points will be illuminated only when the light source is near the viewing direction. However, if images are generated with similar light source directions, the resulting solution may be unstable.

Tomasi and Kanade recognized that more reliable and general solutions can be obtained by fitting a low rank matrix to a data matrix that has some missing elements, and provided a method of doing this. We describe this method in Section 2, and compare it to our own in Section 4. Shum, Ikeuchi and Reddy[17] apply an iterative method due to Wiberg[22] to a related structure from motion problem. This method minimizes the sum of square differences between the fitted, low rank matrix and the elements that are not missing in the data matrix. This approach has the virtue of converging to a locally optimal solution, however it is not guaranteed to find the globally optimal one. We describe this method in Section 2, and later describe experiments in which we apply their method to our problems. Other authors (eg., Hartley[6] McLauchlan et al.[11], and Soatto and Perona[18]) have also applied iterative error minimization methods to the problem of determining structure-from-motion. We show, as others have also noted (eg., Hartley[6]) that these methods can often converge to the wrong answer if they do not have a good starting point.

We present a novel method of fitting a low rank matrix to a matrix with missing elements. Our method works by combining constraints on the solution derived from small submatrices of the full matrix. Specifically, we show that for every r-tuple of columns in the original matrix, the space spanned by all possible completions of these columns must contain the column space of the completely filled matrix. Solutions satisfying these partial constraints can be found using standard techniques of linear algebra, which take account of the effects of noise.

We then show how this algorithm can be applied to determining structure-from-motion. This requires modifications to account for object translation, and allows us to take advantage of the particular pattern of missing data that occurs in such problems. We also point out the potential of our basic algorithm for handling intensity image description problems. We describe experiments to evaluate the accuracy of our method, and compare it to previously proposed methods, both theoretically and experimentally.

Our method has a heuristic approach to handling noise, and does not produce an optimal solution. Therefore, it is most suitable as a way of providing a starting point for iterative methods. We will experimentally show its value as a starting point, and also, in some cases, as a quick and dirty solution.

2 Background

In this section we first show in more technical detail how the vision problems we consider have been posed as linear surface fitting problems. We then review previous methods for handling missing data. After presenting our own algorithm, we will compare it to previous approaches in Section 4.

2.1 Linear Fitting

The methods proposed by Ullman and Basri and Tomasi and Kanade assume that a sequence of matched image points is given, and that these were projected from 3-D to 2-D using a camera model that is affine, or a subset of affine (eg. orthographic, weak perspective, or paraperspective [13, 19, 1]). The translation component of motion is estimated separately, by estimating the translation of the center of mass of the points. This method is no longer suitable when there is missing data. To simplify our presentation, we will first consider the case in which there is no translation. In Section 3.4 we will then show how to handle translational components of motion.

Let $\mathbf{p}_i = (x_i, y_i, z_i)^T$ be the i'th 3-D point feature. Then we may write the transformation mapping the model to image j as a 2×3 matrix, $\mathbf{S}_j$. Letting $\mathbf{q}_{i,j} = (u_{i,j}, v_{i,j})$ stand for the image point produced by $\mathbf{p}_i$, this gives:

$$\begin{pmatrix} u_{i,j} \\ v_{i,j} \end{pmatrix} = \begin{pmatrix} S_{j_{1,1}} & S_{j_{1,2}} & S_{j_{1,3}} \\ S_{j_{2,1}} & S_{j_{2,2}} & S_{j_{2,3}} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

For the case of scaled-orthographic projection, $\mathbf{S}_j$ is the first two rows of a scaled rotation matrix.

We can now represent an entire image sequence in simple matrix notation. Let $\mathbf{P}$ be a matrix whose i'th column is $\mathbf{p}_i$:

$$\mathbf{P} \equiv \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & . & . & . & y_n \\ z_1 & z_2 & . & . & . & z_n \end{pmatrix}$$

Let $\mathbf{S}$ stand for a matrix containing all rows of all transformations S_j, that is, row $2i-1$ of $\mathbf{S}$ is the first row of $\mathbf{S}_i$, and row $2i$ of $\mathbf{S}$ is its second row.

$$\mathbf{S} \equiv \begin{pmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ . \\ . \\ . \\ \mathbf{S}_{\frac{m}{2}} \end{pmatrix}$$

Finally, let $\hat{\mathbf{Q}}$ stand for a single matrix containing all tracked image points. Each column of $\hat{\mathbf{Q}}$ contains all the image coordinates of a single point. That is, $\hat{Q}_{(2i-1),j}$ gives the u coordinate of $\mathbf{p}_i$ in image j, and $\hat{Q}_{(2i),j}$ gives its v coordinate.

$$\hat{\mathbf{Q}} \equiv \begin{pmatrix} \hat{u}_{1,1} & \hat{u}_{1,2} & . & . & . & \hat{u}_{1,n} \\ \hat{v}_{1,1} & \hat{v}_{1,2} & . & . & . & \hat{v}_{1,n} \\ \hat{u}_{2,1} & \hat{u}_{2,2} & . & . & . & \hat{u}_{2,n} \\ \hat{v}_{2,1} & \hat{v}_{2,2} & . & . & . & \hat{v}_{2,n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \hat{u}_{\frac{m}{2},1} & \hat{u}_{\frac{m}{2},2} & . & . & . & \hat{u}_{\frac{m}{2},n} \\ \hat{v}_{\frac{m}{2},1} & \hat{v}_{\frac{m}{2},2} & . & . & . & \hat{v}_{\frac{m}{2},n} \end{pmatrix}$$

where $(\hat{u}_{i,j}, \hat{v}_{i,j})$ are the noisy $(u_{i,j}, v_{i,j})$ coordinates. Let $\mathbf{E}$ be a matrix of the same dimension as $\hat{\mathbf{Q}}$, whose elements indicate the noise encountered in sensing the corresponding elements of $\hat{\mathbf{Q}}$. Then we may simply describe the generation of the whole image sequence as:

$$\mathbf{SP} + \mathbf{E} = \hat{\mathbf{Q}}$$

We further define:

$$\mathbf{Q} \equiv \hat{\mathbf{Q}} - \mathbf{E} = \mathbf{SP}$$

Although we have no direct access to the error-free data, $\mathbf{Q}$, we wish to estimate it from $\hat{\mathbf{Q}}$. Because $\mathbf{S}$ has three columns, it can only have rank three, and consequently $\mathbf{Q}$ can have only rank three.[1]

The matrix containing the coordinates of sensed image points, $\hat{\mathbf{Q}}$, will be a noisy version of $\mathbf{Q}$, and will have full rank. However, if we find the $\mathbf{Q}$ matrix of rank three that is closest to $\hat{\mathbf{Q}}$, we can obtain a maximum likelihood estimate of $\mathbf{Q}$ when the error is Gaussian. In particular, we can use SVD to find the $\mathbf{Q}$ matrix that minimizes $\sum_i \sum_j (\hat{Q}_{ij} - Q_{ij})^2$. We can also use SVD to factor this $\mathbf{Q}$ matrix into $\mathbf{S}$ and $\mathbf{P}$, providing us with estimates of the structure and motion of the 3-D points, up to an affine transformation. See Tomasi and Kanade for further details.

When some points are not visible in some images, $\hat{\mathbf{Q}}$ will contain missing entries for these points. We still know that the error free data, were it all visible, would give rise to a $\mathbf{Q}$ matrix of rank 3. We can therefore find the affine structure and motion that minimizes error by finding the $\mathbf{Q}$ matrix that is closest to $\hat{\mathbf{Q}}$ where we now compare only those elements actually present in $\hat{\mathbf{Q}}$ to the corresponding elements of $\mathbf{Q}$. The problem of finding structure from motion in the presence of missing data then, depends on finding the nearest rank three matrix to a data matrix with missing elements.

A similar formulation has been used by Shashua, Moses, Hayakawa, and Belhumeur and Kriegman to describe the set of intensity images produced by a 3-D Lambertian structure exposed to a point light source at infinity. Let the direction of the source in image j, scaled by its intensity be the 3-D vector: $-\mathbf{h}_j$. In this case, let $q_{i,j}$ be the intensity of image pixel i in image j. Finally, assume that this intensity is produced by a 3-D planar patch of the scene with a surface normal $\mathbf{n}_i$ and albedo ρ_i. Then, intensities are produced according to the equation:

$$q_{i,j} = \mathbf{h}_j \cdot \rho_i \mathbf{n}_i$$

provided that this produces a positive value (ie. each surface normal is facing the light source). Since we can never recover the absolute magnitude of $\mathbf{h}_j$ or ρ_i from a set of these equations, we may without loss of generality assume that $\rho_i \mathbf{n}_i$ is simply an unconstrained 3-D vector (that is, the constraint that albedo is less than one has no effect without some limitation on the scale of $\mathbf{h}_j$). Next, assume we take a set of images in which the viewpoint is fixed, but the lighting intensity and direction may vary. Let $\mathbf{N}$ be a scene matrix in which the j'th column is $\rho_i \mathbf{n}_i$. Let $\mathbf{H}$ be a lighting matrix in which the j'th row is $\mathbf{h}_j$. And let $\hat{\mathbf{Q}}$ be a data matrix of the $q_{i,j}$. Then we have:

$$\hat{\mathbf{Q}} = \mathbf{H}\mathbf{N} + \mathbf{E}$$

[1]That is, the columns (or rows) of $\mathbf{Q}$, regarded as points, must occupy no more than a 3-D linear subspace of $\mathcal{R}^m$ ($\mathcal{R}^n$). Any column (or row) will be a linear combination of any other three linearly independent columns (or rows).

$$\mathbf{Q} \equiv \hat{\mathbf{Q}} - \mathbf{E} = \mathbf{HN}$$

where $\mathbf{E}$ is again a matrix of sensing errors. Again $\mathbf{Q}$ has rank three. Again, our noisy images provide a full rank matrix $\hat{\mathbf{Q}}$, and again, we can estimate $\mathbf{Q}, \mathbf{H}$ and $\mathbf{N}$ from $\hat{\mathbf{Q}}$ by using SVD to find the nearest rank three matrix to $\hat{\mathbf{Q}}$, along with its decomposition. Shashua uses the matrix $\mathbf{N}$ to determine whether a novel image could be generated by the same scene, by determining whether the novel image is a linear combination of the rows of $\mathbf{N}$.

For this problem, $\hat{\mathbf{Q}}$ will have missing elements when we can detect that pixel values do not arise from the formula $q_{i,j} = \mathbf{h}_j \cdot \rho_i \mathbf{n}_i$. This occurs when some of the scene surface normals do not face the light source. For if $\mathbf{h}_i \cdot \rho_j \mathbf{n}_j < 0$, the scene point is *self-shadowed*, producing zero intensity plus noise, not negative intensity, where "noise" includes effects not accounted for in this lighting model, such as interreflections. If a pixel value appears darker than some threshold, we can infer that it may be self-shadowed, and delete it from the matrix. Similarly, if we can detect that some pixel values are due to specularities because they saturate the sensor, we may remove them from the data matrix. As we will see, we can use extra images to fill in this missing data, so it will be safe to be aggressive in deleting questionable intensity values. Belhumeur and Kriegman use $\mathbf{N}$ to determine the illumination cone that describes the set of images that a scene can produce, allowing for self-shadowing, but they still build $\mathbf{N}$ using images in which there is no self-shadowing.

2.2 Missing data

The problem of finding a good approximation to a matrix with missing elements appears to be much harder than that of approximating a noisy, but complete matrix. To see this, consider the case in which there is no sensing error. With no missing data, this is readily solved using SVD. With missing data, this becomes a problem of determining how to fill in missing elements while keeping the matrix at rank r. The filled in matrix will have rank r if and only if every $(r+1) \times (r+1)$ submatrix has determinant zero. If one treats missing elements as unknowns, these determinants give rise to a series of nonlinear equations. Thus, it appears that our problem has become an optimization problem with nonlinear constraints.

The problem of filling in missing matrix elements, or otherwise perturbing a matrix so that it will have a specified rank has received considerable attention in mathematics, inspired in part by problems that arise in control. Solutions have been found for specialized versions of the problem, for example, when the missing elements form a rectangular block of the matrix ([4]). However, solutions do not exist for the general problem of fitting a low rank matrix to a matrix with missing elements, even in the absence of sensing error, and closely related problems are known to be NP-hard (eg., [14]).

Related problems have also been studied in statistics. While many of the methods developed are specialized for regression analysis, two methods are related to techniques that have been used in computer vision. In *Regression imputation*, linear fits to the data that are present are used to predict the values of missing entries. In Model-based techniques, iterative methods such as EM are used to find maximum likelihood estimates for unknown values, given some model of their properties. A taxonomy of these methods and a detailed discussion can be found in Little and Rubin[10].

Several methods have been suggested for computing structure-from-motion with missing data. Solutions can generally be divided into two parts: methods for obtaining an initial estimate of structure and/or motion (*initialization* methods); and methods for refining this estimate. Methods of refining an initial estimate can be further divided into batch and filtering methods. In batch methods all the data is assumed to be available, and an iterative method is used that progressively reduces error and converges to a solution that locally minimizes the sum of squares distance between the data and the inferred solution. Newton's method and related gradient descent algorithms are examples of such batch methods. We will refer to these methods generally as *iterative*.[2] In filtering methods, a solution is computed using initial frames of the motion sequence, and then this solution is updated as each new frame is acquired. We view our own algorithm as a method for providing an initialization method that can serve as the starting point for an iterative algorithm. We therefore especially consider previous initialization methods and evidence that indicates when initialization is important.

Tomasi and Kanade propose one initialization method, which they then refine using gradient descent. Their algorithm is related to regression imputation techniques in statistics. In their method, one first locates a rectangular subset of the matrix with no missing data. Next one applies their algorithm to the problem with no missing data. Then the missing elements of an additional column (or row) are filled in by finding the linear combination of a basis for the columns (or rows) of the submatrix that best fits the non-missing elements of the new column (or row).

This solution seems like a reasonable heuristic because missing data is filled in using overconstraining information, and Tomasi and Kanade demonstrate it working effectively

[2]This is a bit imprecise, because initialization methods may use SVD. However, though SVD may be computed with an iterative algorithm, it is an algorithm guaranteed to converge quickly to the globally optimal solution, and so differs in important ways from the algorithms we call iterative.

on two motion sequences. However, the proposed method has several potential disadvantages. First, the problem of finding the largest full submatrix of a matrix with missing elements is NP-hard (max-clique can be easily reduced to this problem) so heuristics must be used to find a starting point. Second, the data is not used symmetrically. A small subset of the data may be used to compute the first missing elements, which may lead to significant inaccuracies in these elements. It is not clear how these inaccuracies will propagate in the computation of additional missing elements. Third, for an $m \times n$ matrix, the method requires $O(max(m, n))$ applications of SVD . While two experiments on real data are shown, it is not clear how effective a starting point the method will provide in general. In Section 4 we will compare the theoretical properties and experimental performance of this algorithm to those of our own.

McLauchlan, Reid and Murray present a filtering method for determining affine structure and motion. They initialize their method by applying Tomasi and Kanade's algorithm to the first few frames of the motion sequence, using only points present in all these first frames. Their filtering method is suitable for the case when there is missing data. However, they show experiments only for the case in which there is no missing data. In this case, they show that their initial estimate is sufficient to allow the method to converge to the correct solution. The contribution of McLauchlan et al. is to show that a filtering method can more efficiently produce results essentially as accurate as the optimal solution produced by the Tomasi and Kanade algorithm.

However, it should be noted that when no missing data is present, the problem of determining structure and motion is an optimization problem with a unique local minimum that is the globally optimal solution. When there is missing data, the problem begins to have many locally optimal solutions that are not globally optimal. Therefore, McLauchlan et al's experimental validation that their algorithm converges to the right solution when there is no missing data is only weak evidence that the algorithm will also work well in the presence of significant amounts of missing data.

Soatto and Perona[18] describe an interesting filtering method for determining motion assuming scaled-orthographic projection. Their focus is on deriving filters in which the state space to be estimated is small. Their method is also geared to handle missing data, since only motion estimates, not structure estimates, are fused. They also present experiments in which there is no missing data, and do not consider in detail the question of when their algorithm will converge to a correct solution in the presence of missing data.

Hartley[6] presents a method for finding projective structure and motion from multiple frames using the Levenberg-Marquardt algorithm, a variation on Newton's method of gradient descent. This is a batch algorithm. Hartley initializes this algorithm using the results of a two-frame linear algorithm for projective structure and motion. While Hartley does not apply this algorithm to the case of missing data, it is clear that the gradient descent method can be applied readily in the case of missing data, simply by omitting error terms involving missing elements. However, it is not clear how the results of the two frame algorithm can be combined to initialize the algorithm, when there is missing data. In some sense, our algorithm is a solution to this problem for the affine case. Hartley also stresses the importance of having a good initial estimate, to ensure convergence to a correct solution.

Shum, Ikeuchi, and Reddy[17] have also done recent work relevant to the missing data problem. They consider the problem of deriving structure and motion based on 3-D data whose significance can be weighted, leading to a weighted least squares problem of fitting a low rank matrix to a data matrix. For this problem they suggest the use of an iterative method due to Wiberg[22]. This iterative method clearly can also be applied to the missing data problem considered in this paper. This method is described in detail in [17]. We can briefly describe the method as formulating the least squares problem as a bilinear optimization, and then iteratively holding one set of variables constant while the others are optimized, so that each optimization is linear. We use their method in our experiments, because it has good convergence properties and is easy to implement. For the problem they consider, Shum et al. state that a random starting point is sufficient to produce a good final solution. However, their experiments on this point cannot be used to draw conclusions for the problem of determining 3-D structure from a sequence of 2-D images.

Finally, Kahl and Heyden[9] present a method of handling missing data related to our earlier version of this work (Jacobs [8]). We discuss this relationship further after presenting our algorithm.

3 A Novel Algorithm

We now describe a new algorithm for fitting a low rank matrix to a matrix with missing data. In the absence of sensing error this algorithm will produce a correct answer, or report that it does not have enough constraint available to solve the problem. However, when sensing error is present, the algorithm will not necessarily produce a locally optimal solution. Therefore, it is most suitable as an initialization method to be followed by a batch, iterative method to refine the solution.

We formulate a set of constraints on the solution that hold exactly in the absence of error, and then use SVD to find a solution that comes close to satisfying these constraints when error is present. Our algorithm can be applied to fit matrices of any low rank to data, but for notational simplicity we will assume that we are looking for a rank three matrix. Our presentation has three parts. First we derive a set of constraints on the solution that are valid in the error-free case. Then we describe a basic algorithm that applies these constraints, and show how to allow for sensing error in this algorithm. Finally, we discuss a number of heuristic modifications that increase robustness.

One can define an optimal solution to the missing data problem as one of finding a low rank matrix whose elements are as near as possible to the elements present in the original matrix. Our algorithm does not produce such an optimal solution. The goal of our algorithm, rather, is to use simple linear algebra operations to estimate the solution. This estimate can either be used directly, or as the starting point for an iterative algorithm. We discuss these issues in more detail in subsequent sections.

3.1 Constraints for the Error-Free Case

We can regard each column of $\mathbf{Q}$ as giving the coordinates of a point in an m-dimensional space. In this view, SVD finds the 3-D linear subspace, $\mathcal{L}$, that is "closest" to the n m-dimensional points in $\hat{\mathbf{Q}}$, when this matrix contains no missing elements. $\mathcal{L}$ is closest to these points in that it minimizes the sum of square distances to them. At the same time, SVD constructs the columns of $\mathbf{Q}$ by finding each point in $\mathcal{L}$ that is closest to the corresponding point in $\hat{\mathbf{Q}}$.

When $\hat{\mathbf{Q}}$ has missing elements, we are free to fill in these elements to improve the fit between model and data. A column of $\hat{\mathbf{Q}}$ with missing elements is a point with some coordinates unknown; that is, it is an affine subspace in an m-dimensional space (and is parallel to the axes of this space). Our problem in this case is to find the 3-D linear subspace, $\mathcal{L}$, that is closest to these affine spaces, as measured by the sum of square distances between them. $\mathbf{Q}$ and the missing elements of $\hat{\mathbf{Q}}$ can then be found by assigning each column of $\mathbf{Q}$ to be the point in $\mathcal{L}$ closest to the affine space represented by the corresponding column of $\hat{\mathbf{Q}}$, and the missing elements of this column of $\hat{\mathbf{Q}}$ can be filled by taking the point in this affine subspace closest to $\mathcal{L}$.

To begin, we suppose there is no noise. Then $\mathcal{L}$ intersects all the affine spaces described by the columns of $\hat{\mathbf{Q}}$. Let A_i, A_j, A_k denote three of these affine subspaces that do not intersect [3]. Since $\mathcal{L}$ contains a unique point other than the origin in common with each of these affine spaces, $\mathcal{L}$ must lie inside the space spanned by the points in these three spaces. Therefore, each triple of columns of $\hat{\mathbf{Q}}$ constrains $\mathcal{L}$ to lie in some linear subspace of $\mathcal{R}^m$. The significance of this constraint will depend on the number of missing elements in these columns. If the columns have no missing elements, they specify $\mathcal{L}$ exactly. Alternately, if the columns have many missing elements, the affine spaces they give rise to will be of high dimension, and may span a very high dimensional space; in this case the constraint that $\mathcal{L}$ lie in this space will be a weak one.

We can combine constraints derived from many different triples of columns of $\hat{\mathbf{Q}}$ just by taking the intersection of the linear spaces to which they lead. Specifically, let $\mathcal{T}$ be some set of triples of affine spaces corresponding to columns of $\hat{\mathbf{Q}}$, and let $span(A_i, A_j, A_k)$ denote the linear space spanned by the points in the affine spaces A_i, A_j, A_k. Then we have:

$$\mathcal{L} \subseteq \cap_{\{i,j,k\} \in \mathcal{T}} \quad span(A_i, A_j, A_k) \tag{1}$$

Note that we get a subset, not an equality relationship. This means that if

$$\cap_{(i,j,k) \in \mathcal{T}} \quad span(A_i, A_j, A_k)$$

is three-dimensional we know $\mathcal{L}$ exactly, but if this intersection is of higher dimension we do not. Moreover, if we are unsure of the rank of $\mathbf{Q}$, $\cap_{(i,j,k) \in \mathcal{T}} \quad span(A_i, A_j, A_k)$ gives us an upper bound on this rank.

3.2 The Algorithm

In a practical algorithm, we must allow for the possibility of sensing error. With error, $\cap_{(i,j,k) \in \mathcal{T}} \quad span(A_i, A_j, A_k)$ quickly becomes empty, because there is no linear subspace that can perfectly match our data by intersecting the affine spaces associated with each column. So in this case, we compute a linear subspace that comes close to lying in this intersection.

To begin, we select l triples of columns of $\hat{\mathbf{Q}}$. For structure-from-motion, this is addressed in Subsection 3.4. Let $\mathcal{S}_i$ denote the linear space spanned by the i'th triple of columns. A convenient way to take the intersection of these spaces is by taking the complement of the span of their complementary spaces. Let $\mathcal{S}_i^\perp$ denote the orthogonal complement of $\mathcal{S}_i$ Let N_i denote a matrix representation of $\mathcal{S}_i^\perp$. That is, each column of

[3]This assumption that the spaces do not intersect is not significant in practice. Since the affine spaces are parallel to the coordinate axes, two spaces intersect only when for every row they have either identical values, or one has a missing value. Such triples are easily discarded in the algorithm we describe.

N_i represents a vector orthogonal to the space $\mathcal{S}_i$. Further, let $\mathcal{S}$ denote the space lying in the intersection of all the $\mathcal{S}_i$, so that from above we know that $\mathcal{L} \subseteq \mathcal{S}$. Our task is to compute $\mathcal{S}$.

Now note that:

$$\mathbf{N} = [N_1 N_2 ... N_l]$$

provides us with a representation of $\mathcal{S}^{\perp}$. This is because a vector lying in the complement of any $\mathcal{S}_i$ cannot lie in $\mathcal{S}$, the intersection of all the $\mathcal{S}_i$. Therefore, $\mathcal{S}$ can be found by taking the nullspace of $\mathbf{N}$.

In the presence of sensing error, the matrix $\mathbf{N}$ will typically have full rank, and its nullspace will be empty. In this case, we want to find a 3-D linear space that is close to being this nullspace, that is, the space that is most easily made into the nullspace by allowing for small amounts of noise. This is done by taking the SVD of N, and finding its three least significant components. This gives us an estimate of $\mathcal{L}$ as the 3-D linear subspace most nearly orthogonal to the space spanned by $\mathbf{N}$. Specifically, $\mathcal{L}$ is estimated as the linear space that is orthogonal to the matrix closest to $\mathbf{N}$ according to the Frobenius norm, such that this close matrix has a rank three null space.

It is important to note that the problem of finding the three least significant components of $\mathbf{N}$ does not typically present numerical difficulties. This is essentially equivalent to the problem of finding the three eigenvectors associated with the three smallest eigenvalues of a matrix. Although finding the exact values of these nearly zero eigenvalues might prove difficult, finding their associated eigenvectors need not be so (if it were, one could never hope to reliably find the nullspace of a matrix). Essentially, the speed and accuracy with which we can find the three least significant components of $\mathbf{N}$ will depend on the gap between the values of the three smallest singular values of $\mathbf{N}$ and the fourth smallest singular value. As our experiments demonstrate, in practice this problem can be accurately solved. For more on this subject, see Golub and van Loan[5].

The three least significant components of the SVD of $\mathbf{N}$ provide a representation of $\mathcal{L}$ as three orthonormal column vectors that span $\mathcal{L}$. We then use these to construct $\mathbf{Q}$. For each column of $\hat{\mathbf{Q}}$, we find the linear combination of the columns representing $\mathcal{L}$ that best fit the elements that are not missing. Specifically, assume $\mathcal{L}$ is represented by three orthonormal columns, which together we call $\mathbf{L}$. Let $\hat{Q}_i$ represent a column of $\hat{\mathbf{Q}}$, let p be the indices of non-missing elements of $\hat{\mathbf{Q}}_\mathbf{i}$, and let $\hat{Q}_i^p$ be the column obtained by taking only the non-missing elements of $\hat{Q}_i$, and let L^p similarly be the submatrix of $\mathbf{L}$ consisting of rows with indices in p. Then we estimate:

$$Q_i \approx L((L^p)^+ \hat{Q}_i^p)$$

where $(L^p)^+$ is the pseudoinverse of L^p.

3.3 Implementation Details

We now mention several details that are important to the implementation of this algorithm. First, we describe how we compute the nullspace of the space spanned by three affine spaces. Let C_i, C_j, C_k be three columns of $\hat{\mathbf{Q}}$ that give rise to three affine spaces A_i, A_j, A_k. We note that if some row, h, of C_i, (or of C_j or C_k) is missing an element, then row h will have a zero value for every element of the null space of $span(A_i, A_j, A_k)$. To see why this is true, consider two different completions of C_i that are identical except in row h. Any column of the null space must have an inner product of zero with each of these possible completions. This can only happen if the column of the null space is zero in row h. Therefore, to compute the nullspace, we need only find the nullspace of the submatrix of $[C_i C_j C_k]$ that is fully occupied, filling in the remaining rows of the nullspace with all zeros.

Second, it can happen that the null space of $span(A_i, A_j, A_k)$ is not an accurate constraint on $\mathcal{L}$, due to numerical instability. As a simple example, suppose that A_i, A_j, A_k are just three points that, along with the origin, nearly lie in some 2-D plane, P. Then their span is 3-D, and in the absence of sensing error, would specify $\mathcal{L}$ exactly. However, in practice any 3-D linear subspace that includes P will lie close to all three points, and when noise is added their span will be a pretty arbitrary 3-D linear subspace that includes P. It would therefore be mistaken to expect $\mathcal{L}$ to lie in their span. This situation, however, is readily detected. We compute the nullspace by applying SVD to the submatrix of $[C_i C_j C_k]$ that is fully occupied. If this submatrix is not, or nearly not of full rank, we know that the constraints derived from it are unreliable.

Third, is the issue of how we choose triples of columns with which to compute $\mathbf{N}$. Here we describe our method for the generic algorithm. Subsection 3.4 describes modifications for application to structure-from-motion. We randomly sample a fixed set of column triples of size l to compute $\mathbf{N}$. However, we do not know ahead of time how many columns of $\mathbf{N}$ each of these triples will produce. If $\mathbf{N}$ grows too big, finding its SVD will be unnecessarily costly. We therefore also set a maximum value on the number of columns of $\mathbf{N}$ needed, and we stop generating column triples when this size is reached, even if we have not yet checked l triples. Also, in an effort to provide information about all rows of $\mathbf{N}$, we bias our selection of column triples. If we have no information about some rows of $\mathbf{N}$, then half the time we select a triple of columns that all have entries in one of those rows.

Fourth, it is always possible that the column triples that we have generated do not sufficiently constrain $\mathcal{L}$ to allow us to accurately determine it, because Equation 1 is only a subset relationship. We can also check this possibility. Recall that $\mathcal{L}$ is computed by taking the three least significant components of $\mathbf{N}$, based on its three smallest singular values. If the fourth smallest singular value of $\mathbf{N}$ is also very small, we know that our computation of $\mathcal{L}$ is unreliable. In this case, we simply report that the algorithm failed to find an answer, although we might also try the computation again, hoping that a larger value for l or a different random selection of column triples would produce more information.

Fifth, it is possible that $\mathbf{N}$ does not directly contain enough information to allow us to determine every column and row of $\hat{\mathbf{Q}}$. For example, a row of $\mathbf{N}$ will consist of all zeros when every triple of columns selected had at least one missing element in that row. In this case, we have gathered no information about that row's value in the nullspace we are computing. Also, it is possible that $\mathcal{L}$ does not allow us to estimate Q_i as we have indicated, because (L^p) does not have full rank, and we can not take its pseudoinverse. In either of these situations, we simply estimate the maximum subrectangle of $\mathbf{Q}$ for which we have data, and then extend this subrectangle to a least-squares estimate of the full rectangle, in a manner that is similar to that proposed by Tomasi and Kanade. It is possible that in attempting to extend a partial solution we do not have sufficient information to stably complete the matrix. In this case, we report that our solution is unstable. At the same time, we can also use a sixth heuristic, that of attempting our method on the transpose of the data matrix, rather than on the original matrix. If this produces a more accurate solution, we use it.

3.4 Applying the Algorithm to Structure-from-Motion

So far we have described a generic algorithm that can be applied to any problem of linear fitting with missing data. The problem of structure-from-motion differs from this generic problem in two ways: the fact that the x and y coordinates of a point are either both known, or both missing; and the presence of translation. We modify our basic algorithm slightly to take account of these facts.

First, a point's x and y coordinates are observed together. This means that $\hat{\mathbf{Q}}$ cannot have a completely arbitrary pattern of missing elements. To take advantage of this, we apply the algorithm described above to the transpose of $\hat{\mathbf{Q}}$, so that each column contains all the x or all the y coordinates in a single frame. The generic algorithm is effective when column triples are chosen that have many entries in common. Since we know that the x and y columns from a single frame have all their entries in common, it makes sense to always consider these together. So, rather than selecting a random sampling of triples of columns, we select all pairs of frames in the motion sequence. Each of these provides us with four columns that we may use to build $\mathbf{N}$.

Second, when translation is present in a motion sequence, we may write the image generation process in the form:

$$\mathbf{S} = \begin{pmatrix} S_{1_{1,1}} & S_{1_{1,2}} & S_{1_{1,3}} & t_{1_x} \\ S_{1_{2,1}} & S_{1_{2,2}} & S_{1_{2,3}} & t_{1_y} \\ & \cdot & & \\ & \cdot & & \\ & \cdot & & \\ S_{\frac{m}{2}_{1,1}} & S_{\frac{m}{2}_{1,2}} & S_{\frac{m}{2}_{1,3}} & t_{\frac{m}{2}_x} \\ S_{\frac{m}{2}_{2,1}} & S_{\frac{m}{2}_{2,2}} & S_{\frac{m}{2}_{2,3}} & t_{\frac{m}{2}_y} \end{pmatrix} \qquad \mathbf{P} \equiv \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & . & . & . & y_n \\ z_1 & z_2 & . & . & . & z_n \\ 1 & 1 & . & . & . & 1 \end{pmatrix} \qquad \mathbf{SP} + \mathbf{E} = \hat{\mathbf{Q}}$$

where (t_{i_x}, t_{i_y}) represents the translation in frame i. This differs from the generic case we have discussed because now $\hat{\mathbf{Q}}$ has rank four, and one of the vectors spanning its row space is known to be $\mathbf{1}$, a vector of all ones. We therefore modify our algorithm to look for a 4-D linear space that includes this vector.

Specifically, we consider all pairs of frames, i and j. For each pair, we form the submatrix of:

$$\begin{pmatrix} 1 & \hat{X}_{i,1} & \hat{Y}_{i,1} & \hat{X}_{j,1} & \hat{Y}_{j,1} \\ 1 & \hat{X}_{i,2} & \hat{Y}_{i,2} & \hat{X}_{j,2} & \hat{Y}_{j,2} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & \hat{X}_{i,n} & \hat{Y}_{i,n} & \hat{X}_{j,n} & \hat{Y}_{j,n} \end{pmatrix}$$

consisting of the rows that have no missing elements. Suppose there are k such rows. Provided that $4 < k$, this $k \times 5$ submatrix will have rank four, in the absence of sensing error, and its nullspace will be represented by a $k \times (k-4)$ matrix. We may use this nullspace to create $\mathbf{N}$ in just the way we did for the generic algorithm. Since sensing error will cause this submatrix to have rank five, we may use SVD to determine and discard the least significant component, estimating the true, error free submatrix. As before, if this component is significant, we may detect that our estimate of the component of $\mathbf{N}$ due to this submatrix is likely to be unreliable, and discard it.

With these modifications, our algorithm can now be regarded as one that combines together partial estimates of the scene structure from each pair of frames. While such a two-frame algorithm would be easy to apply without missing data, when each pair of

Frame ↓	coord. ↓	Point ⇒	1	2	3	4	5	6	7	8	9	10	11	12
1	x		X	X	X	X	X	X	X	X	X	-	-	-
	y		X	X	X	X	X	X	X	X	X	-	-	-
2	x		X	X	X	X	X	X	-	-	-	X	X	X
	y		X	X	X	X	X	X	-	-	-	X	X	X
3	x		X	X	X	-	-	-	X	X	X	X	X	X
	y		X	X	X	-	-	-	X	X	X	X	X	X
4	x		-	-	-	X	X	X	X	X	X	X	X	X
	y		-	-	-	X	X	X	X	X	X	X	X	X

Table 1: An example pattern of missing data in a four frame motion sequence. 'X' indicates that a value is observed. '-' indicates that the value is missing.

frames provides an estimate of structure for only a subset of the scene points it is not obvious how these can be combined, and merged into a common reference frame. In a sense, our algorithm is a solution to this problem.

4 Comparison to past methods

We are now in a position to better situate our new algorithm relative to previous work. Our algorithm uses standard methods of linear algebra, primarily SVD, to estimate structure and motion from missing data. The algorithm does not produce a solution that is optimal, in the sense of minimizing the sum squared error between the estimated structure and motion and the observations. Therefore, it is suitable as a method for initializing a descent-type algorithm, which would improve our estimate, converging to a locally optimal solution. Also, in some situations the initial estimate produced by our algorithm may be sufficient without further refinement. In Section 5 we will evaluate the overall effectiveness of our algorithm for these purposes. However, in this section, we will compare our algorithm to other methods of obtaining an initial estimate of structure and motion. Primarily, we compare to the imputation method of Tomasi and Kanade, with some remarks about the implications of this comparison for initialization methods used by filtering algorithms.

One significant question is how well each method makes use of the information that is available to solve the structure-from-motion problem. We can show with a simple example that our method is using more of this information than the imputation method.

In Table 1 we show an example, in which nine out of twelve points are visible in each frame of a four frame motion sequence. For each of the six pairs of frames there are six points present. Using these, our algorithm derives a 6×5 submatrix (including a row of ones) with rank four, and a rank two nullspace. Overall, $\mathbf{N}$ has twelve columns, two for each pair of frames. These are not independent, and $\mathbf{N}$ will actaully have rank eight (proof omitted). However, the nullspace of $\mathbf{N}$ has rank four, which describes the 4-D linear subspace $\mathcal{L}$. Therefore, our algorithm derives a correct solution.

This problem cannot, however, be solved using the imputation method. Two frames can be used to derive the affine structure of six points, but only three of these points appear in any additional frame. Therefore, the projection of this affine structure into another frame cannot be determined. On the other hand, when any triple of frames is considered there are only three points present in all three. This is not enough information to determine affine strucutre or to determine rigid structure assuming scaled orthographic projection. This demonstrates that our algorithm is applying more of the available information in finding a solution. This may lead to more accurate results in the face of sensing error, even when both methods can find a solution.

We should note that our method also may not be making use of all available constraint. For example, a single column of the matrix places some constraints on $\mathcal{L}$, since not all linear subspaces will intersect the affine space that arises from all completions of a single column. However, our method uses only r-tuples of columns to constrain $\mathcal{L}$. Another way to think about the missing data problem is to note that the missing elements are filled in to produce a rank r matrix if and only if every $(r+1) \times (r+1)$ submatrix has determinant zero. This can give rise to both linear, and non-linear constraints on the solution. We conjecture that our method is essentially applying only the linear constraints that arise. If this is so, it means that more powerful solution methods are possible, but perhaps only by solving systems of non-linear equations.

We now present an experimental comparison between our method and the imputation method. In general, with the imputation method it is not clear how to choose the initial submatrix, and how to decide to which column or row to extend it. Therefore, we have chosen to evaluate the methods in a simple scenario, in which these decisions are straightforward.

We generate points in a synthetic motion sequence. In each sequence we select points randomly inside a cube. We then generate frames at uniform intervals as we rotate the object ninety degrees about the z-axis (in-plane rotation), and ninety degrees about a randomly chosen axis in the plane (out-of-plane rotation). We also translate the points by a total of half the width of the cube, in a random direction. In each frame, we introduce six new points, which will all be visible for a total of five frames. Table 2

Frame ↓ Pt. →	1	2	3	4	5	6	7	8	9	10	11	12	.	.	.
1	X	X	X	X	X	X	-	-	-	-	-	-			
2	X	X	X	X	X	X	X	X	X	X	X	X			
3	X	X	X	X	X	X	X	X	X	X	X	X			
4	X	X	X	X	X	X	X	X	X	X	X	X			
5	X	X	X	X	X	X	X	X	X	X	X	X			
6	-	-	-	-	-	-	X	X	X	X	X	X			
.	-	-	-	-	-	-	-	-	-	-	-	-	.		
.														.	
.															.

Table 2: Pattern of missing data used to compare our method to imputation method.

illustrates this pattern of missing data. The total number of frames is varied. As the number of frames increases, the problem becomes more difficult because each cohort of points is mutually visible over a smaller and smaller rotation, and so their structure can be less accurately determined. This provides a simple example of the type of sequence in which structure from motion algorithms are commonly applied. We then add Gaussian noise with a standard deviation of .25% of the maximum range of the elements. For a 500×500 pixel image this would be comparable to a standard deviation of 1.25 pixels.

With this pattern of missing data, it seems clear that the imputation method would be expected to begin with a fully occupied 5×6 submarix, and then extend to neighboring columns and rows. We tested two possibilities: beginning in the upper left corner and extending to the right and then downwards until the matrix was completed; and beginning in the middle of the matrix, and extending first to the right and down, then to the left and up.

Tables 3 and 4 compare the results of these two imputation methods to those of our algorithm. We show both the sum of squares error between the reconstructed matrix and the elements present in the original matrix, and also the error in the affine structure of the points that each algorithm produces. In both cases, the original Tomasi and Kanade algorithm, using SVD, can be applied to the filled in matrix to determine the points' structure. Since scene structure will be known only up to an affine transformation, we measure error after first finding the affine transformation that optimally aligns the reconstructed points with the true scene points.

Our algorithm produces results that are superior to those of the imputation method by a factor of between 7 and 100. This experiment also demonstrates the importance of correctly handling missing data. Our algorithm doesn't just do a better job of correctly

Num. Frames	NEW	IMPUTE TOP	IMPUTE MIDDLE
9	.000136/(.000002)	.004/(.001)	.00047/(.00003)
11	.000168/(.000002)	.011/(.002)	.008/(.001)
13	.00028/(.00004)	.011/(.002)	.011/(.001)
15	.0015/(.0008)	.016/(.002)	.014/(.001)
17	.0029/(.0008)	.024/(.002)	.018/(.001)

Table 3: Comparison between our algorithm and imputation methods. Synthetic motion sequences are generated as described in text. The number of frames used is shown in the first column. The second column shows the average of the sum of squared error between the reconstructed matrix and points present in data matrix for our algorithm. The third column shows results for the imputation method beginning in the top left of the data matrix. The fourth column shows results for imputation beginning in the middle. Results shown are median/(standard deviation), over 100 trials.

Num. Frames	NEW	IMPUTE TOP	IMPUTE MIDDLE
9	.000052/(.000001)	.004/(.001)	.0005/(.0001)
11	.00013/(.000005)	.014/(.002)	.011/(.002)
13	.00036/(.00004)	.017/(.002)	.021/(.003)
15	.0026/(.0009)	.029/(.003)	.034/(.003)
17	.007/(.002)	.053/(.004)	.053/(.004)

Table 4: Results as in Table 3. Here, average error is shown in reconstructing the affine structure of scene points.

"hallucinating" the missing points, it also does a better job of determining the true scene structure.

This experiment also suggests a possible comparison between our method and filtering algorithms. McLauchlan et al, for example, initialize their algorithm in exactly the same way as the imputation method that begins at the top of the matrix. Like the imputation method on this example, they then update their estimate as each point and frame is added. The methods of doing this are different, but for this example the imputation method that starts in the upper left corner of the observation matrix has something of the flavor of the filtering method of McLauchlan et al., and performs much less accurately than our algorithm. However, we have not implemented or experimented with their algorithm, so this observation should be treated only as a suggestion for a more detailed comparison.

Finally, Kahl and Hayden[9] solve the missing data problem by deriving constraints on the rotational part of the affine transformation and combining these. They describe the relationship between the constraints they derive and the ones we do stating: "[Jacobs method] can be seen as the 'dual' of [our] closure constraints...". They also point out that the method given for handling translation in [8] is incorrect. They then perform an experimental comparison between their method and the generic version of our algorithm. They apply our generic algorithm by estimating a rank four matrix, to capture rotation and translation. While this is a reasonable approach, the method we give above is preferable, since it makes explicit use of the fact that translation causes the row space of the error free matrix to contain a vector of all ones, and improves on our earlier generic method in other ways, described above. It is not clear how our new method would compare to that of Kahl and Heyden, although the methods seem closely related.

5 Experiments

In this section we describe experiments to test the accuracy of our proposed method. In these experiments we use our method as a starting point for iterative refinement to a locally optimal solution, which is done using the algorithm of Shum et al. This algorithm has the limitation that it does not allow for translation in the motion. We therefore use motion sequences in which translation has been removed, and then compare the results to those in which our algorithm is applied using sequences that are identical, but with the addition of translation. Note that in [8] we reported experiments on structure-from-motion using only the generic algorithm. The method described in this paper achieves considerably better performance.

In these experiments we compare eight different methods. The first (labeled NEW in graphs) is our algorithm for fitting a rank three matrix to a matrix with missing elements. When used to find the intensity manifold we randomly sample triples of columns from the matrix, to build the nullspace. When used in structure-from-motion, we consider all pairs of images in the sequence. NEW does not allow for translation in the motion sequence. The second method, NEWTRANS, is just like NEW, but estimates a rank four matrix in which one row is composed of all ones. NEWIT uses NEW to get an initial estimate of the solution, then refines this estimate using the algorithm of Shum et al. We must also measure the extent to which NEW contributes to NEWIT finding good solutions. We therefore compare NEWIT to iterative refinement using random starting points. RAN1 picks a random matrix as the starting point for Shum et al.'s algorithm. RAN3 repeats this process with three different random starting points, and chooses the solution with the least resulting error. RAN5 works analogously with five random starting points. We compare these results to (GT) an estimate of ground truth and (GT-IT) an estimate of the globally optimal solution (the one with smallest error) found by using the solution in (GT) as a starting point for the iterative method. While this cannot be guaranteed to converge to the globally optimal solution, it seems to provide a good estimate of the best obtainable solution. When using simulated data, we have ground truth available. For a real motion sequence, we use a solution based on all the data for (GT). The code used in all of our experiments is publicly available from the website: "http://www.neci.nj.nec.com/homepages/dwj/".

We evaluated methods in three ways. First, we have defined the goal of linear fitting as minimizing the sum of square differences between the elements present in a data matrix and the corresponding elements of a new, low rank matrix. For this reason, we evaluate methods based on how well they succeed in finding a low rank matrix to minimize this error measure. Second, we measure the accuracy of each method in reconstructing scene structure. Since this structure can be determined only up to a linear transformation, we first apply a linear transformation to the output that best fits it to ground truth, where available. Then we measure the deviation between the true structure, and the structure found by each algorithm. The emphasis of these experiments is on the problem of structure-from-motion, but we also include synthetic experiments that illustrate the use of the algorithm in finding the set of intensity images (the intensity manifold) that can be produced by a lambertian scene. In a few cases, our algorithm determined that it could not find a stable answer. Average errors are shown with these cases excluded, and the number of unstable cases shown separately. Third, we measure how often each iterative method succeeded in finding the best available solution. This is measured as the percentage of times each method produced an error within 10% of the smallest error found by any iterative algorithm.

A number of specific thresholds must be set in these experiments. We run the Shum et al. algorithm either for 100 iterations, or until it converges with ten decimal places of accuracy. In our algorithm, we select pairs of images or triples of columns until their nullspace occupies ten times as many columns as the original matrix (with a maximum limit of 1000 column triples, which in practice we hardly ever reach). In computing the nullspace to each triple of columns, we take to be zero any singular values less than .1. We compute L by taking the three least significant components of the matrix containing all the nullspaces to column triples; if the fourth smallest singular value of this matrix is less than .001 we determine that the result is unreliable, since there is not a well-defined, 3-D nullspace to the matrix.

We generate synthetic data in two settings. First, we generate points in a synthetic motion sequence. In each sequence we select twenty points randomly inside a cube. We then generate twenty frames at uniform intervals as we rotate the object ninety degrees about the z-axis (in-plane rotation), and ninety degrees about a randomly chosen axis in the plane. This provides a simple example of the type of sequence in which structure from motion algorithms are commonly applied. We then assume that each point is occluded for some fraction of the sequence. This fraction is randomly chosen for each point from a uniform distribution. We vary the expected value of the occlusion length to vary the number of missing elements in our experiments. The occluded frames for each point are randomly assigned to either the beginning of the sequence, the end of the sequence, or are split between the beginning and end, each with equal probability. We then add Gaussian noise with a standard deviation of .25% of the maximum range of the elements. For a 500×500 pixel image this would be comparable to a standard deviation of 1.25 pixels.

Next we generate random intensity images. First, we generate a scene containing 100 points, in which every scene point's surface normal is chosen with a uniform distribution on the unit half-sphere with positive z components. We select the albedo of each point from a uniform distribution from 0 to 1. We then select light sources, of unit intensity, and a direction that is drawn from the same distribution as the scene surface normals. Each light source combines with the scene to generate an intensity image, with intensities ranging from minus one to one. Gaussian noise is then added, with a standard deviation of .01. Finally, intensities less than zero were considered to be missing values.

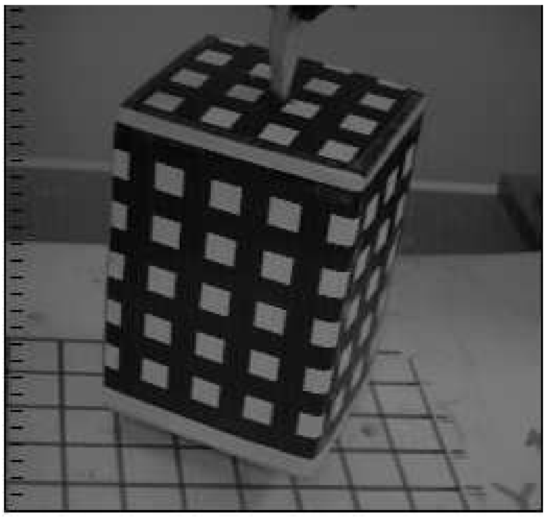


Figure 1: One frame of the real motion sequence used.

Algorithm	.2	.3	.4	.5	.6	.7
RAN1	.1325/(.0088)	.2743/(.0149)	.6942/(.0314)	1.052/(.0379)	1.630/(.0421)	2.336/(.0482)
RAN3	.0104/(.0037)	.0175/(.0056)	.0670/(.0137)	.2042/(.0246)	.5730/(.0337)	1.291/(.0610)
RAN5	.0003/(5e-6)	.0031/(.0025)	.0207/(.0058)	.0759/(.0111)	.3735/(.0263)	.9991/(.0462)
NEW	.0013/(2e-5)	.0029/(.0001)	.0491/(.0081)	.1299/(.0138)	.3292/(.0265)	.9326/(.0680)
NEWIT	.0003/(5e-6)	.0007/(2e-5)	.0228/(.0075)	.0926/(.0161)	.2686/(.0255)	.9048/(.0670)
NEWTRANS	.0012/(3e-5)	.0040/(.0007)	.0949/(.0117)	.2607/(.0184)	.5526/(.0261)	.8394/(.0352)
GTIT	.0003/(5e-6)	.0007/(2e-5)	.0077/(5e-5)	.0069/(.0004)	.0117/(.0006)	.0210/(.0012)

Table 5: This table shows the error in affine structure computed on synthetic motion sequences, for each algorithm and for missing data ranging from 20% to 70% of the data matrix. Results are shown as mean/(standard deviation).

Finally, we have run our algorithm on data obtained from a real motion sequence described in [15]. This sequence contained forty points tracked across eight images. One of the frames is shown in Figure 1. In this sequence we generated artificial occlusions exactly as described for the random motion sequences. We estimated the translation by measuring the translation of the center of mass of those points present in adjacent frames. Translation was removed before applying all methods except NEWTRANS. The matrix was then scaled to give it roughly the same sized elements as in the synthetic motion sequences, so that all the same parameters could be appropriately used.

Our new methods sometimes report that they were unable to find a stable solution. Table 6 indicates how often this occurs, as the fraction of matrix elements that are missing varies. It shows that the method begins to encounter problems only when 70% of the

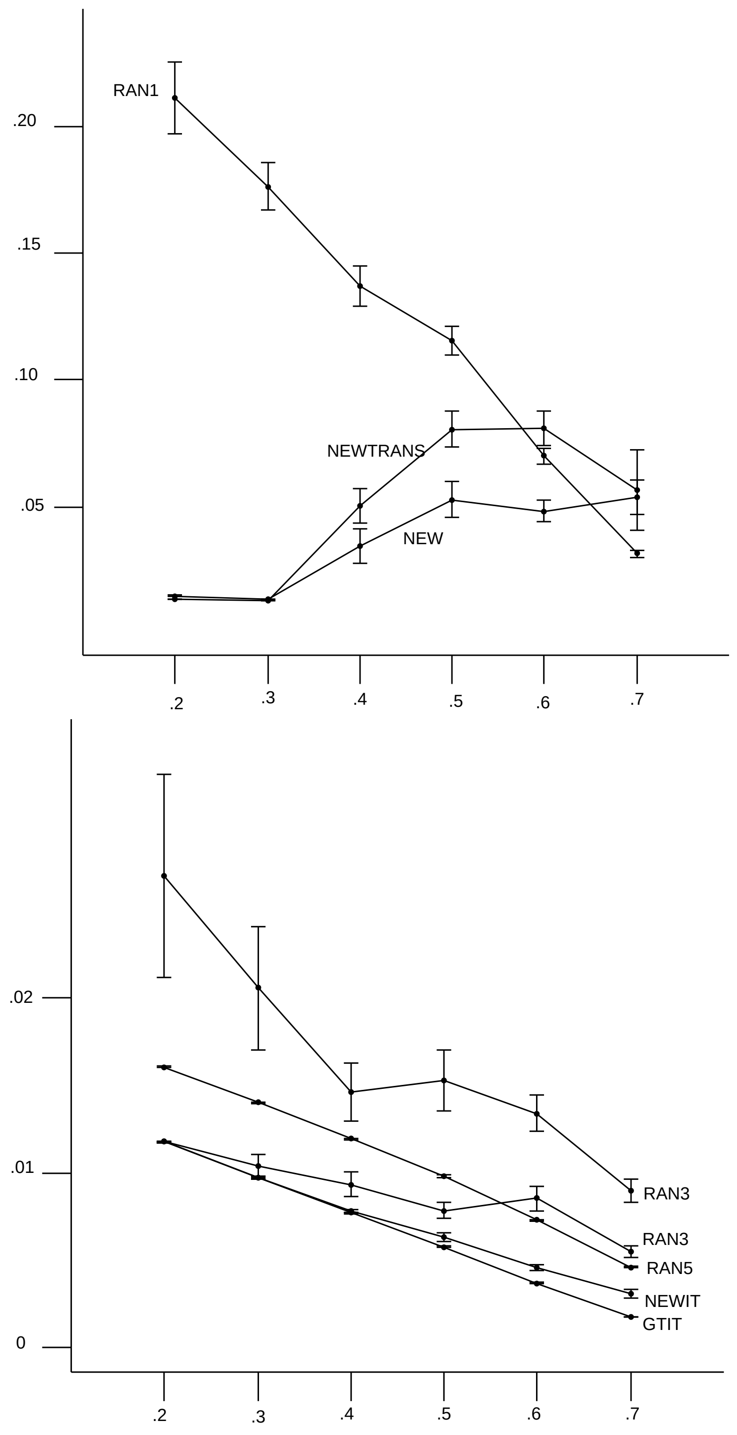


Figure 2: Structure-from-motion: The average error of six solution methods, with two comparison solutions. Each point shows the sum of square errors between the data and a rank three fit, averaged over 500 trials. Error bars show one standard deviation in the data. Average error is plotted against the average fraction of frames, f, in which a point is occluded. The top plot shows three solutions. On the bottom, we plot methods that produce less error.

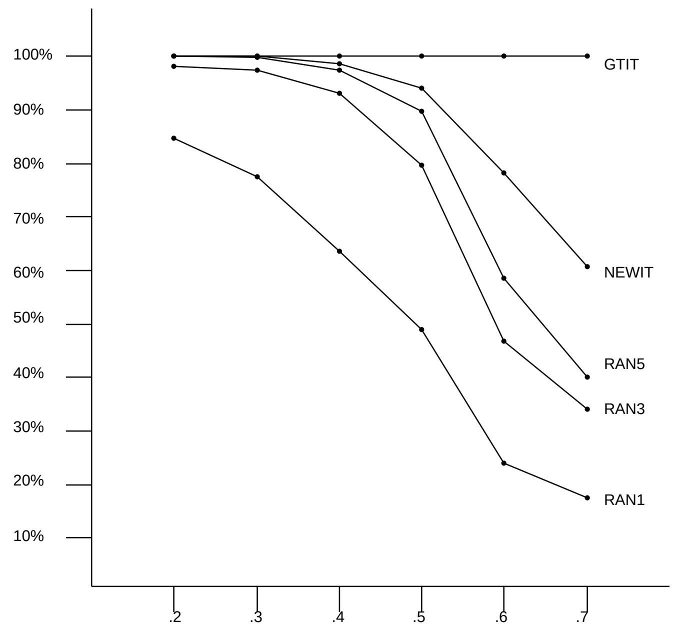


Figure 3: Synthetic structure-from-motion: The fraction of times each iterative method converged to within 10% of the best available solution found by any method.

Occlusion Level	.2	.3	.4	.5	.6	.7
NEW Percentage Stable, synthetic	100	100	100	100	100	92.8
NEWTRANS Percentage Stable, synthetic	100	100	99.8	100	99.6	89.2
NEW Percentage Stable, real	100	100	100	100	100	96
NEWTRANS Percentage Stable,	100	100	100	100	100	96

Table 6: The percentage of trials of synthetic motion sequences in which the new algorithm found a stable solution.

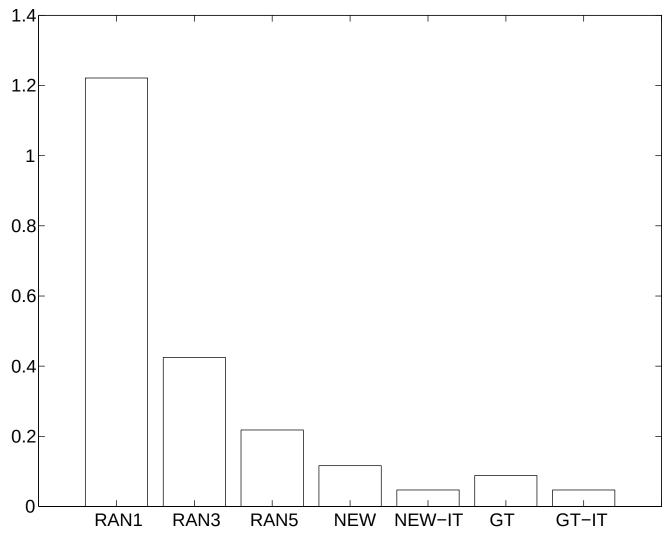


Figure 4: Average residual error produced in experiments fitting a rank three matrix to intensity images with missing data, averaged over 100 trials. RAN1, RAN3 and RAN5 show the iterative method with one, three and five random starting points, respectively. NEW shows our method. NEW-IT shows our method used as a starting point for an iterative solution. For comparison, GT shows ground truth, and GT-IT shows ground truth used as a starting point for an iterative solution.

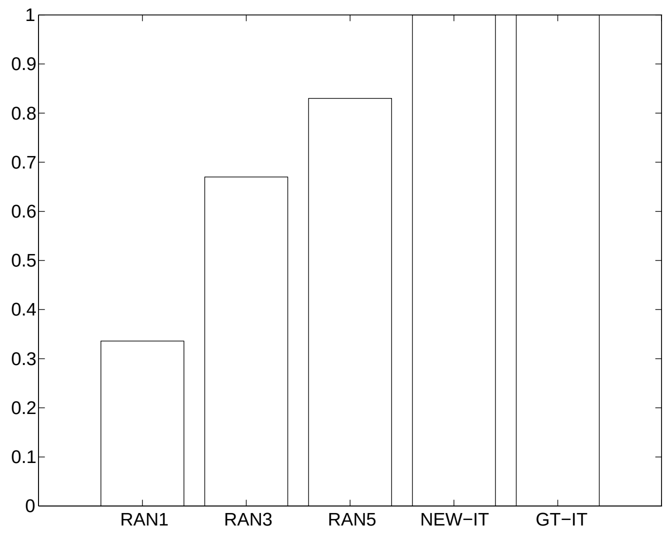
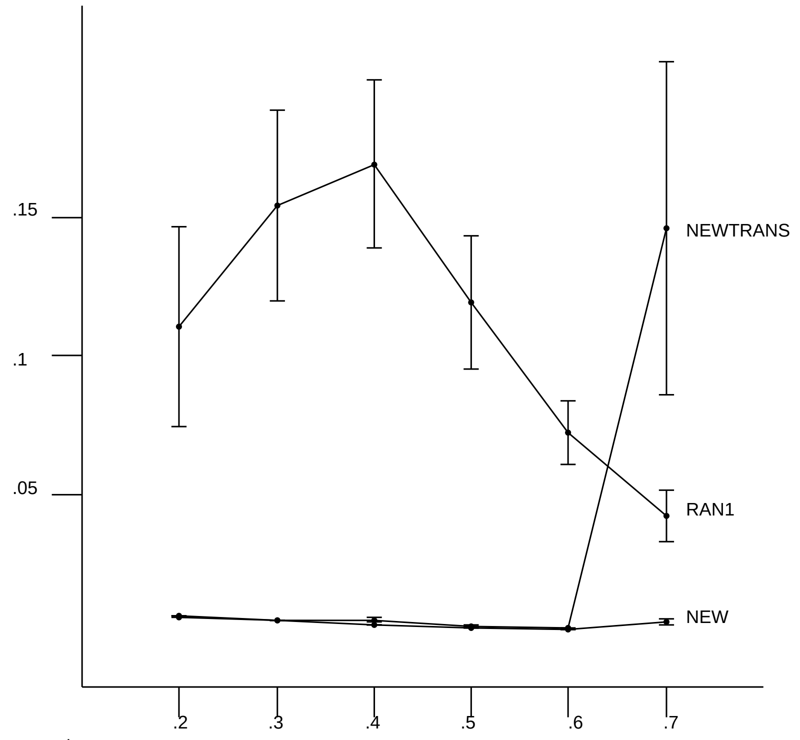


Figure 5: Here we show how often the iterative methods converged to the best available solution for intensity images. RAN1, RAN3 and RAN5 show the iterative method with one, three and five random starting points, respectively. NEW-IT shows our method used as a starting point for an iterative solution, and for comparison, GT-IT shows the iterative solution with ground truth as a starting point.



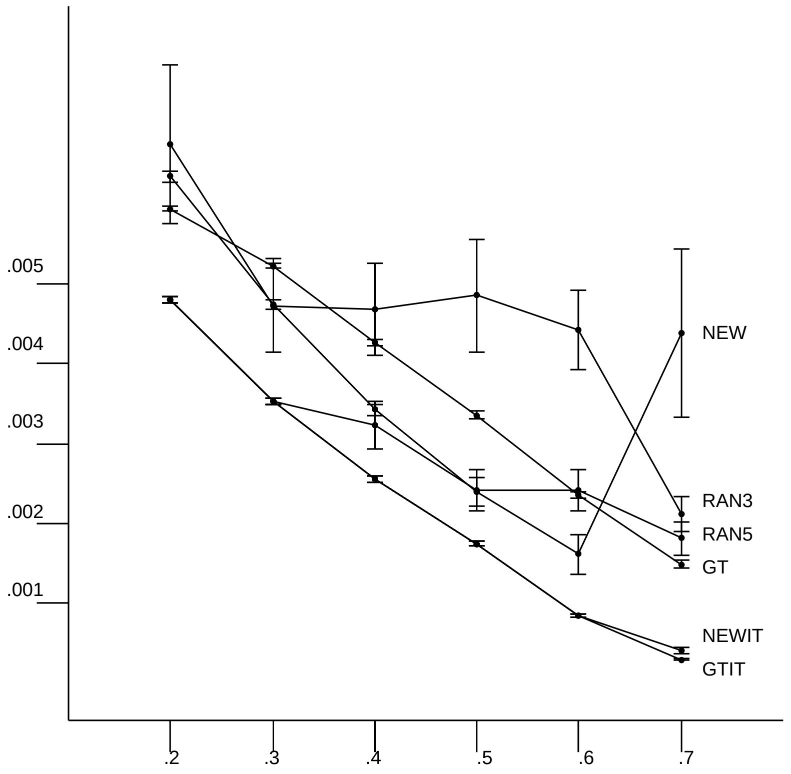


Figure 6: Structure-from-motion, a real sequence: Average errors for the real motion sequence, displayed as in Figure 2.

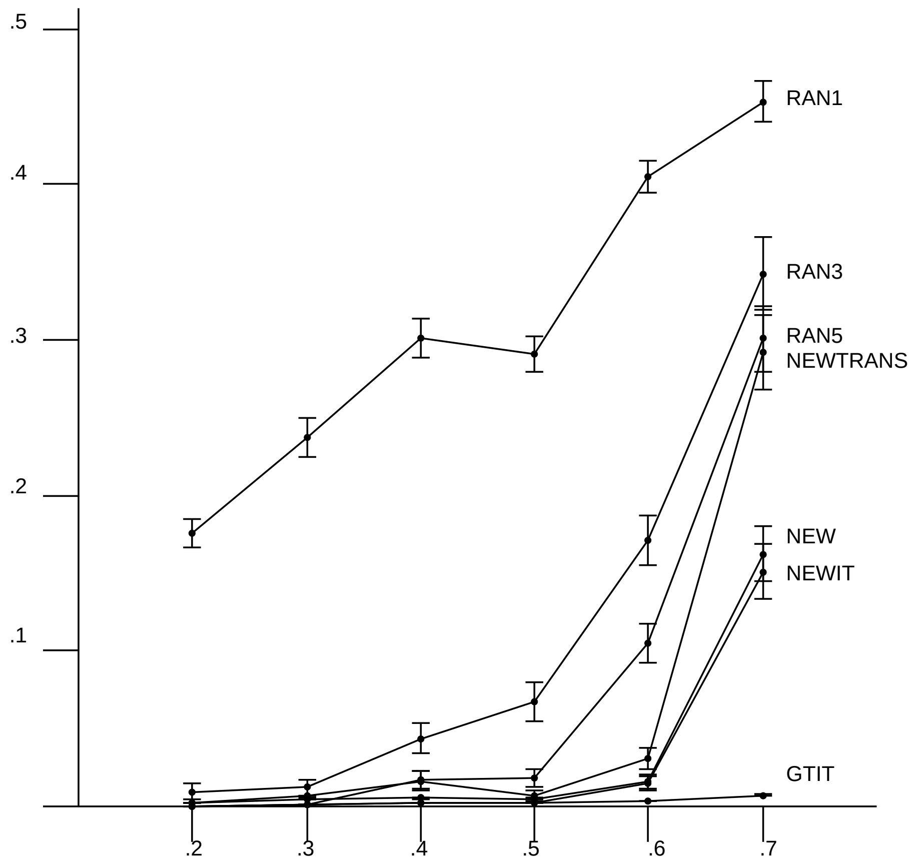


Figure 7: Structure-from-motion, a real sequence: Average errors in determining the affine structure for the real motion sequence.

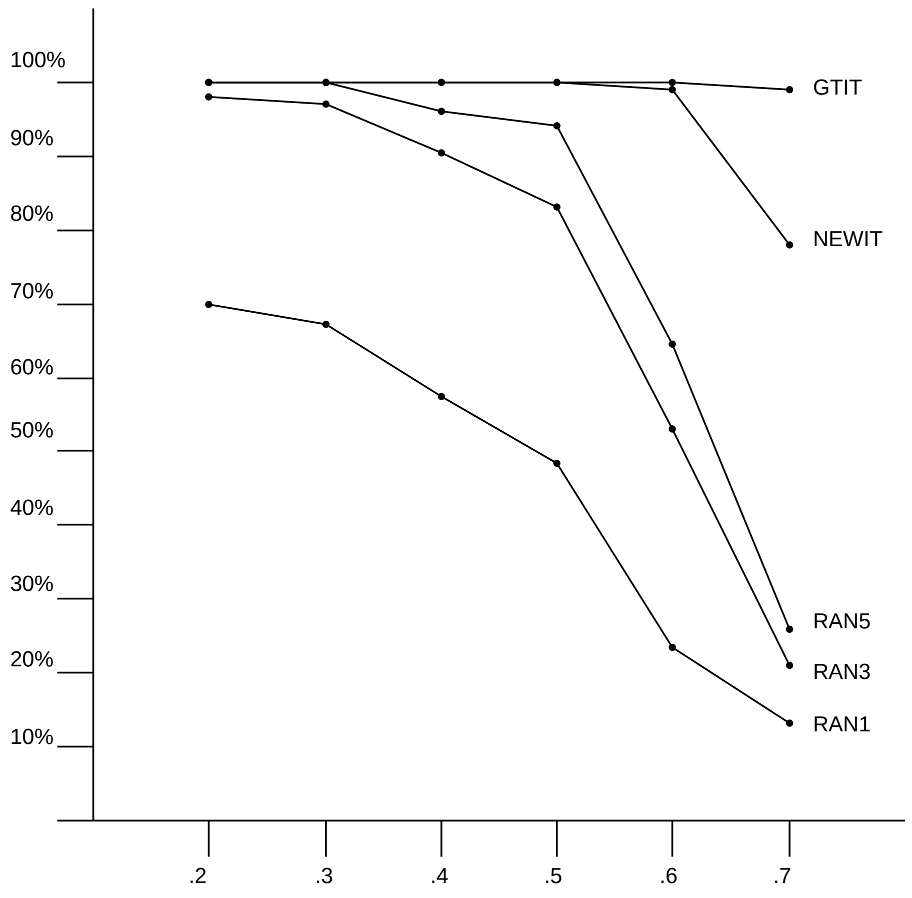


Figure 8: The fraction of times each iterative method converged to within 10% of the best available solution found by any method, for the real motion sequence.

matrix elements are missing.

Beginning with the synthetic sequences, Figure 2 shows the sum of squares error between the data matrix elements that are present, and the fitted matrix, for each solution method. Table 5 shows the errors derived in the affine structure. The range of values made them difficult to display in a graph. Figure 3 shows how often each iterative method produced the best solution of any method. Figure 4 shows the sum of squares error for the synthetic intensity images, and Figure 5 shows how often each iterative method found the best solution. These are shown for all trials. And Figures 6, 7 and 8 show similar results for the real motion sequence.

The first point to notice is that our new method produces good results, especially as a starting point for an iterative solution. In the case of synthetic and real motion, our method followed by an iterative solution produces significantly smaller errors than the best solution found using five random starting points. Our method also finds the globally optimal solution much more frequently than we do with five random starting points. In fitting linear surfaces to intensity images, our method produces essentially perfect results, even though this problem seems to be quite difficult when a random starting point is used. This suggests that our method produces an effective starting point for finding the correct solution, one that is much better than random.

Second, although we do not test NEWTRANS as a starting point for iterative refinement, we can see that it produces results approximately as good as those found by NEW.

Third, we can see that when there are significant amounts of data missing, the motion problem can be quite difficult. When 70% of the data are missing, even our new method makes a significant number of errors. The difficulty of a motion problem will certainly depend not just on the number of missing data, but also on the particular patterns of occlusion and on the magnitudes of motion and the number of frames. However, we feel that our experiments are based on a reasonable scenario, and that even more difficult motion situations, with less motion and more occlusion, are of interest. Therefore, our simulations indicate that real problems of interest may present a difficult challenge for any existing algorithm, if an optimal solution is desired with very high probability.

6 Conclusions

Several vision problems have recently been treated as linear fitting problems. This is especially attractive, because this is a linear optimization problem that can be efficiently solved using linear methods. The most notable example of this is the affine structure-from-motion algorithm of Tomasi and Kanade. However, while the main virtue of the Tomasi and Kanade algorithm is that it can integrate information using an extended motion sequence, in such sequences it is quite likely that many points will appear and disappear. This makes finding the optimal solution a non-linear optimization problem.

The main contribution of this paper is to find a linear method that approximates the solution to this non-linear problem. We also show how to specialize this solution for to determine structure-from-motion. The result of this method can then be used as the starting point for an iterative method that finds a locally optimal solution. We show theoretically that our approach can handle situations with less available data than other methods. Empirically we show that this method is superior to other proposed methods of initializing iterative algorithms, and describe experiment measure the overall performance of our algorithm, primarily for the problem of structure-from-motion.

Acknowledgment

Liana Lorigo provided significant assistance with our implementation and experiments, and suggested some useful improvements to the algorithm. I'd also like to thank John Oliensis for helpful discussions, and Ronen Basri for helpful discussions, and in particular, for pointing out the connection between the missing data problem in structure from motion, and in the problem of characterizing the intensity images of a scene. Anonymous reviewers also made several useful suggestions.

References

[1] Basri, R., 1996. "Paraperspective ≡ Affine," *Int. J. of Comp. Vis.*, **19**(2):169-179.

[2] Basri, R. and Ullman, S. 1988. "The Alignment of Objects with Smooth Surfaces". *Computer Graphics, Vision, and Image Processing: Image Understanding*, **57**(3):331–345.

[3] Belhumeur, P. and Kriegman, D., 1996. "What is the Set of Images of an Object Under All Possible Lighting Conditions?", *IEEE Conf. on Comp. Vis. and Pat. Rec.*:270–277.

[4] Demmel, J., 1987, "The Smallest Perturbation of a Submatrix Which Lowers the Rank and Constrained Total Least Squares Problems," *SIAM J. Numer. Anal.***24**(1):199–206.

[5] Golub, G. and Van Loan, C., 1996, *Matrix Computations, 3rd Edition*, Johns Hopkins Press, Baltimore, MD.

[6] Hartley, R., 1993, “Euclidean Reconstruction from Uncalibrated Views,” *Proc. 2nd European-US Workshop on Invariance*:237–256.

[7] Hayakawa, H., 1994, “Photometric stereo under a light source with arbitrary motion,” *Journal of the Optical Society of America*, **11**(11): 3079–3089.

[8] Jacobs, D., 1997, “Linear Fitting with Missing Data: Applications to Structure-from-Motion and to Characterizing Intensity Images,” *IEEE Conf. on Comp. Vision and Pat. Rec.*:206-212.

[9] Kahl, F. and Heyden, A., 1999, “Affine Structure and Motion from Points, Lines and Conics,” *International Journal of Computer Vision*, **33**(3):163-180.

[10] Little, R. and Rubin, D., 1987. *Statistical Analysis with Missing Data*, John Wiley and Sons.

[11] McLauchlan, P., Reid, I., and Murray, D., 1994, “Recursive Affine Structure and Motion from Image Sequences,” *Eur. Conf. on Comp. Vision*:217–224.

[12] Moses, Y., 1993. *Face recognition: generalization to novel images*, Ph.D. Thesis, Weizmann Institute of Science.

[13] Poelman, C. and Kanade, T., 1997, “A Paraperspective Factorization Method for Shape and Motion Recovery,” *IEEE Trans. PAMI*, **19**(3):206–219.

[14] Poljak, S. and Rohn, J., 1993, “Checking Robust Nonsingularity is NP-Hard,” *Math. Control and Signals Systems*, **6**:1–9.

[15] Sawhney, H., Oliensis, J. and Hanson, A., 1990. “Description and Reconstruction from Image Trajectories of Rotational Motion”, *Int. Conf. on Comp. Vis.*:494–498.

[16] Shashua, A., 1992. *Geometry and Photometry in 3D Visual Recognition*, MIT TR–1401.

[17] Shum, H., Ikeuchi, K., and Reddy, R., 1995. “Principal Component Analysis with Missing Data and Its Applications to Polyhedral Object Modeling”, *IEEE Trans. PAMI*, **17**(9):854-867.

[18] Soatto, S. and Perona, P., 1995. “Dynamic Rigid Motion Estimation from Weak Perspective”, *Int. Conf. on Comp. Vis.*:321–328.

[19] Sugihara, A., 1996, “Object Recognition by Combining Paraperspective Images,” *Int. J. of Comp. Vis.*, **19**(2):181-201.

[20] Tomasi, C. and T. Kanade, 1992, “Shape and Motion from Image Streams under Orthography: a Factorization Method,” *Int. J. of Comp. Vis.*, **9**(2):137-154.

[21] Ullman, S. and Basri, R., 1991, “Recognition by Linear Combinations of Models,” *IEEE Trans. PAMI*, **13**(10):992-1007.

[22] Wiberg, T., 1976. “Computation of Principal Components when Data are Missing”, *Proc. Second Symp. Computational Statistics*:229–236.