Investigation into Matrix Factorization when Elements are Unknown

Technical Report

Aeron Buchanan Morgan
Visual Geometry Group
Department of Engineering Science
University of Oxford

30th November 2004
Abstract

The problem of low-rank matrix factorization has seen significant attention in recent computer vision research. Problems that use factorization to find solutions include structure from motion, non-rigid object tracking and illumination based reconstructions. Matrix decomposition algorithms, such as singular value decomposition, can be used to obtain the factorizations when all the input data are known, reliably finding the global minimum of a certain cost function. However, in practice, missing data leads to incomplete matrices that prevent the application of standard factorization algorithms. To date, many algorithms have been proposed to deal with the missing data problem. This report presents the results of an investigation into these algorithms and discusses their effectiveness. It is seen that they rarely find the global minimum. Newton based methods, which have not been previously applied to this problem, are investigated and shown to find the global minimum more reliably, but do not fulfil the expectations one may have of optimization routines. However, it is argued that they are more easily extended to overcome the shortfalls of the basic approach than the other algorithms reviewed. Furthermore, the suitability of the global minimum as a solution is covered, creating more avenues of investigation for the improvement of factorization schemes. Future research looking into such extensions is described with the aim of engineering a successful algorithm.
Acknowledgements

I would very much like to thank Andrew Fitzgibbon for being so enthusiastic and patient and David Capel for his notes on Parametric Optimization [9].
**Contents**

1 **Introduction**  
   1.1 Example Problems .................................................. 5  
   1.2 Matrix Factorization ................................................ 7  
   1.3 Notation ..................................................................... 8  
   1.4 Applications ............................................................. 9  
   1.5 The Error Function .................................................... 14  

2 **Existing Solutions** ....................................................... 18  
   2.1 Direct Methods .......................................................... 18  
   2.2 Alternation Approaches .............................................. 26  
   2.3 Other Literature ......................................................... 30  

3 **Error Surface Exploration** ........................................... 32  
   3.1 Gradient Descent ....................................................... 32  
   3.2 Newton .................................................................... 33  
   3.3 Damped Newton ........................................................ 34  
   3.4 Damped Newton with Line Search ................................. 34  
   3.5 Hybrid System .......................................................... 35  

4 **Results** ..................................................................... 38  
   4.1 Synthetic Performance Comparison ............................... 38  
   4.2 Real Problems ........................................................... 43  
   4.3 Priors .................................................................... 51  

5 **Conclusions** ............................................................... 52  

A **An Alternation Derivation** ........................................... 53  

B **Higher Order Terms** .................................................... 55
Problem: for partially known $\mathbf{M}$, find $\mathbf{A}$ and $\mathbf{B}$ such that

$$\mathbf{M} = \mathbf{AB}^\top$$

In practice:

$$\min_{\mathbf{A},\mathbf{B}} \| \mathbf{W} \odot (\hat{\mathbf{M}} - \mathbf{AB}^\top) \| \quad \text{assuming} \quad \hat{\mathbf{M}} = \mathbf{M} + \mathbf{N}(0, \sigma^2)$$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\mathbf{M}}$</td>
<td>The acquired measurement matrix. This is the only variable (other than $\mathbf{W}$) whose elements are known at all. It is modelled as a noisy version of $\mathbf{M}$.</td>
</tr>
<tr>
<td>$\mathbf{W}$</td>
<td>The weight matrix. A matrix whose elements describe the confidence in the knowledge of the corresponding elements of $\hat{\mathbf{M}}$.</td>
</tr>
<tr>
<td>$\mathbf{M}$</td>
<td>The rank $r$ matrix to be factorized.</td>
</tr>
<tr>
<td>$\mathbf{A}$</td>
<td>A columnspace for $\mathbf{M}$.</td>
</tr>
<tr>
<td>$\mathbf{B}$</td>
<td>A rowspace for $\mathbf{M}$.</td>
</tr>
<tr>
<td>$m$</td>
<td>The number of rows in $\mathbf{M}$, $\hat{\mathbf{M}}$, $\mathbf{W}$ and $\mathbf{A}$.</td>
</tr>
<tr>
<td>$n$</td>
<td>The number of columns in $\mathbf{M}$, $\hat{\mathbf{M}}$ &amp; $\mathbf{W}$ and the number of rows in $\mathbf{B}$.</td>
</tr>
<tr>
<td>$r$</td>
<td>The rank of $\mathbf{M}$ and the number of columns in $\mathbf{A}$ and $\mathbf{B}$.</td>
</tr>
<tr>
<td>$m_{ij}$</td>
<td>The element of $\mathbf{M}$ on the $i^{\text{th}}$ row and $j^{\text{th}}$ column.</td>
</tr>
<tr>
<td>$\mathbf{m}_i$</td>
<td>The $i^{\text{th}}$ row of $\mathbf{M}$ as a column vector.</td>
</tr>
<tr>
<td>$\mathbf{m}_j$</td>
<td>The $j^{\text{th}}$ column of $\mathbf{M}$.</td>
</tr>
<tr>
<td>$\mathbf{M}_{(m \times n)}$</td>
<td>Dimensions of vectors and matrices are given in sub scripted parentheses.</td>
</tr>
<tr>
<td>$\delta_{ij}$</td>
<td>A Kronecker delta function equalling one when $i = j$ and zero otherwise.</td>
</tr>
<tr>
<td>$\odot$</td>
<td>The Hadamard element-wise matrix product: $\mathbf{R} = \mathbf{P} \odot \mathbf{Q} \implies r_{ij} = p_{ij} q_{ij}$</td>
</tr>
<tr>
<td>$\mathbf{M}^+$</td>
<td>The pseudo-inverse of $\mathbf{M}$ allowing $\mathbf{M}^+ \mathbf{M} = \mathbf{I}$ for $r = n &lt; m$.</td>
</tr>
<tr>
<td>$\mathbf{M}_{</td>
<td>r}$</td>
</tr>
<tr>
<td>$\mathbf{M}(:,)$</td>
<td>Column-wise vectorization of $\mathbf{M}$: $\mathbf{M}(:,) = (\mathbf{m}_1^\top \mathbf{m}_2^\top \ldots \mathbf{m}_m^\top)^\top$</td>
</tr>
<tr>
<td>$\mathbf{M}(\cdot)$</td>
<td>Row-wise vectorization of $\mathbf{M}$: $\mathbf{M}(\cdot) = (\mathbf{m}_1^\top \mathbf{m}_2^\top \ldots \mathbf{m}_m^\top)^\top$</td>
</tr>
</tbody>
</table>

Figure 1: Summary of notation and operator symbols used in this report.
Chapter 1

Introduction

Matrix factorization is a powerful tool for finding solutions to several central computer vision problems. These include *structure from motion* [33, 38], *object illumination* [5] and *non-rigid model tracking* [8]. To cement the motivation for investigating factorization, these applications are described in detail later in this chapter. It will become clear that a major hurdle for computer vision in all these applications is missing data. Standard factorization algorithms (e.g. *singular value decomposition*) cannot provide solutions when data are missing. If we want to overcome real-life challenges, the ability to cope with missing data must be part of the algorithms we intend to use. To date, there have been many algorithms introduced to provide a solution. However, none manage to consistently provide either good performance, satisfactory solutions or both. An actual example of each of the three applications mentioned above is introduced in the next section to illustrate the problems in a more real sense. It is then important to review the mathematics that is common to all the applications that are covered. Section 1.2 sets the mathematical scene and comments on some of the finer points of the problems to ensure that the important aspects of the problem can be differentiated from those which are secondary and so are not considered in this report. Chapter 2 presents the many algorithms that have been proposed in the literature. Newton based methods are presented next, being a class of algorithms that have not yet been explored for this problem. A comparison of all these schemes is given in Chapter 4 using extensive synthetic tests and the real example problems introduced in this chapter. Finally, the conclusions summarize the report and describe the direction of further research.
1.1 Example Problems

The rest of this chapter will provide the mathematical details for these examples, so here is a more qualitative description of possible applications for a factorization algorithm.

Figure 1.1 shows selected frames from three real image sequences. The first is an example of a structure from motion (SFM) problem. It is a toy dinosaur rotating on a turntable. It is a rigid scene seen from many angles. The sequence is given to tracking software that finds visible features in the scene and tracks the image of each feature as it moves around in image-space. A measurement matrix is filled by the lists of the 2D image coordinates of each feature. Impressively, both the structure of the scene (the 3D coordinates of the tracked features on the dinosaur) and the position and orientation of all the cameras can be obtained from the measurement matrix alone. Holes in the measurement matrix arise because not every feature is physically visible in every frame and because the tracking software is not completely reliable and fails to find a visible feature in some frames, so SFM is a missing data problem in practice.

The second sequence is an example of a non-rigid scene. It is non-rigid on two levels; at the scene level (where there are three independently moving objects, namely the effectively rigid scenery and two giraffes) and at the object level (the giraffes are very much deforming objects). The segmentation of the three objects (i.e. the scene level non-rigidity) is not considered in this report and is left for future work (see Chapter 5). Instead, the background giraffe is considered in isolation. Again, tracking software is used to generate the data that is used to create the measurement matrix. The reasons for missing data in the non-rigid reconstruction problem are the same as for the rigid situation. Here the occlusions are very clear in the first frame shown where the giraffe is behind a bush and in the second frame shown where the foreground giraffe walks in front.

The third sequence demonstrates the setup for illumination based reconstructions. A static scene is lit by a distant light source from different directions. During the filming, the movement of the light source was controlled so that the distance between the bulb and the face was very much larger than the size of the face, thus approximating the requirement of it being an infinite distance. In this case the measurement matrix is generated directly from the intensity values of the images. Shadows and specular highlights have to be omitted from the measurement matrix because such visual events are not part of the diffuse model (as will be explained in Section 1.4.4 on page 13).
Figure 1.1: Three sequences demonstrating actual problems requiring matrix factorization. Top row: a toy dinosaur rotates on a turntable (a sequence of 36 image frames). Middle row: a background giraffe walks out from behind a bush and is momentarily occluded by a giraffe in the foreground walking in front of it (120 frames). Bottom row: a static scene is illuminated from many directions (20 frames).
1.2 Matrix Factorization

The general problem being addressed in this report is matrix factorization. Given a matrix, $M$, of known rank, $r$, find two smaller matrices, $A$ and $B$, such that:

$$M_{(m \times n)} = A_{(m \times r)}B^T_{(n \times r)}$$  \hspace{1cm} (1.1)

The previous section described examples of real problems. The process common to the applications is as follows; an imaging device records one or more images of the real world in a digital form. These images are processed and the entries of $M$ are filled. The solution sought is an $A$ and $B$ that satisfy Equation 1.1. In practice there are four significant aspects that make a solution to Equation 1.1 difficult:

1. There is not a unique answer to any such problem, as any invertible $r \times r$ matrix, $G$, may always be introduced as a transform on the factoring matrices:

$$M = (AG)(G^{-1}B^T) = A' B'^T$$  \hspace{1cm} (1.2)

Here, $G$ represents the gauge freedom [39]. It turns out that this is not actually as inconvenient as it may seem. In most applications it can either be disregarded or easily circumnavigated, e.g. by registration or normalization.

2. $M$ is not known exactly. Instead, a noisy version of $M$ is acquired. For convenience, the differences between the known noisy elements and the supposed true values are assumed to be independently drawn from a particular distribution. Although no one distribution may be used in general, the algorithms reviewed in this report all assume that a zero-mean Gaussian distribution is valid. The algorithms I propose in Chapter 3 are not constrained to a particular distribution, but using the Normal distribution we have:

$$\hat{M} = M + N(0, \sigma^2)$$  \hspace{1cm} (1.3)

$\hat{M}$ is the measurement matrix. It is so likely that $\hat{M}$ will have a rank greater than $r$ that the alternatives are ignored. Because the expectation, $<\hat{M}>$, is $M$, the problem to be solved becomes that of finding the rank $r$ matrix that is closest to $\hat{M}$, i.e.

$$\min_{A,B} \left\| \hat{M} - AB^T \right\|$$  \hspace{1cm} (1.4)

When the Frobenius or 2-norm is taken as a distance measure, singular value decomposition gives the correct answer (see Golub and Van Loan [12], page 72 and Reid and Murray [26]).

3. For various application-specific reasons (see previous section), not all the elements of $\hat{M}$ are known. Any minimization scheme employed must be adapted to ignore the elements of $\hat{M}$ that do not have values. Here, we introduce a weight matrix, $W$, whose elements reflect the knowledge of the corresponding elements of $\hat{M}$. Zeros in $W$ mean that those positions in $\hat{M}$ do not have values and ones mean they do. Values between zero and one may also be
used, denoting relative confidences in the values in $\hat{M}$. The matrix $\hat{w}$ can be included in the minimization thus:

$$\min_{\hat{A},\hat{B}} \| \hat{w} \odot (\hat{M} - \hat{A}\hat{B}^T) \|$$  \hspace{1cm} (1.5)

This problem cannot be solved using standard factorization techniques (such as singular value decomposition).

4. In real applications, it cannot be guaranteed that all the data entered into the measurement matrix $\hat{M}$ is consistent with the model. Maybe because the model does not fully explain all observed situations (such as shadows and specularities in the illuminated face example) or because the process that generates the elements of $\hat{M}$ from an image sequence is not completely reliable (e.g. feature tracking software for SFM problems sometimes present a track that is actually jumping between more than one similar looking feature as the path of a single feature point). Such data are called outliers and can significantly mislead algorithms, encouraging them to return erroneous results. Algorithms that are not distracted by outliers and in some way manage to give correct solutions despite their presence are called robust algorithms. None of the missing data algorithms reviewed in this report are robust and outlier rejection is left for future investigation.

In Equation 1.5 we have the focus of this report. It is used by all the applications described and as such, all the algorithms included in this report, which have been introduced as ways to tackle this formula, may be used for finding solutions to any of the forms of the problem.

### 1.3 Notation

In the literature there are many labels used for the components of Equation 1.5. In this report, the same variable names will be used throughout, aiding direct comparisons of the approaches that have been proposed over the years. Specifically, the measurement matrix, $\hat{M}$, is the only matrix with known elements—unknown elements being signified by the corresponding zero elements of the weight matrix, $\hat{w}$—and is assumed to be a noisy version of $M$. The noise on each element is taken as being independently drawn from a zero-mean Normal distribution, with variance $\sigma^2$. The desired factoring matrices are denoted by $A$ and $B$. $M$ is known to be rank $r$ and so, given $M \in \mathbb{R}^{m \times n}$, we have $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{n \times r}$ with $\hat{w}, \hat{M} \in \mathbb{R}^{m \times n}$. The weight matrix will mostly be considered as a matrix of zeros and ones, although arbitrary element values on the interval $(0, 1)$ can be considered. Also note that $M$, without loss of generality (see Section 2.1), is always considered to be portrait, with $m > n$.

More generally, scalars are italicized ($s$), vectors are bold ($v$) and matrices are presented in the 'typewriter' typeface ($M$). Different styles of the same letter will always refer to the same matrix, namely: $m_{ij}$ is the element on the $i^{th}$ row and in the $j^{th}$ column, $\hat{m}_j$ the $j^{th}$ column and $\hat{m}_i^\top$ the $i^{th}$ row of matrix $\hat{M}$. Subscript and superscript indices will also be used to represent column and row numbers, respectively, for block segmentation.
of matrices. Vectors should always be taken as column vectors, with row vectors being represented using transpose notation.

The symbol \( \odot \) signifies the Hadamard element-wise product of two matrices, so

\[
R = P \odot Q \quad \implies \quad r_{ij} = p_{ij}q_{ij}
\]

The rank truncation of a matrix, \( M \), to a lower rank \( r \) (that is, the closest rank \( r \) matrix to \( M \)), is denoted with \( M \downarrow r \), i.e. \( \tilde{M} = M \downarrow r \) such that \( \|M - \tilde{M}\| \) is minimized. This can be achieved by taking the singular value decomposition (SVD) of \( M \) as \( U\Sigma V^\top \), then zeroing out all the diagonal entries of \( \Sigma \) beyond the \( r^{th} \) to give \( \tilde{\Sigma} \) and finally rebuilding the matrix:

\[
M \downarrow r = U\tilde{\Sigma}V^\top.
\]

For the row-wise vectorization of \( M \), \( M(\cdot) \) is used, i.e. \( (m^1\top \, m^2\top \, \ldots \, m^m\top)^\top \).

Finally, the pseudo-inverse of a matrix \( M \) is represented by \( M^+ \). It is used when \( M \) is rectangular \((m \neq n)\) and an actual inverse does not exist. If \( M \) is square and full rank then \( M^+ = M^{-1} \), whereas if \( r = n < m \) then \( M^+ = (M^\top M)^{-1}M^\top \). In general, the pseudo-inverse of a rank \( r \) matrix \( M \) may be constructed using the SVD thus:

\[
M^+ = \Sigma^+\tilde{U}^\top \quad \text{where} \quad \Sigma^+ = \text{diag}\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right)
\]

with \( \sigma_i \) being the \( i^{th} \) singular value.

A summary of this section is provided for reference as a separate sheet with this document.

### 1.4 Applications

With the general formulation described above, it is now easy to explain how the computer vision applications mentioned at the start of the chapter can be thought of so as to make factorization a method for solutions. The rest of this section goes through the models used in each application.

#### 1.4.1 Structure from Motion

The term ‘structure from motion’ refers to the process of recovering scene structure (the world positions of 3D points and cameras) from the motion of the 2D projections of the points through an image sequence. It is this problem that is given the most attention in this report, mainly because it is the one which is treated most frequently in the literature. Tomasi and Kanade [33] presented factorization as a way of solving the SFM problem. The basis is the affine camera model, which describes cameras as orthographically projecting scene points onto an image plane. For \( F \) views (or frames) of \( P \) points, the equation of projection is

\[
m_f^p = P_f^p x_p + t_f^p \quad f = 1, \ldots, F; p = 1, \ldots, P
\]

with \( m_f^p \) being the two-dimensional vector encoding the \( f^{th} \) image of the \( p^{th} \) point, \( P_f^p \) a \( 2 \times 3 \) matrix describing the \( f^{th} \) camera’s orientation and scaling effects, \( t_f^p \) the translation.
vector moving the image of the world coordinate origin to the $f^{th}$ frame’s origin and $x_p$ being the three-vector for the global position of the $p^{th}$ point. The equations for all the points in one view may be concatenated to give

$$\begin{bmatrix} m_1^f & m_2^f & \cdots & m_p^f \end{bmatrix} = p^f \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix} + t^f 1^T \tag{1.9}$$

or

$$M^f = p^f x^f + t^f 1^T \tag{1.10}$$

where $1$ is a $P$-vector of ones. Indeed, all the views may be concatenated in the same way for a conveniently compact matrix form of the model:

$$\begin{bmatrix} M^1 \\ M^2 \\ \vdots \\ M^F \end{bmatrix} = \begin{bmatrix} p^1 \\ p^2 \\ \vdots \\ p^F \end{bmatrix} X + \begin{bmatrix} t^1 \\ t^2 \\ \vdots \\ t^F \end{bmatrix} 1^T \tag{1.11}$$

or

$$M = PX + t 1^T \tag{1.12}$$

which, in turn, may be written as a single matrix multiplication, putting the problem into the form addressed in this report:

$$M = \begin{bmatrix} P & t \end{bmatrix} \begin{bmatrix} X \\ 1^T \end{bmatrix} = AB^T \tag{1.13}$$

The measurement matrix, $M$, collects all of the coordinates of all the images of all the points. It is the known component of Equation 1.13, obtained from the image sequence [32]. Under the affine camera model, it is a rank 4 matrix giving $r = 4$, $m = 2F$ and $n = P$.

The inclusion of the vector $t$ and hence the constraint that the last column of $B$ should be all ones is inconvenient. Here, we can turn to the gauge freedom for help. Consider a matrix $G$ of the form:

$$G = \begin{bmatrix} I \\ 0^T \\ -v \\ 1 \end{bmatrix} \tag{1.14}$$

to be used in Equation 1.13 to give the following transformation:

$$M = \begin{bmatrix} P & t \end{bmatrix} \begin{bmatrix} I \\ 0^T \\ -v \\ 1 \end{bmatrix} \begin{bmatrix} I \\ 0^T \\ v \\ 1 \end{bmatrix} \begin{bmatrix} X \\ 1^T \end{bmatrix} = \begin{bmatrix} P & -Pv + t \end{bmatrix} \begin{bmatrix} X + v 1^T \\ 1^T \end{bmatrix} \tag{1.15}$$

or

$$M = \begin{bmatrix} P & t' \end{bmatrix} \begin{bmatrix} X' \\ 1^T \end{bmatrix} \tag{1.16}$$

Using the new factorization, let us look at the row-wise summation of the elements of $M$; that is, the column sum of $M$.

$$\sum_{j=1}^{n} m_j = \sum_{j=1}^{n} Px' + t' 1 = nt' + P \sum_{j=1}^{n} x'_j \tag{1.18}$$
We can eliminate the first term of the right-hand side of Equation 1.18 if we can make \( \sum_{j=1}^{n} x_j' \) equal zero.

\[
\sum_{j=1}^{n} x_j' = 0 \quad \Rightarrow \quad \sum_{j=1}^{n} x_j + v = n v + \sum_{j=1}^{n} x_j = 0 \quad (1.19)
\]

\[
\text{or} \quad v = \frac{1}{n} \sum_{j=1}^{n} x_j \quad (1.20)
\]

So, the gauge freedom of the problem can be used to set the origin of the 3D world points to be at their centroid. Now, following on from Equation 1.18, we have

\[
\sum_{j=1}^{n} m_j = n t' \quad \text{or} \quad t' = \frac{1}{n} \sum_{j=1}^{n} m_j \quad (1.21)
\]

This says that the image of the centroid of a set of points is the centroid of the images of all the points, bringing us to the operation of “mean-centring” the measurement matrix:

\[
\bar{M} = M - \bar{m} j^T \quad \text{with} \quad \bar{m} = \frac{1}{n} \sum_{j=1}^{n} m_j \quad (1.22)
\]

Equation 1.13 is affine projection in homogeneous coordinates and highlights the fact that \( \bar{M} \) is actually a three-dimensional structure embedded in a four-dimensional space, i.e. an offset hyperplane. By mean-centring the measurement matrix, the hyperplane is translated to the origin and \( \bar{M} \) becomes rank 3, with no constraints:

\[
\bar{M} = p' \chi' \quad (1.23)
\]

Sadly, manipulating \( M \) to be mean-centred is an option that is not available when elements of \( M \) are unknown and so affine SFM with missing data remains the larger rank 4 problem in Equation 1.13. The way minimization is affected by this situation is discussed in the next section (Section 1.5).

As mentioned above, with all three of the reconstruction from motion applications mentioned here, the practicalities of missing data is all too apparent when it comes to their implementations. When filling \( \bar{M} \) from the coordinates of tracked scene points, occlusions or failures in the tracking process leave elements for which values are not known, sometimes making the measurement matrix very sparse.

### 1.4.2 Non-Rigid Structure from Motion

Extending this model to account for non-rigid motion simply adds another layer of factorization. Assume that non-rigid objects can change shape by being a weighted sum of a set of \( B \) basis shapes, \( S^n \), which are \( 3 \times P \) matrices holding the positions all the 3D points (in order) for each shape. So, for frame \( f \), we have:

\[
x_f' = \lambda_1^f S^1 + \lambda_2^f S^2 + \ldots + \lambda_B^f S^B = \sum_{i=1}^{B} \lambda_i^f S^i \quad (1.24)
\]
and so
\[ M' = P' X' = \sum_{i=1}^{B} \lambda_i^f P^i S^i \] (1.25)

leading to the points for all frames to be written in one expression:
\[
M_{(2F \times P)} = \begin{bmatrix}
\lambda_1^1 P^1 & \lambda_2^1 P^1 & \cdots & \lambda_B^1 P^1 \\
\lambda_1^2 P^2 & \lambda_2^2 P^2 & \cdots & \lambda_B^2 P^2 \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_1^F P^F & \lambda_2^F P^F & \cdots & \lambda_B^F P^F \\
\end{bmatrix}
\begin{bmatrix}
S^1 \\
S^2 \\
\vdots \\
S^B \\
\end{bmatrix}
\] (2F×3B)
\[
= AB^\top
\] (1.27)
i.e. the measurement matrix is a rank 3B matrix, with the special feature that A has a very specific structure (two-row sub-blocks can be reordered into rank 1 matrices). See Bregler, Hertzmann and Biermann [8] for more details.

### 1.4.3 Structure from Perspective Motion

The perspective camera model is essentially the perspective version of Equation 1.8, i.e. using homogeneous coordinates in a projective space:
\[
\lambda^f m^f_{p(3 \times 1)} = p^f_{(3 \times 4)} x_{p(4 \times 1)} \quad f = 1, \ldots, F; p = 1, \ldots, P
\] (1.28)

where \( \lambda \) is an arbitrary scale factor introduced to deal with the lack of uniqueness inherent with representing coordinates using homogeneous vectors. As before, the equations for all the points and all the frames can be stacked into one matrix equation:
\[
\begin{bmatrix}
\lambda_1^1 m_1^1 & \lambda_2^1 m_1^1 & \cdots & \lambda_P^1 m_1^1 \\
\lambda_1^2 m_2^1 & \lambda_2^2 m_2^1 & \cdots & \lambda_P^2 m_2^1 \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_1^F m_F^1 & \lambda_2^F m_F^1 & \cdots & \lambda_P^F m_F^1 \\
\end{bmatrix}
= \begin{bmatrix}
p^1 \\
p^2 \\
\vdots \\
p^F \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_P \\
\end{bmatrix}
\] (1.29)
\[
= AB^\top
\] (1.30)
The addition of equality up to scale is the crucial factor that sets this problem apart from the simple affine model. Only two-thirds of the measurement matrix is known here, as the image coordinates of the points only give the inhomogeneous part, and not the scale factor. Having said that, we can actually choose \( F + P \) of them arbitrarily, but the rest must be calculated. The calculation of the projective depths proceeds by operating on the epipolar geometry of frame pairs [38]. Once the measurement matrix is complete, it is known to be rank 4 and so can be factorized. This model is mentioned here for completeness and will not be discussed further in this report.
1.4.4 Surface from Illumination

Another problem that can be formulated as a single matrix product is that of recovering surface normals from a series of varied illumination images. Considering only diffuse illumination, the light reflected from a flat surface with normal $\mathbf{n}$ from a light source at infinity in direction $\mathbf{d}$ is simply the dot-product of the two vectors. Therefore, given a set of $M$ images of a diffuse object under varied lighting conditions (with the camera’s orientation relative to the object constant throughout) we can take the object’s surface as being made up of planar segments—one under each pixel—to build up a measurement matrix by entering all the pixel intensity values from each image along each row. Each column is therefore the intensity of a single point on the surface over all illumination directions. Each element of the measurement matrix, thus each pixel (i.e. surface point), is calculated as

$$m_{ij} = \mathbf{d}_i^\top \mathbf{n}_j \quad \forall ij$$

and concatenating these together in the usual way gives

$$M_{(M \times N)} = \begin{bmatrix} \mathbf{d}_1^\top \\ \mathbf{d}_2^\top \\ \vdots \\ \mathbf{d}_F^\top \end{bmatrix} \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \cdots \\ \mathbf{n}_P \end{bmatrix}_{(3 \times N)}$$

$$= \mathbf{AB}^\top$$

i.e. the factorization of the measurement matrix gives the illumination direction for every frame and surface normal for every image pixel.

The lack of perfectly diffuse surfaces in the real world means that implementations join the ranks of missing data problems. It was introduced earlier that elements of the measurement matrix must be ignored when a pixel’s colour value shows the surface to be exhibiting reflectance outside the diffuse model. This is covered by values below a low threshold representing a surface going into shadow and values above a high threshold representing the reflectance becoming specular. Helpfully, specularities are often small and a moving light source, which is inherent to this application, means that both specularities and shadows will move across the surface leaving a good proportion of the measurement matrix with data for processing.
Figure 1.2: A cross-section through a real error surface showing three minima. It was generated by evaluating the error function using the data from the dinosaur sequence. The error is a function of many more than two variables so this cross-section was chosen by picking three minima found by the damped Newton algorithm (see Chapter 3). Analysis confirms that these are minima in all dimensions.

1.5 The Error Function

\[ F(A, B) = \| W \odot (\hat{M} - AB^\top) \|_F^2 \quad (1.34) \]

When all of the elements in \( \hat{M} \) are known and \( W \) contains only ones, the error function has one minimum and that is found by the SVD. Nonlinear minimization algorithms also quickly find the global minimum. When data are missing, only iterative approaches can provide solutions. To make their job difficult, the weighted error function, \( F \), has many minima (see Figure 1.2). By introducing the weight matrix, the error surface is drastically changed. As well as multiple minima, the global minimum in the full data case is, in general, no longer a minimum in the missing data case. Not only must algorithms deal with local minima, they must also incorporate useful regularization to get closer to the answer that would be obtained if all data were available.

Looking further into the space in which \( F \) is to be minimized, we have the following insight. Let

\[ \mathbf{x} = \left( \begin{array}{c} A(\cdot) \\ B(\cdot) \end{array} \right) \quad (1.35) \]

be the row-wise unravelling of both \( A \) and \( B \) into an \( r(m+n) \)-vector. The error can now be written in vector form relating directly to the \( r(m+n) \)-dimensional space of the error.
Here, \( S_{ij} \) is the \( ij^{th} \) \( r(m+n) \)-square selection matrix, forming the appropriate dot-product between the rows of \( A \) and \( B \). It has a very simple structure:

\[
S_{ij} = \begin{bmatrix}
0_{mr \times mr} & 0_{r \times r} & \cdots & 0_{r \times r} \\
0_{r \times r} & \frac{1}{2} I_{r \times r} & \cdots & 0_{r \times r} \\
\vdots & \vdots & \ddots & \vdots \\
0_{r \times r} & 0_{r \times r} & \cdots & 0_{nr \times nr}
\end{bmatrix}
\]  

(1.37)

that is, \( S_{ij} \) is a matrix of all zeros, except for a half identity matrix in the \( ij^{th} \) \( r \times r \) block of the upper right segment and another in the \( ji^{th} \) block of the lower left segment. The halves have been introduced to make \( S_{ij} \) symmetric. Looking at a cross-section through this error surface, from a point \( p \) and in the direction \( q \), by substituting \( x \) with \( p + tq \), we can see that the error surface can be quartic:

\[
F(t) = \sum_{i,j=1}^{m,n} w_{ij}^2 \left( \hat{m}_{ij} - (p + tq)^\top S_{ij}(p + tq) \right)^2
\]  

(1.38)

\[
= \sum_{i,j=1}^{m,n} w_{ij}^2 \left( \kappa_{0ij} + \kappa_{1ij} t + \kappa_{2ij} t^2 + \kappa_{3ij} t^3 + \kappa_{4ij} t^4 \right)
\]  

(1.39)

with

\[
\kappa_{0ij} = \left( \hat{m}_{ij} - p^\top S_{ij} p \right)^2
\]  

(1.40)

\[
\kappa_{1ij} = 4 \left( p^\top S_{ij} p - \hat{m}_{ij} \right) p^\top S_{ij} q
\]  

(1.41)

\[
\kappa_{2ij} = 2 \left( p^\top S_{ij} p - \hat{m}_{ij} \right) q^\top S_{ij} q + 4 \left( p^\top S_{ij} q \right)^2
\]  

(1.42)

\[
\kappa_{3ij} = 4 \left( p^\top S_{ij} q \right) q^\top S_{ij} q
\]  

(1.43)

\[
\kappa_{4ij} = \left( q^\top S_{ij} q \right)^2
\]  

(1.44)

noting that \( p^\top S_{ij} q = q^\top S_{ij} p \). If the direction \( q \) is chosen to be a coordinate direction (i.e. a vector of zeros except for one entry set to one), \( q^\top S_{ij} q \) is zero and the error function is a sum of quadratics. This is used by alternation to form closed-form solutions as will be seen in Section 2.2. However, directions can be chosen to reveal the quartic nature of the surface (this is similar to the fact that a ruled quadric can look linear in cross-section).
Figure 1.3: A graph of the missing-data error as it is minimized by the algorithm described in Algorithm 3.5 from 100 random starting points. The red horizontal line is the error of the answer given by the SVD of the full measurement matrix, i.e. the answer we want. The iterative scheme has found five minima, none of which are a satisfactory answer. Although three of the minima appear to be better than the SVD solution, it must be emphasised that the error function is only looking at a subset of residuals and so can favour solutions that are poor when compared to the full input. This test was performed on random synthetic data.

It may be helpful to consider the situation in an algebraic light. Taking the columns of \( M \) to be points in an \( m \)-dimensional space, factorization is the process that fits an \( r \)-dimensional hyperplane through the origin to those points, minimizing the squared perpendicular distance from the plane to the points (i.e. minimizing Equation 1.4). If the points are thought to lie on an \((r - 1)\)-dimensional hyperplane, that does not include the origin, factorization can provide the answer in that situation as well. As a brief example, here is the least-squares error function for fitting a line to a set of 3D points:

\[
F(d, \alpha, v) = \left\| X - \left( d\alpha^T + v1^T \right) \right\|^2_F
\]

where \( X \) holds all the points to be fitted as columns, \( d \) is the direction of the line, \( \alpha \) holds all the distances along the line of the nearest points on the line to each point in \( X \) and \( v \) is the offset of the line from the origin. The \( \alpha_s \)s in this case would be measured from \( v \) in ‘lengths-of-\( d \)’. As with the SFM formulation, we may stack \( d \) and \( v \) into one matrix
and put $\alpha$ and 1 into another to give the form of the function in Equation 1.4. Hence, constraining the last column of $B$ to be all ones, finds the least squares solution to fitting an offset $(r - 1)$-dimensional hyperplane. Note that this is the same performing principal component analysis on a set of points without the origin as their centroid (the mean point), i.e. the centroid must be found in the minimization along with the principal components.

When the weight matrix is included, it is like a window that the minimization looks through. Seeing only a proportion of the residual matrix means that it can do better than the theoretical minimum. Figure 1.3 shows an example of an iterative algorithm falling into local minima. It is an example of the situation where the solution that is sought (the solution given by the SVD of the full measurement matrix) has a higher error than solutions attained by iterative schemes. Because the error function being minimized by the iterative scheme only looks at a subset of elements from the residual matrix (the difference between the input, $M$, and a pair of proposed factors, $A$ and $B$), it has a distorted view of how well it is doing, hence the seemingly superior results. In practical situations there is no way of knowing how much the solution given by the missing-data minimization is away from the ‘true’ answer.
Chapter 2

Existing Solutions

Here is a review of the existing solutions to the problem of the decomposition of incomplete matrices.

2.1 Direct Methods

A space can be described by a span of vectors, \( \{ s_1, s_2, \ldots, s_r \} \), linear combinations of which form all points in that space. If all the vectors in the span are linearly independent then it is a minimal span, or basis, for the space. That space may exist, for convenience, in more dimensions than the number of vectors needed to describe it, i.e. as \( r \) \( m \)-vectors with \( m > r \), and so can be thought of as a subspace. An \( r \)-dimensional subspace in an \( m \)-dimensional space may be represented in matrix form by an \( m \times r \) matrix with the basis vectors as columns, denoted here by \( S \). Now, a rank \( r \) matrix, \( M \), with \( r < \min(m, n) \) is rank deficient and has linearly dependent rows and columns. Therefore, the columns of such a matrix can be taken as points in an \( r \)-dimensional subspace. The span of that subspace is called the column space of the matrix. Each matrix column can be described with a set of coefficients, or coordinates:

\[
m_j = s_1 \lambda_{1j} + s_2 \lambda_{2j} + \ldots + s_r \lambda_{rj} = S \lambda_j \quad \forall j
\]  

Congenating all these equations together horizontally gives the factorization equation with which we are now familiar:

\[
M = [m_1 \ m_2 \ \cdots \ m_n] = S [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_m] = SA \equiv AB^T
\]  

The order of the stacking does not effect the calculation of any one column’s coefficients, drawing attention to invariance of factorization to column ordering. Indeed, any column-wise operations may be performed prior to factorization and the process is still valid. Undoing those operations on the matrix afterwards will always give a valid decomposition of the original matrix. Transposing \( M \) is also a valid pre-factorization event. That is, given \( M = AB^T \), \( M' = A'B'^T \) and \( M' = M^T \), then \( A' = B \) and \( B' = A \). As such, the terms ‘row’ and ‘column’ may be interchanged in these explanations (if done consistently) and we can always assume that \( m > n \), i.e. that \( M \) is portrait.
2.1.1 A Special Case Solution

The idea of row and column spaces and them being coefficients for each other in the construction of their matrix, immediately suggests a way of filling a specific group of matrices with missing elements: those that have \( r \) complete (i.e. all elements are known), linearly independent columns. Let the matrix with missing elements be \( \mathbf{M} \in \mathbb{R}^{m \times n} \) and be rank deficient, i.e. \( r < \min(m, n) \). Let the complete columns be \( \{ s_{j'} \} \), for \( j' = 1 \ldots r \). Let \( \mathbf{S} \) be an \( m \times r \) matrix with all these \( s_{j'} \) vectors as its columns. \( \mathbf{S} \) is, of course, a valid column space of the matrix \( \mathbf{M} \). All other columns are linear combinations of the columns of \( \mathbf{S} \). Take \( \mathbf{m}_j \) as one of the columns with missing elements. If at least \( r \) elements of \( \mathbf{m}_j \) are known, then we have at least \( r \) equations of the form

\[
s_{i'1}\lambda_1 + s_{i'2}\lambda_2 + \ldots + s_{i'r}\lambda_r = m_{i'j} \quad \forall i'
\]

for \( i' \) the indices of the known elements of \( \mathbf{m}_j \). By creating an \( r \)-vector from these known elements, \( \hat{\mathbf{m}}_j \), the following equation can be formed to determine the coefficients for this column:

\[
\hat{\mathbf{S}}(r \times r)\lambda(r \times 1) = \hat{\mathbf{m}}_j(r \times 1)
\]

with \( \hat{\mathbf{S}} \) constructed by taking the rows of \( \mathbf{S} \) corresponding to those elements taken from \( \mathbf{m} \). As \( \hat{\mathbf{S}} \) has full rank, this equation can be solved for the coefficients, \( \lambda \), and the missing elements can be found directly:

\[
\mathbf{m}_j = \mathbf{S}\hat{\mathbf{S}}^{-1}\hat{\mathbf{m}}_j
\]

Based on the above, an algorithm for rebuilding a rank-deficient matrix with missing elements, hence a factorization for it, can be formed. It is given in Algorithm 2.1. Of course, this algorithm will only work on matrices for which \( r \) full columns are known. Despite this being a rare occurrence in practice, Rother and Carlsson [27] had sequences in which a ‘groundplane’ was visible in all views and so satisfied the condition and motivated their algorithm based on this rowspace idea using the concept of infinite homographies.

It should be clear that the minimum number of elements that must be known for complete factorization is \( r(m + n - r) \), which is illustrated by an example in Figure 2.1. We need an \( m \times r \) element column space and \( n \) sets of \( r \) coefficients to describe all the columns. If
Algorithm 2.1 Factoring a matrix with missing elements in all but $r$ columns.

**inputs** $M \in \mathbb{R}^{m \times n}, W \in \{0, 1\}^{m \times n}, (m, n, r) \in \mathbb{Z}$

$M$ is rank $r$ and has unknown values in all but $r$ linearly independent columns

1: declare $v \in \mathbb{R}^{n-r}$, $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{n \times r}$
2: $c \leftarrow 1$
3: $col \leftarrow 1$
4: while $(col - c < n - r)$ do // fill $A$ with $r$ complete columns
5: if $\sum(w_{col}) = m$ then // column $m_{col}$ is complete
6: $a_{c} \leftarrow m_{col}$
7: $b^{col} \leftarrow 0$
8: $b_{col,c} \leftarrow 1$
9: $c \leftarrow c + 1$
10: end if // record column as having missing elements
11: $v_{col-c+1} \leftarrow col$
12: end if
13: $col \leftarrow col + 1$
14: end while
15: for $i = 1$ to $n - r$ do // process the incomplete columns
16: $j \leftarrow v_{i}$
17: declare $\hat{m} \in \mathbb{R}^{r}$, $\hat{S} \in \mathbb{R}^{r \times r}$
18: $c \leftarrow 1$
19: $row \leftarrow 1$
20: while $(c < r) \& (row < m)$ do // take $r$ known elements of column
21: if $w_{col,j} = 1$ then // element is known
22: $\tilde{m}_{c} \leftarrow m_{row,j}$
23: $\hat{S}^{c} \leftarrow S^{row}$
24: $c \leftarrow c + 1$
25: end if
26: $row \leftarrow row + 1$
27: end while
28: if $c=r$ then // $r$ known elements found
29: $b^{j} \leftarrow \hat{S}^{-1}\tilde{m}$
30: else
31: cannot reconstruct this column
32: end if
33: end for

**outputs** $A, B$
the column space is taken directly from the columns of the matrix itself, then $r$ sets of $r$ coefficients are already known to be the identity matrix and so can be subtracted from the count (this is the gauge freedom).

### 2.1.2 A General Solution

In general, to determine a specific unknown element of $m_j$, $S$ need not be formed from complete columns; if the indices of $r$ known elements in $m_j$ are again $i'$ and the index of an unknown element is $i$, then any columns of $M$ with known values in the $i^{th}$, plus all the $i^{th}$ entries, can be used to determine $m_{ij}$. The algorithm is similar to Algorithm 2.1, but rather than starting with a search for an all encompassing column space, instead there are searches for partial column spaces in the inner loop. An outline is given in Algorithm 2.2.

**Algorithm 2.2** Rebuilding a matrix with missing elements.

**inputs** $M \in \mathbb{R}^{m \times n}$, $W \in \{0, 1\}^{m \times n}$, $(m, n, r) \in \mathbb{Z}$

- $M$ is rank $r$ and has at least $r(m + n - r)$ known elements - see text.

1: repeat
2: for $j = 1$ to $n$ do // each column
3: 
4: 
5: 
6: 
7: 
8: for all $i \in \{i\}$ do // each unknown element
9: 
10: if size$(\{j\}) \geq r$ then
11: 
12: 
13: 
14: end if
15: end for
16: end for
17: until matrix full

**outputs** $M$

It should now be easy to see that $M$ must have at least $r$ elements known on every column (otherwise deficient columns can not have their coefficients calculated) and at least $r$ elements on every row (otherwise no other elements on that row can be determined). Obviously, these minimum conditions are necessary, but not sufficient. The configuration of missing elements is the crucial factor. Having exactly the minimum number of known values can be achieved with any matrix of the form shown in Figure 2.1, but it is easy to create matrices with these minimum properties that will fail (see Figure 2.2). Helpfully, the larger the matrix size to rank ratio, the larger the number of elements that can be missing
without loss of information. This is particularly good in a SFM context because it tends to be easy to make the number of frames a in video sequences much larger than the rank constraint of the scene (unless one is after excessively non-rigid structure). The bad side of this view is that increasing the number of frames doesn’t necessarily increase the number of tracked feature points and computers only have finite resources and are unable to deal with arbitrarily large computations.

Why is Algorithm 2.2 not the solution we are looking for? The reason is noise. Noisy measurements make it much harder to guess the original matrix, especially when only a proportion of the matrix entries are known. At this stage, there are two main strategies which help deal with noisy matrix entries. The first is using as many elements as possible when calculating each column’s coefficients, rather than only the $r(r+2)$ that are required when the data is noise-free. The pseudo-inverse is an invaluable tool in implementations of the first strategy. The second helps the first by suggesting that newly calculated entries should also be used to calculate further coefficients. Note that Algorithm 2.2 has been presented in a way that incorporates both these strategies. Algorithm 2.2 is, in fact, a generalized version of an algorithm included in Tomasi and Kanade’s original paper [33]. Their implementation only uses the second strategy, but it does explicitly deal with non-mean-centred measurement matrices. An initialization method in a paper by Guerreiro and Aguiar [14], which can be described as columnspace stitching, is an alternative, but still equivalent, implementation. Algorithm 2.2 has, very recently, been proposed again by Chen and Suter [10]. A perspective version has been suggested by Saito and Kamijima [29], although it is more philosophically similar than algorithmically equivalent.

### 2.1.3 Error Magnification

One illustration of how having holes in the measurement matrix makes reconstruction, and hence factorization, hard was presented in Section 1.5. In the context of Algorithm 2.2, the difficulty can be seen as error propagation. Errors in the reconstruction of the first parts of the matrix are magnified when those parts are used in the calculations for the rest of the matrix. Successive mini-columnspaces are adjusted to align with those previously extracted to build up the full columnspace. Each slightly noisy columnspace stitch adds in
successively more error. In the full data case, the SVD uses all elements to get an idea of the noise and so deals with it in an optimal fashion. For the noisy missing data case, strategy one above is a step towards this. Strategy two tries to help, but ends up propagating errors. Unfortunately measurement matrices generated from real-life data (an example is shown in Figure 2.3) often have structures closer to that shown in Figure 2.4 rather than the one in Figure 2.1 and the second strategy becomes a necessity.

### 2.1.4 Jacobs’ Reconstruction Method

An alternative approach has been introduced by Jacobs [20]. Rather than effectively building up a column space, Jacobs suggests working down to one from the full $m$-dimensional space in which the column space exists. Jacobs shows that an orthogonal complement of $S \equiv \mathbb{A}$ can be incrementally built up until it describes a subspace of the desired size, i.e. a span of $(m - r)$ $m$-vectors.

The orthogonal complement of a matrix $S$ is defined as

$$N = \text{null}(S^\top)$$

It is the span of all the vectors that are orthogonal to the span of the vectors in $S$, i.e. $S^\top N = 0$.

Jacobs’ algorithm uses orthogonal complements to carve away at $\mathbb{R}^m$ to find the $r$-dimensional subspace from which the columns of $M$ have been taken. As the algorithm progresses, $N$ is built up. The final step is to set $S = \text{null}(N^\top)$. An advantage here is that when dealing with a noisy measurement matrix, an excess of columns may be added to $N$ by using as many elements from $M$ as possible, so that all those values can be used in the
Figure 2.4: Rank 2 reconstruction example. Let filled be known and open unknown. The circled unknown entry can not be reconstructed directly as there are no other columns with known entries on the 2nd, 7th and 8th rows. However, there are enough known elements to rebuild the matrix and hence factorize it. The 9th element of every 3 x 3 block with 8 known elements can be calculated, creating more 3 x 3 blocks with 8 known entries. This recursive filling is implemented in Algorithm 2.2. It is fine for the noise-free situation, but errors propagate in the noisy matrix and become ever larger towards the two unknown corners. Note that this structure has the required r(m + n − r) known elements.

calculation of S. The final step would then be a minimization step rather than an actual null space calculation, but it is easily implemented with the SVD.

See Algorithm 2.3 for an overview of the general process. For a noise-free M, take r columns of M to form a matrix Si. If all the elements of Si are known then it is a valid column space and the process can terminate. If they are not, then the algorithm proceeds as follows. Firstly, note and remove all the rows of Si in which there is an unknown element to form ̂Si. Calculate ̂Ni, the orthogonal complement of ̂Si. Form Ni by inserting into ̂Ni zero rows corresponding to those that were removed from Si. Concatenate Ni onto ̂N. If N is the required size, i.e. has at least (m − r) columns, and is rank r, then set S = null(N⊤), otherwise continue. Every span, Ni, that is added to N is orthogonal to S and so when N is complete it exactly defines S.

The procedure of truncating Si and then rebuilding Ni afterwards is an algorithmic way of dealing with the fact that unknown elements represent unknown freedom in the subspace Si. To understand this, consider an example Si—two columns taken from a rank 2 matrix (with asterisks representing unknown entries) - and the formulation of its orthogonal complement:

\[
\begin{pmatrix}
* & 3 & 2 & 7 & 5 \\
2 & 4 & * & 1 & 3
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c \\
d \\
e
\end{pmatrix}
= 0
\]  

(2.7)

Because there could be any value in the positions taken by the asterisks, a and c must be set to zero, leaving b, d and e to be calculated from the second, fourth and fifth columns of S⊤i. The vector (0, b, 0, d, e)⊤ is still perpendicular to Si and so is a member of the subspace defined by Ni = null(S⊤i).

When processing a noisy measurement matrix, the conditions for stopping are less well
Algorithm 2.3 Jacobs’ matrix factorization method.

inputs $M \in \mathbb{R}^{m \times n}$, $W \in \{0, 1\}^{m \times n}$, $(m, n, r) \in \mathbb{Z}$

1: declare $S \in \mathbb{R}^{m \times r}$, $S_i \in \mathbb{R}^{m \times r}$, $N \in \mathbb{R}^{m \times 0}$

2: for $i$ all $n$-choose-$r$ column selections do

3: $S_i \leftarrow i$th $r$-tuple of columns of $M$

4: $\{i'\} \leftarrow$ indices of complete rows in $S_i$

5: $k \leftarrow$ size $\{i'\}$

6: if $k > r$ then

7: declare $\hat{S}_i \in \mathbb{R}^{k \times r}$

8: $\hat{S}_i \leftarrow$ the $\{i'\}$ rows of $S_i$

9: $l \leftarrow k - \text{rank}(\hat{S}_i)$

10: declare $N_i \in \mathbb{R}^{m \times l}$, $\hat{N}_i \in \mathbb{R}^{k \times l}$

11: $\hat{N}_i \leftarrow \text{null}(\hat{S}_i^T)$

12: $N_i \leftarrow$ rows $\{i'\}$ from $\hat{N}_i$, $\mathbf{0}^T$ otherwise

13: $N \leftarrow [N \ N_i]$ (the nullity of $N$ is at least $l - k$)

14: end if

15: end for

outputs $S = \text{null}(N^T)$

defined. In Algorithm 2.3, it is suggested that all column selections should be tested, but $n$-choose-$r$ can easily be very large and so alternatives must be employed in practice. For example, simply randomize the selection of $r$-tuples of columns and set a hard limit for the number of iterations (Jacobs’ own implementation uses this method). Alternatively, the singular values of $N$ could be monitored as it grows beyond the minimum $(m - r)$ columns and termination could occur when they stop changing (very costly). Taking more than $r$ columns to form each $S_i$ is another way to help overcome the problem of noise. Unfortunately, the more columns that are chosen, the smaller the number of complete rows. The minimum number of complete rows in $S_i$ that is needed is $r + 1$ because an $S_i$ from a noisy measurement matrix will invariably have full rank and it must have a nullity of at least one for $N_i$ to exist. An subtle alternative to Algorithm 2.3 takes selections of $r + 1$ rows and collects all the columns with known elements on those rows. The full version would process $m$-choose-$(r+1)$ row combinations, but is otherwise the same. To minimize execution time, the transpose of a measurement matrix should always be considered as the input, though the correct orientation to use depends on the implementation. The time complexities of the two versions when all row or column combinations are worked through is dominated by the number of possibilities. A time complexity of $O(k$-choose-$l$) can be as bad as $O(2^k)$, but here (with $k \ll l$) it is approximately $O(k^l)$. So, for the column-wise approach, an input matrix with fewer columns than rows is better, whereas for the row-wise approach the converse is preferable. If the number of iterations is fixed in some way, then the dominant operation within the loop is the important factor. In both versions, it is the nullspace calculation that will influence the time complexity. The SVD can be used for that operation and so the time complexities become $O(mr^2)$ for column-wise and $O(n(r+1)^2)$ for row-wise, meaning that portrait matrices should now be given to the row-wise algorithm.
and vice-versa.

As Jacobs points out in his paper, this algorithm is not optimal, but he suggests that it is a good initialization technique for iterative schemes. The most notable development is that of Martinec and Padlja [24, 25] who extend this approach with the unknown scale factors in the projective SFM solution due to Triggs [38].

### 2.2 Alternation Approaches

Firstly, consider the error function of simple factorization:

\[
F(A, B) = \| \mathbf{M} - AB^\top \|^2_F
\]  

(2.8)

From a given point on the error surface, closed form solutions for new arguments that minimize the function in each variable can be derived in various ways (see Appendix A). Alternation is the strategy of solving for one and then the other variable, iteratively in turn. Algorithm 2.4 demonstrates the simplicity of the alternation scheme. In most cases it will converge on the answer quickly, although it is prone to crawling along shallow valleys when they are encountered.

**Algorithm 2.4** Alternation for factorization: \( \arg\min \{ F(A, B) = \| \mathbf{M} - AB^\top \|^2_F \} \)

**Inputs** \( \mathbf{M} \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, (m, n, r) \in \mathbb{Z} \)

**1:** repeat

**2:** \( A^\top \leftarrow (B^\top B)^{-1} B^\top \mathbf{M} \)

**3:** \( B^\top \leftarrow (A^\top A)^{-1} A^\top \mathbf{M} \)

**4:** until no change

**Outputs** \( A, B \)

This algorithm can be extended slightly by allowing the rows and columns of \( A \) and \( B \) respectively, to be weighted (for example, in the context of SFM, if there is more confidence in the tracking of certain points, or in the detection of points in certain views). \( \mathbf{W}_{\text{rows}} \) and \( \mathbf{W}_{\text{cols}} \) are diagonal matrices to allow

\[
F(A, B) = \| \mathbf{W}_{\text{rows}} (\mathbf{M} - AB^\top) \mathbf{W}_{\text{cols}} \|^2_F
\]  

(2.9)

Again, a simple alternation strategy can be derived, as shown in Algorithm 2.5.

Weighting on rows and/or columns only is a bit too restrictive and, indeed, unhelpful when it comes to missing data problems. Fortunately, it is easy to extend alternation further, to include element-wise weighting. We now go to the error function

\[
F(A, B) = \| \mathbf{W} \odot (\mathbf{M} - AB^\top) \|^2_F
\]  

(2.10)

and proceed as above. The difference, having introduced the Hadamard product, is that the equations may only be collected to the vector level and not to the matrix level as with the above two formulations. This gives Algorithm 2.6.
Algorithm 2.5 Row and column weighted alternation:

$$\text{argmin}\left\{ F(A, B) = \| W_{\text{rows}} (\mathbf{M} - AB^\top) W_{\text{cols}} \|^2_F \right\}$$

**Inputs** \( \mathbf{M} \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W_{\text{rows}} \in \text{diag} \mathbb{R}^m, W_{\text{cols}} \in \text{diag} \mathbb{R}^n, (m, n, r) \in \mathbb{Z} \)

1: repeat
2: \( A^\top \leftarrow (B^\top W_{\text{cols}}^2 B)^{-1} B^\top W_{\text{cols}}^2 \mathbf{M}^\top \)
3: \( B^\top \leftarrow (A^\top W_{\text{rows}}^2 A)^{-1} A^\top W_{\text{rows}}^2 \mathbf{M} \)
4: until no change

**Outputs** \( A, B \)

Algorithm 2.6 Weighted alternation: \(\text{argmin}\left\{ F(A, B) = \| \mathbf{W} \circ (\mathbf{M} - AB^\top) \|^2_F \right\}\)

**Inputs** \( \mathbf{M} \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: repeat
2: \( a_i \leftarrow (B^\top \text{diag}(w_i)^2 B)^{-1} B^\top \text{diag}(w_i)^2 m_i \quad \forall i \)
3: \( b_j \leftarrow (A^\top \text{diag}(w_j)^2 A)^{-1} A^\top \text{diag}(w_j)^2 m_j \quad \forall j \)
4: until no change

**Outputs** \( A, B \)

In the specific application of affine SFM an addition constraint arises: the translation of coordinate spaces. As seen in Equation 1.12 above, this adds an extra condition:

\[
\mathbf{M} = \mathbf{PX} + \mathbf{t} 1^\top \\
= \begin{bmatrix} \mathbf{P} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1^\top \end{bmatrix}
\]

namely, that the last column of \( B \) should be made up of ones (i.e. the three-dimensional scene points lie on an offset hyperplane in projective space). The alternation algorithm can be modified to incorporate this, giving Algorithm 2.7, overcoming the problem of mean-centring the measurement matrix when elements are unknown: the iteration scheme in Algorithm 2.7 estimates both the offset 3D hyperplane and the centroid of the data (as described in Chapter 1).

After each iteration, all these algorithms reduce the error. It is therefore guaranteed that they all reach a local minimum.

In 1976, Wiberg [41] introduced Algorithm 2.7. Shum, Ikeuchi and Reddy [30] came back to it in 1995 and, with minor algorithmic modifications, applied it to the measurement matrix of Tomasi and Kanade’s factorization approach. More recently, Guerreiro and Aguiar [16, 15, 14] have presented this algorithm again. Aguiar and Moura [3, 4] have recently suggested using the slightly less useful Algorithm 2.5. It was shown by Roweis [28] that alternation can, in fact, be derived within an EM framework.

There have also been several propositions that are variations on Wiberg’s approach. Vidal and Hartley [40] suggest adding a normalization step (Algorithm 2.8). Huynh, Hartley and Heyden [18] proposed performing alternation on a continually updated version of \( M \). Algorithm 2.9 outlines their algorithm.
Algorithm 2.7 Alternation for SFM: argmin \{ F(A, B) = \| W \odot (M - AB^T) \|_F^2 \} with \( b_r = 1 \)

**Inputs** \( M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: declare \( X \in \mathbb{R}^{n \times r-1}, P \in \mathbb{R}^{m \times r-1}, t \in \mathbb{R}^n 

2: repeat
3: \( a_i \leftarrow (B^T \text{diag}(w_i)^2 B)^{-1} B^T \text{diag}(w_i)^2 m_j \quad \forall i \)
4: \[
\begin{bmatrix}
P & t
\end{bmatrix} \leftarrow A
\]
5: \( x_j \leftarrow (P^T \text{diag}(w_j)^2 P)^{-1} P^T \text{diag}(w_j)^2 (m_j - t) \quad \forall j \)
6: \( B \leftarrow \begin{bmatrix} X^T & 1 \end{bmatrix} \)
7: until no change

**Outputs** \( A, B \)

Algorithm 2.8 Vidal and Hartley’s “PowerFactorization”: argmin \{ F(A, B) = \| W \odot (M - AB^T) \|_F^2 \}

**Inputs** \( M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: repeat
2: \( b_j \leftarrow (A^T \text{diag}(w_j)^2 A)^{-1} A^T \text{diag}(w_j)^2 m_j \quad \forall j \)
3: \( B \leftarrow \text{column-wise orthonormalization of } B \)
4: \( a_i \leftarrow (B^T \text{diag}(w_i)^2 B)^{-1} B^T \text{diag}(w_i)^2 m_j \quad \forall i \)
5: until no change

**Outputs** \( A, B \)

Algorithm 2.9 Huynh, Hartley and Heyden: argmin \{ F(A, B) = \| W \odot (M - AB^T) \|_F^2 \}

**Inputs** \( M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: declare \( X \in \mathbb{R}^{n \times r-1}, P \in \mathbb{R}^{m \times r-1}, t \in \mathbb{R}^n 

2: repeat
3: \( a_i \leftarrow (B^T \text{diag}(w_i)^2 B)^{-1} B^T \text{diag}(w_i)^2 m_j \quad \forall i \)
4: \( M \leftarrow \text{update elements of } M \text{ with large residuals} \)
5: \[
\begin{bmatrix}
P & t
\end{bmatrix} \leftarrow A
\]
6: \( x_j \leftarrow (P^T \text{diag}(w_j)^2 P)^{-1} P^T \text{diag}(w_j)^2 (m_j - t) \quad \forall j \)
7: \( B \leftarrow \begin{bmatrix} X^T & 1 \end{bmatrix} \)
8: until no change

**Outputs** \( A, B \)
Aanaes and Fisker, together with Åström and Carstensen [1], have put forward another method that works on an updated version of the measurement matrix. They use one alternation step after a subspace projection to give Algorithm 2.10. Although not actually related, Guerreiro and Aguiar [16, 15, 14], along with basic alternation, also present a similar project and merge iteration scheme. See Algorithm 2.11.

Algorithm 2.10 Aanaes, Fisker, Åström and Carstensen: \( \text{argmin}\{F(A, B) = \| W \odot (M - AB^\top) \|_F^2 \} \)

**inputs** \( M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: declare \( \hat{M} \in \mathbb{R}^{m \times n} \)
2: \( \hat{M} = M \)
3: repeat
4: \( A \leftarrow \hat{M}_{\downarrow 3} \)
5: \( b_j \leftarrow (A^\top \text{diag}(w_j)^2 A)^{-1} A^\top \text{diag}(w_j)^2 m_j \quad \forall j \)
6: \( \hat{M} \leftarrow W \odot M + (1 - W) \odot (AB^\top) \)
7: until no change

**outputs** \( A, B \)

Algorithm 2.11 Guerreiro and Aguiar’s two-step algorithm: \( \text{argmin}\{F(A, B) = \| W \odot (M - AB^\top) \|_F^2 \} \)

**inputs** \( M \in \mathbb{R}^{m \times n}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z} \)

1: declare \( \tilde{M} \in \mathbb{R}^{m \times n}, \hat{M} \in \mathbb{R}^{m \times n} \)
2: \( \tilde{M} \leftarrow M_{\downarrow r} \)
3: repeat
4: \( \hat{M} \leftarrow W \odot \tilde{M} + (1 - W) \odot \hat{M} \)
5: \( \tilde{M} \leftarrow \hat{M}_{\downarrow r} \)
6: until no change

**outputs** \( \tilde{M} \)

Many of the algorithms here do not explicitly deal with the fact that a measurement matrix with missing entries cannot be mean-centred. Sami Brandt [7] has formulated a closed form solution for \( B \) directly addressing this point. He has built an algorithm very similar to Algorithm 2.10, but using his equations to update \( B \) (which are too involved to present here). A summary of how all the alternation algorithms presented above are related is given in Figure 2.5.

Finally, a point that must be raised is the initialization of these algorithms. All require an initial estimate of \( A \) and \( B \). Although alternation is guaranteed to reach a minimum, it will only be a local minimum and so the starting guess is important. The answer returned by alternation schemes is very sensitive to initialization (see Figure 4.6). The algorithms of Section 2.1 can be used to fill the measurement matrix to obtain a place from which to start. The only other alternative is to set up \( A \) and \( B \) with random values and see where it goes. In Chapter 4 it is shown how effective these starting strategies are.
\begin{align*}
\text{inputs: } & A, B, M, \tilde{W}, r \\
1: & \hat{M} \leftarrow M \\
2: & \text{repeat} \\
3: & \tilde{M} \leftarrow \text{truncate } \hat{M} \text{ to rank } r \text{ (via SVD)} \\
4: & a' \leftarrow (B^{\top} \text{diag}(w_i^2)B + \lambda_1 I)^{-1}B^{\top} \text{diag}(w_i^2) \hat{m}_i \quad \forall i \\
5: & \hat{M} \leftarrow \text{update elements of } \tilde{M} \text{ with large residuals} \\
6: & A \leftarrow \text{first left singular vectors of } \hat{M} \\
7: & [A \ t] \leftarrow A; \hat{M} \leftarrow \hat{M} - tA^2 \\
8: & b_j' \leftarrow (A^2 \text{diag}(w_j^2)A + \lambda_2 I)^{-1}A^2 \text{diag}(w_j^2) \hat{m}_j \quad \forall j \\
9: & B \leftarrow \text{column-wise orthonormalization of } B \\
10: & B \leftarrow \text{Brandt closed-form update of } B \\
11: & B \leftarrow [B \ 1]; A \leftarrow [A \ t] \\
12: & \hat{M} \leftarrow \hat{M} + tA^2 \\
13: & \hat{M} \leftarrow \langle AB^2 \rangle \\
14: & \tilde{M} \leftarrow \tilde{M} + (1 - \tilde{W}) \odot \hat{M} \\
15: & \text{until convergence} \\
\end{align*}

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
ALT & PF & SIR & HHH & BDT & AFAC & GA \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 \\
4 & 4 & 4 & 4 & 5 & 6 & 6 \\
7 & 7 & 7 & 8 & 8 & 9 & 10 \\
11 & 11 & 11 & 12 & 12 & 13 & 13 \\
14 & 14 & 14 & 14 & 15 & 15 & 15 \\
15 & 15 & 15 & 15 & 15 & 15 & 15 \\
\hline
\end{tabular}
\end{center}

Figure 2.5: A table to compare basic alternation with the variants for the minimization of equation 1.5. Columns denote which lines are used by which algorithms. ALT = alternation; PF = PowerFactorization; SIR = Shum, Ikeuchi, and Reddy; HHH = Huynh, Hartley, and Heyden; AFAC = Aanaes, Fisker, Åström, and Carstensen; GA = Guerreiro and Aguiar; BDT = Brandt.

\section*{2.3 Other Literature}

Here is a very brief review of other literature that has not been included above.

Batch methods, such as those employed by Fitzgibbon and Zisserman [11] and Guilbert and Bartolli [17], have not been tested for comparison. These necessarily work on subsections of the measurement matrix and although it is desirable to use as much of the data in one go in order to see past the noise and avoid error magnification (as described in Section 2.1.3), the size of the problems can make an immediate solution intractable. Working on smaller sub-problems and combining the results is a way to cope with large problems. Batch methods will be included in future work.

Also, yet to be investigated are extensions to principal component analysis (PCA). A few authors [31, 6] have looked at incremental PCA, which in itself is not directly helpful to solving the missing data problem. However, the schemes naturally include the ability to weight data as they are incorporated into the analysis.

Robustness is an attribute omitted by the algorithms in this report, despite its importance. On this topic, the work by Torre and Black, who include an outlier term in the error function for their robust PCA [34] and so outlier rejection becomes part of the minimization, is very interesting. This will also be covered by future work.

Another approach that has a high potential is expectation-maximization (EM). It has been employed by a couple of authors [13, 2], but most notably by Torresani et al. [35, 36, 37, 8]. There is also work outside the computer vision field on accelerating and adjusting EM algorithms [21], as well as on the equivalence between EM and gradient methods [22].
Interestingly, robust PCA and EM algorithms can be described as alternation schemes in that, for each iteration, they consider subsets of parameters in turn while assuming the others are fixed. Torre and Black’s robust PCA does not use closed form solutions, but EM algorithms do, hence a strong connection. Yuille et al. [42] also propose an algorithm that falls into this category. They tackle the problem of illumination based reconstruction with a method that incorporates shadow detection (i.e. partial outlier rejection—specularities were not included). Their scheme updates one set of parameters using a closed form solution and another two sets of parameters using gradient descent within each iteration. At the end each iteration, they update their weight matrix based on the updated parameter values.

Irani’s work [19] on factorization with uncertainty is also interesting, although it does not address the missing data problem itself. She rearranges the measurement matrix from a $2F \times P$ matrix into a $F \times 2P$ form with $x$ and $y$ image coordinates in separate columns. The alternative matrix layout allows the Mahalanobis distance (between factorization and measurement) to be minimized directly.

Finally, it is worth mentioning that this report is similar to Triggs et al.’s bundle adjustment review [39], but here the general problem of factorization when data are missing is investigated, rather than looking generally at the specific problem of SFM.
Chapter 3

Error Surface Exploration

Let us continue looking at the missing-data error function

\[ F(A, B) = \left\| W \odot \left( \hat{M} - A B^\top \right) \right\|^2_F \] (3.1)

Section 2.2 dealt with alternation, which can be viewed as a coordinate-descent scheme, where the error surface is descended in each dimension of the parameter space in turn until there is nowhere else to go. Of course, there are several more ways to skate down the error surface. Here, gradient descent, Newton and the damped-Newton method are considered. These have not been investigated as methods to minimize Equation 3.1 in the computer vision literature. A hybrid method is also proposed.

3.1 Gradient Descent

As in Section 1.5, let us vectorize the arguments to the error function, A and B, as \( x \), for example,

\[ x = \left( a_{11} \ a_{12} \ \cdots \ a_{1r} \ \cdots \ a_{mr} \ b_{11} \ b_{12} \ \cdots \ b_{nr} \right)^\top \] (3.2)

Further, let us consider the error function in terms of individual matrix elements,

\[ F(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{i j} \left( m_{i j} - \sum_{k=1}^{r} a_{ik} b_{jk} \right)^2 \] (3.3)

\[ = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{i j}^2 \left( m_{i j}^2 - 2m_{i j} \sum_{k=1}^{r} a_{ik} b_{jk} + \left[ \sum_{k=1}^{r} a_{ik} b_{jk} \right]^2 \right) \] (3.4)
Algorithm 3.1 Gradient descent: argmin\(F(A, B) = \|W \odot (M - AB^\top)\|^2_F\)

**inputs** \(M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z}\)

1: declare function \(F = \|W \odot (M - AB^\top)\|^2_F\)
2: declare \(x \in \mathbb{R}^{r(m+n)}, d \in \mathbb{R}^{r(m+n)}\)
3: \(x \leftarrow \text{vectorize}(A, B)\)
4: repeat
5: \(d \leftarrow \frac{\partial F}{\partial x}\)
6: \(\lambda \leftarrow \text{minimize } F(x - \lambda d)\)
7: \(x \leftarrow x - \lambda d\)
8: until no change

**outputs** \(A, B \leftarrow \text{unvectorize}(x)\)

from which the partial derivatives of \(F\) with respect to all the elements of \(A\) and \(B\) can be more easily seen (see Appendix A).

\[
\frac{\partial F}{\partial a_{ab}} = -2 \sum_{j=1}^{n} w_{aj} b_{jb} \left( m_{aj} - \sum_{k=1}^{r} a_{ak} b_{jk} \right) \tag{3.5}
\]

\[
\frac{\partial F}{\partial b_{cd}} = -2 \sum_{j=1}^{n} w_{ic} a_{id} \left( m_{ic} - \sum_{k=1}^{r} a_{ik} b_{ck} \right) \tag{3.6}
\]

With these partial derivatives, the complete vector derivative of \(F\) can be calculated.

The gradient descent algorithm performs a line search in the direction of the gradient at the current point, and moves to a point on the line that has lower error. Logarithmic or bisection-style searches can be employed, though once a few points on the surface are known then quadratic or cubic fits might provide a faster descent for problems in a small number of dimensions. See Algorithm 3.1.

### 3.2 Newton

Fortunately, the second derivatives of the error function are readily available and so Newton’s method can be employed. The function is approximated as being quadratic, for which there is a closed form solution:

\[
\arg\min_x \left( a + b^\top x + \frac{1}{2} x^\top C x \right) = -C^{-1}b \tag{3.7}
\]

From the Taylor series expansion, the local quadratic approximation of the function \(F\) about \(x\) gives

\[
F(x + \delta) \approx F(x) + \nabla^\top F(x) \delta + \frac{1}{2} \delta^\top H(x) \delta \tag{3.8}
\]
Algorithm 3.2 Newton’s method: \( \arg \min F(A, B) = \| W \odot (M - AB^T) \|_F^2 \).

**inputs** \( M \in \mathbb{R}^{m \times n} \), \( A \in \mathbb{R}^{m \times r} \), \( B \in \mathbb{R}^{n \times r} \), \( W \in \mathbb{R}^{m \times n} \), \((m, n, r) \in \mathbb{Z}^3\)

1: declare function \( F = \| W \odot (M - AB^T) \|_F^2 \)

2: declare \( x \in \mathbb{R}^{r(m+n)} \), \( d \in \mathbb{R}^{r(m+n)} \), \( H \in \mathbb{R}^{r(m+n) \times r(m+n)} \)

3: \( x \leftarrow \text{vectorize}(A, B) \)

4: repeat

5: \( d \leftarrow \frac{\partial F}{\partial x} \)

6: \( H \leftarrow \frac{\partial^2 F}{\partial x^2} \)

7: \( \lambda \leftarrow \text{minimize} F(x - \lambda H^{-1}d) \)

8: \( x \leftarrow x - \lambda H^{-1}d \)

9: until no change

**outputs** \( A, B \leftarrow \text{unvectorize}(x) \)

with the entries of the hessian matrix, \( H \), for a point \( x \), provided by the second derivatives:

\[
\frac{\partial^2 F}{\partial a_{ij} \partial a_{ef}} = 2\delta_{ae} \sum_{j=1}^n w_{aj}^2 b_{jf} b_{jf} \quad (3.9)
\]

\[
\frac{\partial^2 F}{\partial b_{cd} \partial a_{ef}} = 2w_{ec}^2 \left( a_{cd} b_{ef} + \delta_{df} \left( -m_{ec} + \sum_{k=1}^r a_{ek} b_{ck} \right) \right) \quad (3.10)
\]

\[
\frac{\partial^2 F}{\partial a_{ab} \partial b_{gh}} = 2w_{ag}^2 \left( a_{ah} b_{gh} + \delta_{bh} \left( -m_{ag} + \sum_{k=1}^r a_{ak} b_{gk} \right) \right) \quad (3.11)
\]

\[
\frac{\partial^2 F}{\partial b_{cd} \partial b_{gh}} = 2\delta_{cg} \sum_{i=1}^m w_{ic}^2 a_{id} a_{ih} \quad (3.12)
\]

However, rather than jumping to the minimum of this approximation to the surface at \( x \), the gradient direction should be used to guide a line search. Newton’s method is given in Algorithm 3.2.

### 3.3 Damped Newton

Newton’s method assumes that a quadratic surface is always a helpful approximation to the actual error surface. Sometimes it may be better to simply perform a gradient descent step, for example when \( H \) is not positive-definite. Newton’s method can be easily modified to act in the fashion of the Levenberg Marquardt algorithm. Also known as trust region minimization, the procedure is presented in Algorithm 3.3.

### 3.4 Damped Newton with Line Search

Matrix inversion is computationally expensive. Every run through the inner loop of the damped Newton algorithm includes a potentially large matrix inversion. An extension that
Algorithm 3.3 Damped Newton: argmin$F(A, B) = \| W \odot (M - AB^\top) \|_F^2$

inputs $M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z}$
1: declare function $F = \| W \odot (M - AB^\top) \|_F^2$
2: declare $x \in \mathbb{R}^{r(m+n)}, y \in \mathbb{R}^{r(m+n)}, d \in \mathbb{R}^{r(m+n)}, H \in \mathbb{R}^{r(m+n) \times r(m+n)}$
3: $x \leftarrow$ vectorize$(A, B)$
4: $\lambda \leftarrow 0.01$
5: repeat
6: $d = \frac{\partial F}{\partial x}$
7: $H = \frac{\partial^2 F}{\partial x^2}$
8: repeat
9: $\lambda \leftarrow \lambda \times 10$
10: $y = x - (H + \lambda I)^{-1}d$
11: until $F(y) < F(x)$
12: $x \leftarrow y$
13: $\lambda \leftarrow \lambda \div 10$
14: until no change
outputs $A, B \leftarrow$ unvectorize$(x)$

Attempts to reduce the number of inversions combines the damped Newton iteration with the standard line search (Algorithm 3.4). Having calculated the descent vector as $(H + \lambda I)^{-1}d$ the next step is to perform a line search in this direction rather than a rejection-update inner loop. The parameter $\lambda$ is then updated based on where on the line a minimum was found. If the minimum was very close to the start point for that iteration, i.e. a small step, $\lambda$ is increased proportionally to emulate that small step for the next iteration. Conversely, if a large step was performed, $\lambda$ is reduced.

3.5 Hybrid System

Alternation is very fast initially, but often gets stuck in the many shallow valleys that appear when the number of unknown elements increases. Damped Newton is fast in valleys, but fairly ineffectual when far from minima where alternation is better. A graphical depiction of this is given in the results section (Figure 4.7). An interesting proposal, therefore, is to combine the two. The combination is suggested in passing by Little and Rubin [23], but no references about its performance have been found, especially in the computer vision community.

The main decision regarding the implementation of the hybrid scheme is how to determine when to switch from one method to the other. The $\lambda$ parameter of damped Newton is useful in this respect. When the damped Newton method starts to resort to the gradient descent strategy, i.e. when the $\lambda$ parameter is large, the step sizes are getting very small and progress will be very slow. Another interpretation of $\lambda$ is as a measure of the fit of the quadratic approximation. The larger the value of $\lambda$ the worse the fit. Therefore, switching to alternation when $\lambda$ is large could be advantageous.
Algorithm 3.4 Damped Newton with line search: $\text{argmin} F(A, B) = \| W \odot (M - AB^\top) \|_F^2$.

inputs $M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z}$

1: declare function $F = \| W \odot (M - AB^\top) \|_F^2$
2: declare $x \in \mathbb{R}^{r(m+n)}, d \in \mathbb{R}^{r(m+n)}, H \in \mathbb{R}^{r(m+n) \times r(m+n)}$
3: $x \leftarrow \text{vectorize}(A, B)$
4: $\lambda \leftarrow 0.1$
5: repeat
6: $d = \frac{\partial F}{\partial x}$
7: $H = \frac{\partial^2 F}{\partial x^2}$
8: $\alpha \leftarrow \text{minimize } F(x - \alpha(H + \lambda I)^{-1}d)$
9: $x \leftarrow x - \alpha(H + \lambda I)^{-1}d$
10: $\lambda \leftarrow \lambda \div \alpha$
11: until no change

outputs $A, B \leftarrow \text{unvectorize}(x)$

There are many ways in which switching between alternation and damped Newton based on the value of $\lambda$ can be implemented. Algorithm 3.5 gives one example: $\lambda$ reaching a threshold value is used to initiate the switch from damped Newton to alternation. The switch back is made after a set number of alternation steps, with the hope that a place more suited to the Newton method has been found.

Two other strategies were implemented: a very basic version which calculated the new state using both alternation and damped Newton and chose between them using the new error for each (the line search method was used instead of the full damped Newton scheme for speed); and a more complicated version that monitored the value of $\lambda$ within the damped Newton optimization loop (the innermost loop) and bailed out when $\lambda$ became too large. A comparison of these implementations is given in the results.
Algorithm 3.5 The alt./damped Newton hybrid: argmin\(F(A, B) = \|W \odot (M - AB^\top)\|^2_F\)

**inputs** \(M \in \mathbb{R}^{m \times n}, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r}, W \in \mathbb{R}^{m \times n}, (m, n, r) \in \mathbb{Z}\)

**declare** function \(F = \|W \odot (M - AB^\top)\|^2_F\)

**declare** \(x \in \mathbb{R}^{r(m+n)}, y \in \mathbb{R}^{r(m+n)}, d \in \mathbb{R}^{r(m+n)}, H \in \mathbb{R}^{r(m+n) \times r(m+n)}\)

\(\lambda \leftarrow 0.01\)

SWITCH \(\leftarrow 100\) // the value of \(\lambda\) for a change to alternation

COUNT \(\leftarrow 4\) // the number of alternation steps before returning to Newton

\(c \leftarrow\) COUNT

**repeat**

**if** \(c > 0\) **then**

\(a^i \leftarrow (B^\top \text{diag}(w^i)^2 B)^{-1} B^\top \text{diag}(w^i)^2 m_j \forall i\)

\(b^j \leftarrow (A^\top \text{diag}(w_j)^2 A)^{-1} A^\top \text{diag}(w_j)^2 m_j \forall j\)

**else**

\(x \leftarrow \text{vectorize}(A, B)\)

\(d = \frac{\partial F}{\partial x}\)

\(H = \frac{\partial^2 F}{\partial x^2}\)

**repeat**

\(\lambda \leftarrow \lambda \times 10\)

\(y = x - (H + \lambda I)^{-1} d\)

**until** \(F(y) < F(x)\)

\(x \leftarrow y\)

\(\lambda \leftarrow \lambda \div 10\)

\(A, B \leftarrow \text{unvectorize}(x)\)

**end if**

**if** \(\lambda \geq \text{SWITCH}\) **then**

\(\lambda \leftarrow \lambda \div 10\) // start safely on return to Newton

\(c \leftarrow\) COUNT

**end if**

**until** no change

**outputs** \(A, B\)
Chapter 4

Results

Experiments were carried out to discover the relative performance of the algorithms presented in Chapters 2 and 3. Several separate tests explore different aspects of the problem. Firstly, the results from synthetic tests are presented to give an idea of the general relative performance of the algorithms. Not all algorithms were implemented at the time of these tests, but the comparisons that are drawn reflect well the overall picture. Then we return to the example problems introduced in Section 1.1. All the algorithms were run on these examples and the results are informative. Finally in this chapter, the dinosaur sequence is used to quickly demonstrate the suitability of the error function that all the algorithms described above have been designed to minimize.

4.1 Synthetic Performance Comparison

Synthetic tests were run to compare the performance of six algorithms: alternation, PowerFactorization, Aanaes et al.’s variation, damped Newton, damped Newton plus line search and the hybrid method. All were implemented to be as efficient as possible. Each test was simply a general factorization problem of a noisy, incomplete, rank deficient matrix. A summary of the results of 10000 random synthetic tests is shown in Figures 4.1 and 4.2. The graphs show that the relative success of the algorithms is varied. It is the three Newton-based algorithms that perform most successfully. Alternation is less good at minimizing the error function. Vidal and Hartley’s and Aanaes et al.’s algorithms display the worst performance in these tests. Comparing their relative speeds also shows a range of attainments. Damped Newton and damped Newton with line search are the slowest algorithms (with the line search version being slightly quicker as expected). The Aanaes et al. algorithm is also slow, which can be attributed to its use of the SVD in each iteration. Alternation and PowerFactorization are the quickest, with the hybrid coming in about twice as slow. The number of iterations until convergence is also interesting because it is less implementation sensitive. The comparison here shows the hybrid method to be the best and the Aanaes et al. method the worst.
Figure 4.1: Results from 10000 synthetic random general tests. The bar chart displays the number of tests in which each of the six algorithms attained the lowest error of all the methods for individual tests. The same initial guess was used by all algorithms in each test. Alt = alternation; Aan = Aanaes et al.; ADN = hybrid method; Pow = PowerFactorization; DN = damped Newton; DNl = damped Newton plus line search.

Figure 4.2: Results from the same tests used for Figure 4.1. The left chart shows the average relative number of iterations and the right the average time for convergence for the six algorithms. As the tests took different parameters, the number of iterations and times taken are not comparable across all tests. For these graphs, the statistics were normalized to the alternation algorithm for each trial before being averaged. Alt = alternation; Aan = Aanaes et al.; ADN = hybrid method; Pow = PowerFactorization; DN = damped Newton; DNl = damped Newton plus line search.
4.1.1 Coupling

The distribution of known elements in the measurement matrix has a large impact on the success of the factorization process. As discussed in Section 2.1, if the configuration of known elements is such that blocks of the matrix are effectively independent of each other (see Figure 2.2) then, in general, it is impossible to reconstruct the full matrix. Matrices with this structure are deemed to have poor coupling, the extent to which regions of the matrix are linked to each other.

Figure 4.3 shows the results of three experiments in a set of many tests covering a large range of weight matrix configurations. The examples in the figure are representative of the trends in the full results. The experiment at the top of the figure used a dense weight matrix and so had no coupling issues. The second used a weight matrix that is almost decoupled, having very few elements that overlap with both the diagonal quadrants. In the last experiment shown, the weight matrix is completely decoupled and the two diagonal quadrants are entirely independent of each other. The results displayed for each experiment comprise: the structure of the weight matrix on the left, with black representing ‘known’ data and white signifying ‘missing’ data; and a set of six residual matrices showing the success of the three algorithms used in the test. The two leftmost matrices are alternation, a hybrid method is in the centre and the residuals from the damped Newton algorithm are on the right. The results from the three hybrid methods that were implemented are so similar that the results from only one can be shown to convey the trend clearly (see Section 4.2.1). The top row of residuals for each experiment show are the differences between the measurement matrix passed to the algorithms (a noisy version of the actual measurement matrix with the noise added having a magnitude 5% of the true matrix element values) and the results they returned, filtered using the weight matrix (i.e. the errors summed in the calculation of the error function). The bottom row are the residuals of the solutions given by the algorithms compared to the actual (noise free) measurement matrix. The residual matrices are coloured such that ‘hotter’ colours denote larger differences between the appropriate measurement matrices and the reprojections. Only the weight matrix changed between the tests; all other matrices were kept constant, including the initial guess.

When the known data is dense, none of the algorithms have any trouble finding a solution. The pattern of the residuals is identical for all three solutions, suggesting that global minimum has been found.

As the weight matrix approaches the decoupled state, alternation and the hybrid methods start to become unable to reconstruct all of the missing elements of the measurement matrix. Alternations coordinate descent scheme, where A and B are optimized separately of each other, means that the few known elements that link the two large known blocks cannot be used in any significant way. Damped Newton, on the other hand, using a scheme that optimizes both A and B simultaneously, manages to use the few elements that are known in the off-diagonal quadrants to successfully recover those regions.

Once all elements in the off-diagonal quadrants are lost, there is nothing that can be done to reconstruct those portions of the matrix. Note that all three algorithms are told only about the weighted error (generated by the upper residual matrices shown for each experiment in the figure), which shows them all to be doing roughly equally well: the weight matrix is
Dense weight matrix.

Almost decoupled weight matrix.

Completely decoupled weight matrix.

Figure 4.3: A demonstration of the effect coupling has on the reprojection error (see Section 4.1). $\hat{w}$ on left, residuals on right. Error function residuals in top row, actual residuals on bottom row.
Figure 4.4: The overall results of 10000 tests with independently varied matrix size, noise level and visibility (fraction of missing data). All tests were the factorization of a rank 4 synthetic measurement matrix whose elements actually exist in an offset 3D hyperplane, simulating the situation in SFM. Three algorithms were run in each test: general alternation, SFM alternation (Wiberg) and “partially mean-centred” alternation. The left chart shows counts of each of the algorithms obtaining a solution with the lowest error (using the weighted error function) of the three methods for each test. The right chart is the count of the solution being the closest to the actual answer.

hiding the poor fit of the two unknown blocks.

4.1.2 SFM vs General Factorization

The structure from motion problem has the special constraint that the last column of B should be 1, i.e. the three-dimensional scene points lie in an offset hyperplane in a four-dimensional space. The general factorization algorithm does not account for this. Wiberg’s modification to the alternation algorithm is the template for how to perform the correct minimization in this specific situation. Chapter 1 discussed this in more detail.

Synthetic measurement matrices were generated by multiplying together random four-column $A$s and $B$s (with the last column of $B$ set to one). Noise was added and it was then passed, with a random weight matrix, to effectively three algorithms: (1) alternation looking for a general rank four solution, (2) Wiberg’s alternation and (3) alternation looking for a general rank 3 factorization to a “partially mean-centred” measurement matrix. The last set-up took the input matrix and then subtracted a weighted column sum using the weight matrix. This is the missing data equivalent to mean-centring the measurement matrix in an attempt to convert the problem to a rank three general factorization (see Section 1.4.1). 10000 tests were executed over a variety of problems. The size, noise and visibility of the input measurement matrix were all independently randomized for each test.

The results are shown in Figure 4.4. They show that the general factorization is better at minimizing the missing-data error function, but that Wiberg’s alternation for SFM gets answers that are closer to the actual solution. Of course, in actual problems, the real solution is not known and this comparison cannot be made. Attempting to mean-centre the measurement matrix to make it a rank 3 problem by finding the row-wise average of the
visible elements is not effective. Optimizing for the 3D hyperplane and the offset using Wiberg’s formulation is best option.

4.2 Real Problems

The example applications presented in Section 1.1 were used to compare all the algorithms described in this report. A summary of the problems with an overview of the results is given by Figure 4.5. For each problem, all the algorithms were run using two classes of initialization. The algorithms were firstly started from the answer given by Jacobs’ reconstruction algorithm and secondly from 500 or 1000 random starting points. The large number of random initializations was undertaken in an effort to gain a picture of the converge basins of the local minima. The parameter space in which the error function is being minimized is very large and an exhaustive search is intractable. Fortunately it turns out that the convergence basins of the minima in these problems are large enough for many minima to be isolated using only hundreds of initial estimates. In all three problems one minimum was significantly more prominent than all other final error values. This error has been taken as the global minimum of each problem. Although this cannot be said for sure, the evidence strongly suggests that this is the case. The last row of Figure 4.5 is a basic summary of the three base algorithms’ performance and reflects the size of the convergence basins for them. The full plots are given in the following subsections. In these cumulative frequency plots, the height of each step in the curve is representative of the size of the convergence basin for the minimum with that error. The first step corresponds to the global minimum and so the height of the first step can be taken as a measure of algorithm success.

Sensitivity to initialization is a large aspect of the practical situation. Figure 4.6 shows a contour plot of a slice through the error surface (seen in Figure 1.3) around three minima for the dinosaur problem. Superimposed onto the plot are three descent trajectories taken by the damped Newton method as it minimized the error function from three different starting points. Note that the relative spread of the minima in comparison to the spread of the initial states is large. Initial guesses must be close to the global minimum for it to be found. Direct methods (Section 2.1) are often suggested as ways of obtaining a good initial guess. This assertion was tested using Jacobs’ method. The reconstruction given by Jacobs’ algorithm and the results from all algorithms initialized using it did not give the lowest errors seen for any of the problems, i.e. they were not near the global minimum of the error function. For missing data problems it is clear that Jacobs’ method cannot be relied upon for a solution or as a good initialization for any of the iterative schemes that have been tested.

It is important to note that outliers were not included in any of the example problems. Although some algorithms (including the Newton based algorithms) can be extended to be robust to outlying data, none of those tested could cope with outliers. All three measurement matrices were ‘clean’ of erroneous data entries when passed to the algorithms being tested.
Figure 4.5: Summary of the results from real tests. All algorithms presented in the text were used to solve the factorization problem associated with each of the three examples introduced in Section 1.1. The three columns correspond to the three problems: the recovery of 1) the rigid turntable motion of a toy dinosaur; 2) the non-rigid occluded motion of a giraffe in the background; and 3) the light directions and surface normals of a static face with a moving light source. The first row shows a frame from the sequence. The second represents the sparsity of the measurement matrix. The third row is a detail of the accumulation histogram based on the final error and counted over all runs.
4.2.1 Hybrid Methods

Before looking at the results of all the algorithms’ performance on the real problems, here is a quick summary of the performance of the hybrid methods. Looking at a typical attempts to solve for the structure and motion of the rotating dinosaur sequence, it is clear that there is a big difference between damped Newton and alternation. Figure 4.7 shows the error being reduced by the two base algorithms and the three hybrid schemes described at the end of Chapter 3 for one such typical attempt. Across all attempts, examples can be found of each outperforming the other, but the run shown is the mode behaviour.

The key points in Figure 4.7 are that a) alternation converges much quicker than damped Newton initially and b) damped Newton eventually catches up and overtakes alternation, leaving alternation to converge very slowly. The aim of introducing the hybrid schemes is to capture the positive attributes of both algorithms to get a method that can converge quickly both initially and ultimately. Figure 4.7 also shows the descent for the three hybrid schemes. They are very similar and lie between alternation and damped Newton. In almost all tests, the hybrid schemes performed almost identically. For the rest of this chapter they will be all referred together and be represented in figures by just one line to increase clarity.
Figure 4.7: Typical error descent curves for the alternation, damped Newton and hybrid algorithms. All algorithms were started with the same random initial state vector. This is a minimization for the dinosaur sequence.

4.2.2 Rotating Dinosaur

Consider again the rotating dinosaur sequence. Automated tracking software generated feature tracks which were used to fill the measurement matrix. The tracks were then filtered for outliers using the knowledge that motion is rotational (i.e. elliptical tracks—see Section 4.3). The known values fill about 28% of the matrix. Because it is a video of rigid motion, theoretically only 5% of the matrix needs to be known. However, it is still a very sparse matrix.

Each algorithm was run 1000 times from random starting points. To test the algorithms further, they were all run on sub-blocks of the matrix as well. The sequence was split into two- and five-block subsequences, processed and then stitched back together to return to the full measurement matrix.

A representative reconstruction of the stitched solutions is shown in Figures 4.8 and 4.9. For this example, stitching together solutions to the smaller problems exaggerated the problems of error magnification (Section 2.1.3) leaving a very poor result. The runs on the measurement matrix taken as a whole proved to give better results. The left hand pane of Figure 4.14 shows the best reconstruction (at the assumed global minimum) of all the full factorization tests. Here we have a much more plausible answer, but still an unsatisfactory suggestion for turntable motion. This is discussed more at the end of this chapter.

The cumulative frequency plot in Figure 4.10 shows the relative successes of the algorithms in factorizing the full rotating dinosaur measurement matrix. As explained above, the higher the curve, the more successful the algorithm has been on this problem. The ‘project and merge’ algorithm, together with gradient descent, proved to be the worst schemes. All the alternation style methods performed with a similar level of success. Damped Newton has the best performance, with the hybrid methods appearing to be almost as good. None of the other algorithms, including the damped Newton with line search

46
Figure 4.8: On the left are the tracks of points detected in a video sequence of a rotating toy dinosaur. These tracks are enough to fill just over a quarter of the measurement matrix. Note the elliptical trajectories. On the right is a typical reconstruction from any one of the stitched results. The measurement matrix has been filled, but elliptical tracks have not been generated throughout the reconstruction. See Figure 4.9 for a more concise view of this measurement matrix.

Figure 4.9: Images of the measurement matrix from the rotating dinosaur sequence solution obtained by stitching (see Figure 4.8). The hotter the colour, the larger the value in the matrix. On the left is the reconstruction and on the right is the same matrix with the known elements darkened for comparison. Notice that, as the elements that have been filled by the process get further from the central diagonal band of known elements, the magnitude of the residual gets much larger. As all points should remain in view for the whole sequence this image is almost directly related to the error with respect to the unknown ground truth.
Figure 4.10: Cumulative frequency graphs for the results of 1000 randomly initialized runs on the dinosaur sequence. The larger the initial step, the better the algorithm can be considered to be. The left pane shows the full plot, the right shows a detail of the bottom left.

Figure 4.11: Cumulative frequency graph for the results of 500 randomly initialized runs on the giraffe sequence. The full plot in on the left, with a detail from the bottom left of the plot on the right.

algorithm, found the global minimum from any of the 1000 random starting points of these runs.

4.2.3 Occluded Giraffe

The giraffe sequence provides a non-rigid motion example. Features on a giraffe walking behind another giraffe were tracked by hand to give a measurement matrix free of outliers. For this problem (being larger than the dinosaur) each algorithm was run only 500 times. The damped Newton with line search algorithm was not run on the giraffe sequence and so can not be included in the comparison.

Refer to the cumulative frequency graph in Figure 4.11. Again, ‘project and merge’ and gradient descent proved to be the worst schemes. This time, two of the alternation type
Figure 4.12: Example results for the giraffe sequence. (a) Frame 120 showing the whole background giraffe. (b) Frame 48 with the background giraffe occluded by the foreground giraffe. (c) The tracked points in Frame 120 (known coordinates as entered into the measurement matrix). (d) The tracked points in Frame 48. (e) A typical reconstruction of Frame 48 from the output of the gradient descent algorithm. (f) A typical result (Frame 48 again) of those algorithms that actually found the global minimum (e.g. alternation and the Newton based methods).
methods found the global minimum: alternation and PowerFactorization. They both did better than damped Newton. However, the hybrid methods showed themselves as the best algorithms for this problem. None of the other algorithms found the global minimum.

Figure 4.12 demonstrates the variation in success between one of the best performers (alternation) and the worst (gradient descent). For the frame shown, alternation (and the other successful algorithms) managed to propose very plausible positions for the features that could not be seen in that image. Gradient descent found a minima where even the visible points reproject very poorly.

4.2.4 Illuminated Face

For a demonstration of illumination based reconstruction, an image sequence of a static face lit from twenty different directions has been used. As described in Chapter 1, the pixel intensity values are transferred directly into the measurement matrix. Again, outliers must not be included and so all pixels for which the intensity value is not within diffuse illumination limits (pixels representing shadows and specularities) must be omitted. The algorithms were run 1000 times with random starting points for A and B. The results from the ‘project and merge’ algorithm have not been included because no runs using that method ever converged at all.

Figure 4.13 displays the results of the runs on this problem. Here, we can see that all three Newton based methods (damped Newton, line search and hybrid) performed well. The line search method only just made it to the global minimum. Alternation was the only other notable algorithm, but it did not find the lowest minimum. All the other algorithms performed very badly.
Figure 4.14: Point tracks for the dinosaur turntable sequence. See Figure 4.8 for the input tracks. On the left are the best tracks obtained using the algorithms discussed. On the right are tracks obtained using simple priors (orthonormality of the camera matrices in the columnspace).

4.3 Priors

An important question is whether the error function that all the algorithms tested are minimizing is one that can provide satisfactory solutions. As Figure 4.14 shows, minimizing the error function of Equation 1.5 (pure reprojection error) does not give the most satisfactory results. In the same figure is a the global minimum of the same error function plus a regularizing term that penalized unorthonormal cameras:

\[
F(A, B) = \| W \odot (M - AB^T) \| + \sum_{f=1}^{m/2} \{ p_f^2 - 1 \}
\]

Here, \( p_i \) has been used to denote the first three entries of the \( i^{th} \) row of the matrix \( A \). Recall that every pair of rows (more specifically, every odd plus even row pair) of \( A \) has the interpretation \( [P_f \ t_f] \), hence the regularizer used. The improvement in the results through the use of a simple prior is pronounced. I think this clearly shows that the incorporation of prior knowledge of the problem into the minimization is very important. In one respect, an even simpler prior has already been introduced earlier in the report. The knowledge that for the SFM problem, the columns of the measurement matrix should lie on an offset hyperplane of dimension one less than the rank lead to the adaptation of the alternation algorithm used by Wiberg and Shum et al. In that case the inclusion of the prior allowed the retention of the closed form solutions required by alternation. However, in general, it is hard to develop priors that facilitate closed form solutions. The regularized error function above was minimized by damped Newton to give the result shown in Figure 4.14.
Chapter 5

Conclusions

It is clear that factorization is an important tool for computer vision. However, real applications must deal with missing data to be effective and with standard factorization algorithms being unable to deal with missing data, many iterative algorithms have been proposed in the computer vision literature to repair this deficiency. In addition to a review of these existing methods, this report presents several routines based on the Newton method. Such methods have not been presented before in the computer vision literature for the problem of factorization with missing data. All existing methods published on this problem are in some way based on alternation. Hybrid methods, which have also not been covered in the literature, combine these two approaches and have also been described. A comparison of the three classes of algorithm showed that alternation is not a viable choice, mainly due to the excessively slow convergence rates, even though such methods give good initial convergence. Hybrid methods might have their place as they performed consistently across the three real problems used for comparison. However, an important conclusion of the work regards the suitability of the basic error function. Without the incorporation of regularizers to encapsulate prior knowledge of a problem, overfitting invariably means that unsatisfactory results are obtained. Formulating priors such that alternation-style closed form solutions exist for the augmented error function is very difficult. On the other hand, adding priors that are twice differentiable is considerably easier and so Newton based methods provide a flexible framework for factorization and are my preferred strategy.
Appendix A

An Alternation Derivation

The error function, $F(A, B) = \|W \odot (M - AB^\top)\|_F^2$, can be minimized, with respect to either of the two parameters (given a particular value for the other), using a closed form solution. Here, the derivation via zero-crossings of the gradient function is presented. The solution for $A$, for a given $B$, is laid out. The solution for $B$ is very similar and so will not be expanded. Firstly, $F$ is rewritten in terms of the rows of $A$.

$$F(A, B) = \|W \odot (M - AB^\top)\|_F^2$$ (A.1)

$$= \sum_{i=1}^{m} \|w_i^\top \odot (m_i^\top - a_i^\top B)\|_F^2$$ (A.2)

$$= \sum_{i=1}^{m} \|\text{diag}(w_i) (m_i - Ba_i)\|_F^2$$ (A.3)

$$= \sum_{i=1}^{m} (m_i - Ba_i)^\top \text{diag}(w_i)^2 (m_i - Ba_i)$$ (A.4)

$$= \sum_{i=1}^{m} m_i^\top \text{diag}(w_i)^2 m_i - 2m_i^\top \text{diag}(w_i)^2 Ba_i + a_i^\top B^\top \text{diag}(w_i)^2 Ba_i$$ (A.5)

Taking the first and second derivatives of this sum with respect to $a_i$ gives

$$\frac{\partial F}{\partial a_i} = \left(-2m_i^\top \text{diag}(w_i)^2 B + 2a_i^\top B^\top \text{diag}(w_i)^2 B\right)^\top$$ (A.6)

$$= 2 \left(B^\top \text{diag}(w_i)^2 Ba_i - B^\top \text{diag}(w_i)^2 m_i\right)$$ (A.7)

$$\frac{\partial^2 F}{\partial a_i^2} = 2B^\top \text{diag}(w_i)^2 B$$ (A.8)

The second derivative is at least positive semi-definite and if there are more than $r$ non-zero entries along the diagonal of $w_i$ then it is positive-definite (remember that $B$ is defined as a rank $r$, $n \times r$ matrix). In the latter case, there is only one zero-crossing of the first derivative.
and that is a minimum. Setting the gradient to zero gives the solution:

$$\frac{\partial F}{\partial a_i} = 0$$  \hspace{1cm} (A.9)

$$\Rightarrow \quad B^\top \text{diag}(w^i)^2 B a_i = B^\top \text{diag}(w^i)^2 m^i$$  \hspace{1cm} (A.10)

$$\Rightarrow \quad a_i = \left( B^\top \text{diag}(w^i)^2 B \right)^{-1} B^\top \text{diag}(w^i)^2 m^i = (\text{diag}(w^i)B)^+ \text{diag}(w^i)m^i$$  \hspace{1cm} (A.11)

The solution for $B$, given a matrix $A$, is very similar. The result comes out to be

$$b_i = \left( A^\top \text{diag}(w^i)^2 A \right)^{-1} A^\top \text{diag}(w^i)^2 m_j = (\text{diag}(w_j)A)^+ \text{diag}(w_j)m_j$$  \hspace{1cm} (A.12)

Because both these sets of equations for the rows of each matrix are independent within themselves, they may actually be concatenated into two individual matrix equations:

$$A^\top = (\text{diag}(w^i)B)^+ \text{diag}(w^i)M^\top$$  \hspace{1cm} (A.13)

$$B^\top = (\text{diag}(w_j)A)^+ \text{diag}(w_j)M$$  \hspace{1cm} (A.14)

This should not be surprising. These solutions can be seen directly from the original equation, as the pseudo-inverse gives the least-squares solution to an overconstrained (inconsistent) system of equations.
Appendix B

Higher Order Terms

Differentiating the error function further reveals more of the quartic nature of the main error function. The following derivatives are made with respect to the individual elements of $A$ and $B$, continuing from the equations in Section 3.

Third derivatives:

\[
\begin{align*}
\frac{\partial^3 F}{\partial a_{ab} \partial a_{cf} \partial a_{op}} &= 0 \\
\frac{\partial^3 F}{\partial b_{cd} \partial a_{cf} \partial a_{op}} &= 2w_{ec}^2 \delta_{eo} \left( \delta_{df} b_{cf} + \delta_{dp} b_{cf} \right) \\
\frac{\partial^3 F}{\partial a_{ao} \partial b_{gh} \partial a_{op}} &= 2w_{ao}^2 \delta_{ao} \left( \delta_{bh} b_{gh} + \delta_{hp} b_{gh} \right) \\
\frac{\partial^3 F}{\partial b_{cd} \partial b_{gh} \partial a_{op}} &= 2w_{ag}^2 \delta_{eg} \left( \delta_{bp} a_{oh} + \delta_{hp} a_{oh} \right) \\
\frac{\partial^3 F}{\partial a_{ab} \partial a_{ef} \partial b_{qr}} &= 2w_{aq}^2 \delta_{ae} \left( \delta_{br} b_{qf} + \delta_{fr} b_{qf} \right) \\
\frac{\partial^3 F}{\partial b_{cd} \partial a_{ef} \partial b_{qr}} &= 2w_{cq}^2 \delta_{cq} \left( \delta_{df} a_{ef} + \delta_{fr} a_{ef} \right) \\
\frac{\partial^3 F}{\partial a_{ao} \partial b_{gh} \partial b_{qr}} &= 2w_{aq}^2 \delta_{aq} \left( \delta_{bh} b_{ar} + \delta_{br} b_{ar} \right) \\
\frac{\partial^3 F}{\partial b_{cd} \partial b_{gh} \partial b_{qr}} &= 0
\end{align*}
\]
Forth derivatives:

\[
\frac{\partial^4 F}{\partial a_{ab} \partial a_{ef} \partial a_{op} \partial a_{st}} = 0 \\
\frac{\partial^4 F}{\partial b_{cd} \partial a_{ef} \partial a_{op} \partial a_{st}} = 0 \\
\frac{\partial^4 F}{\partial a_{ab} \partial b_{gh} \partial a_{op} \partial a_{st}} = 0 \\
\frac{\partial^4 F}{\partial b_{cd} \partial b_{gh} \partial a_{op} \partial a_{st}} = 2 w^2_c \delta_{eg} \delta_{os} (\delta_{dp} \delta_{ht} + \delta_{hp} \delta_{dt}) \\
\frac{\partial^4 F}{\partial a_{ab} \partial a_{ef} \partial b_{qr} \partial a_{st}} = 0 \\
\frac{\partial^4 F}{\partial b_{cd} \partial a_{ef} \partial b_{qr} \partial a_{st}} = 2 w^2_c \delta_{cq} \delta_{es} (\delta_{dp} \delta_{rt} + \delta_{fr} \delta_{dt}) \\
\frac{\partial^4 F}{\partial a_{ab} \partial b_{gh} \partial b_{qr} \partial a_{st}} = 2 w^2_a \delta_{ag} \delta_{qs} (\delta_{bh} \delta_{rt} + \delta_{br} \delta_{ht}) \\
\frac{\partial^4 F}{\partial b_{cd} \partial b_{gh} \partial b_{qr} \partial a_{st}} = 0 \\
\frac{\partial^4 F}{\partial a_{ab} \partial a_{ef} \partial a_{op} \partial b_{uv}} = 0 \\
\frac{\partial^4 F}{\partial b_{cd} \partial a_{ef} \partial a_{op} \partial b_{uv}} = 2 w^2_c \delta_{eo} \delta_{cu} (\delta_{dp} \delta_{pv} + \delta_{dp} \delta_{pv}) \\
\frac{\partial^4 F}{\partial a_{ab} \partial b_{gh} \partial a_{op} \partial b_{uv}} = 2 w^2_a \delta_{ao} \delta_{qu} (\delta_{bh} \delta_{pv} + \delta_{br} \delta_{pv}) \\
\frac{\partial^4 F}{\partial b_{cd} \partial b_{gh} \partial a_{op} \partial b_{uv}} = 0 \\
\frac{\partial^4 F}{\partial a_{ab} \partial a_{ef} \partial b_{qr} \partial b_{uv}} = 2 w^2_a \delta_{ae} \delta_{qu} (\delta_{br} \delta_{fv} + \delta_{fr} \delta_{bv}) \\
\frac{\partial^4 F}{\partial b_{cd} \partial a_{ef} \partial b_{qr} \partial b_{uv}} = 0 \\
\frac{\partial^4 F}{\partial a_{ab} \partial b_{gh} \partial b_{qr} \partial b_{uv}} = 0 \\
\frac{\partial^4 F}{\partial b_{cd} \partial b_{gh} \partial b_{qr} \partial b_{uv}} = 0
\]

Particularly note that the fourth derivatives are functions of the weights only.
Bibliography


