This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Trust No One: Low Rank Matrix Factorization Using Hierarchical RANSAC

Magnus Oskarsson Kenneth Batstone Kalle Åström
Centre for Mathematical Sciences
Lund University, Sweden
magnuso@maths.lth.se *

Abstract

In this paper we present a system for performing low rank matrix factorization. Low-rank matrix factorization is an essential problem in many areas, including computer vision with applications in affine structure-from-motion, photometric stereo, and non-rigid structure from motion. We specifically target structured data patterns, with outliers and large amounts of missing data. Using recently developed characterizations of minimal solutions to matrix factorization problems with missing data, we show how these can be used as building blocks in a hierarchical system that performs bootstrapping on all levels. This gives a robust and fast system, with state-of-the-art performance.

1. Introduction

We will in this paper address the problem of robust estimation of low rank factorizations of matrices with missing data and outliers. Many problems in geometric computer vision can be formulated as such. Two examples are

- Affine structure-from-motion (SfM), where the observation matrix containing feature tracks can be factorized into the camera motions and the 3D structure.
- Photometric stereo, where the directions of light sources and the surface normals are separated by factorizing the measurement matrix composed of pixel intensities under a Lambertian model.

Other applications can be found in [2, 4, 12, 36]. These problems often lead to measurement matrices with highly structured data, in terms of which measurements that are available. In this paper we specifically target problems that exhibit such structured data patterns. Without missing data, the optimal solution to low rank matrix factorization, under the L^2-norm, is given by truncating the singular value decomposition of the measurement matrix, see [10]. When there are measurements missing in the data, there is no closed form solution to the L^2-norm minimization problem. The Wiberg algorithm [35] was the first method to handle missing data. A modified version of the Wiberg algorithm was presented in [29]. In [5], a damped Newton method is proposed to handle the missing data. If there are gross outliers in the data, optimizing the L^2-norm can give poor results. In [1], Aanaes *et al*. proposed an iteratively re-weighted least squares approach to optimize the objective function for robustness to outliers. Using more robust norms to outliers was considered in [20], where algorithms based on alternating optimization under the Huber-norm and the L^1-norm were introduced. Eriksson and Hengel generalized the Wiberg algorithm in [11], to using the L^1-norm. Sometimes, extra constraints can be posed on the factorization matrices. In [6, 37] constraints that the solution should lie on a certain manifold are considered and incorporated in the formulation. Due to the non-convexity of the matrix factorization, most methods mentioned above are based on alternating optimization, and are prone to get trapped in local minima. To address this issue, several works, such as [9, 8, 25, 12, 30] re-formulate the problem to minimize the convex surrogate of the rank function – the nuclear norm. This makes it possible to use convex optimization to find the global optimum of the approximated objective function. These approaches can handle the problems when the rank is not known a priori. However, for applications with a given rank, the nuclear norm based methods usually perform inferior to the bilinear formulation-based methods [7]. The convex formulations often have problems with very high amounts of missing data and outliers. A way of handling highly structured data matrices is to divide the whole matrix into overlapping sub-blocks and combine the sub-block solutions, see [27, 17, 24, 23, 16]. Most of these methods do not consider both outliers and missing data at the same time. There are a number of works that target specific computer vision applications for incomplete data. Examples are relative orientation problems, [18, 31], batch structure from motion estimation, [13], multi-dimensional scaling, [32], and shape estimation, [19]. It has also been

*This work has been financed by ELLIIT, MAPCI and eSSENCE.

shown that the specific problem of structure from motion with missing data is NP-hard, [28].

In this paper we largely build upon the work in [16] where minimal solvers for low rank factorizations of matrices with missing data were introduced. The emphasis was on how to analyze, describe and solve minimal problems. In this paper we address a number of algorithmic challenges (speed, accuracy, avoidance of local minima, robustness to outliers). Our contribution is a system that estimates a low rank factorization of a measurement matrix with, large amounts of missing data, in highly structured data patterns. It is based on bootstrapping minimal solvers, which gives speed and robustness to outliers. Running the solvers in a hierarchical manner gives tractable behaviour for larger input matrices, and easy parallelization. The system makes it possible to add additional constraints on the solution, throughout the pipeline.

2. Problem formulation

We will look at the problem of finding a low rank matrix approximation to a given matrix X. Any rank K matrix X of size $M \times N$ can be written as $UV = X$, where U is a rank K matrix of size $M \times K$ and V is a rank K matrix of size $K \times N$. If X represents measurements of something, that in theory should have rank K, we can not expect this equation to hold exactly due to noise. We could then instead look at the norm of the residual between the measurement matrix X and the model UV, *i.e.*

$$e = \|X - UV\|_F. \tag{1}$$

In many cases one does not have access to all measurements, *i.e.* not all entries of X are known. We can then represent which measurements are known by the *index matrix* W of size $M \times N$, where the entries are either zero, if the corresponding measurement is unknown, or one if the corresponding measurement is known. We can then write the corresponding residual norm as,

$$e = \|(X - UV) \odot W\|_F, \tag{2}$$

where $\odot$ represents element-wise multiplication. In addition to measurement noise, one can also have gross outliers in the data, in some applications. In this case minimizing an error norm based on an L^2-distance often gives bad results. In order to decrease the influence of the outliers, robust norms are used, such as the L^1-norm or the Huber norm. In [3] a more refined loss function is proposed. If we assume that the inlier residuals approximately follow a Gaussian distribution, whereas outlier residuals have approximately uniformly distributed errors, then this leads to the loss function

$$l(r) = -\log(c + \exp(-r^2)), \tag{3}$$

where r is the residual error. Truncating the squared error

$$l(r) = \begin{cases} r^2 & \text{if } |r| \leq \epsilon \\ \epsilon^2 & \text{otherwise} \end{cases} \tag{4}$$

gives a good approximation. If we denote the residual matrix $R = (X - UV) \odot W$, with entries r_{ij}, we can formulate our problem as

$$\underset{U,V}{\text{minimize}} \quad \sum_{i,j} l(r_{ij}). \tag{5}$$

The final error depends on ϵ which we set as a parameter in our algorithm. This is the bound that differentiates an inlier from an outlier measurement.

3. Matrix factorization with missing data

We will use the characterization of low rank matrix factorization problems that was described in [16]. For completeness and readability, we will in short describe some of the results from that paper.

A key question is to study for which index matrices W of type $M \times N$, the problem of calculating the rank K matrix $X = UV$ is minimal and well-defined. For this we introduce the manifold $\Omega = R^{M \times K} \times R^{K \times N}$ of possible solutions $z = (U, V)$. The solution set $\Omega_{X,W} \subset \Omega$ for a given data matrix X and a index matrix W is defined as

$$\Omega_{X,W} = \{z = (U, V) \in \Omega | W \odot (X - UV) = 0\}. \tag{6}$$

Typically if the index matrix W has to many non-zero elements, then there are too many constraints in $W \odot (X - UV) = 0$ and the solution set $\Omega_{X,W}$ is empty for general X. If there is a solution $X = UV$, then we are interested to known if the solution is unique up to the so called *Gauge freedom* $z = (U, V)$ vs $z = (UH, H^{-1}V)$, where H is a general invertible $K \times K$ matrix.

Assume that input data matrix $X = U_0 V_0$ of size $M \times N$ has been generated by multiplying matrices U_0 of size $K \times M$ and V_0 of size $K \times N$. Assume also that both of these matrices are general in the sense that all $K \times K$ submatrices of both U_0 and V_0 have rank K. Furthermore assume that $M \geq K$ and $N \geq K$. An index matrix W is said to be *rigid*, if the solution set $\Omega_{X,W}$ locally around the point $z_0 = (U_0, V_0)$ only consists of the set $z(H) = \{(U_0 H, H^{-1} V_0) | H \text{ invertible } K \times K \text{ matrix}\}$.

Since we are assuming that every sub-minor of U_0 and V_0 has full rank, one may actually fix the Gauge freedom by keeping one such sub-minor fixed. For example we could study the solutions for the points $z = (U, V)$ such that the first K rows of U are equal to those of U_0.

For two index matrices W_1 and W_2 of the same size we say that $W_1 \leq W_2$ if the inequality holds for every element. We say that $W_1 < W_2$ if $W_1 \leq W_2$ and $W_1 \neq W_2$. It is

trivial to see that if W is rigid and if $W \leq W'$ then W' is also rigid. It also can be shown that if W' is rigid and overdetermined, then there is at least one $W < W'$ that is rigid and minimal. We say that an index matrix W is *minimal* if it is rigid and satisfies $\sum_{ij} W(i,j) = MK + NK - K^2$. For a minimal index matrix W and for general data X the solution set $\Omega_{X,W}$ consists of a finite number of points n_W up to the gauge freedom.

3.1. Henneberg extensions

We will now describe how to generate the minimal problems. The inspiration comes from rigid graph theory, where the Henneberg construction is used to generate the Laman graph, see [22, 14]. The idea is that one starts with the smallest minimal index matrix, and by a series of extensions generate every minimal index matrix. For the rank K problem the smallest index matrix is a matrix of size $K \times K$ consisting of ones only.

There are both *constructive* extensions and *non-constructive* extensions. For a constructive extension from W to W', one can infer the number of solutions $n_{W'}$ from n_W and construct the solver, denoted by $f_{W'}$ from f_W. For non-constructive extensions, it can be shown that W is minimal if and only if W' is minimal. However, we can in general neither infer the number of solutions $n_{W'}$ from n_W nor derive a solver $f_{W'}$ from f_W. Certain of these constructive extensions are particularly fast and efficient. The simplest one is as follows.

Given a minimal index matrix W for a rank-K problem of size $M \times N$, an extended minimal index matrix W' of size $M \times (N+1)$ is formed by adding a column with exactly K elements set to one. The number of solutions are identical, i.e. $n_W = n_{W'}$. Extending an algorithm from f_W to $f_{W'}$ is straightforward. A similar extension can be done by adding a row with K indices.

3.2. Henneberg reductions

There is also a simple recursive method to check if an index matrix W can be generated using only Henneberg 1 extensions. The procedure is as follows. Start with an index matrix of size $M \times N$. If $M = N = K$ then the index matrix is minimal if and only if the matrix consists only of ones. If M or N is larger than K, we calculate the minimal number of ones for a row or column. If this number is less than K, then it can be shown that the index set in question is non-rigid. If this number is larger than K it can be shown that the index set (if minimal) cannot be generated by Henneberg 1 extensions only. Finally if the number is K, then we can remove the row (or column) with exactly K ones and study this index matrix, which now is of smaller size.

The algorithm terminates after at most $M + N - 2K$ steps. After running the algorithm we determine if the index set is minimal and can be constructed by a series of Henneberg 1 extensions. But we also obtain the pattern of extensions. Thus we obtain an efficient method of calculating the unique solution (U, V) from a data matrix X so that $W \odot (X - UV) = 0$.

3.3. Glues

Assume that the solutions to two sub-problem $\{W_1, X_1\}$ and $\{W_2, X_2\}$ are given by $\{U_1, V_1\}$ and $\{U_2, V_2\}$ respectively. To construct the solution to $\{W, X\}$, the idea is to find a transformation matrix $H \in \mathbb{R}^{K \times K}$ to transform the subspace U_2 to the same coordinate framework as the subspace U_1. Using this transformation we have

$$U_2 V_2 = (U_2 H)(H^{-1} V_2). \tag{7}$$

Now $U_2 H$ and $H^{-1} V_2$ are in the same coordinate framework as U_1 and V_1 respectively. The remaining problem is to solve for H. We have the following constraint, that states that U_1 and $U_2 H$ should coincide for the overlapping rows as

$$U_1(\mathtt{I}_{12}, :) = U_2(\mathtt{I}_{12}, :)H, \tag{8}$$

where $\mathtt{I}_1$ and $\mathtt{I}_2$ denotes the indices of overlapping rows in U_1 and U_2 respectively and $U(\mathtt{I}, :)$ denotes the sub-matrix of U by taking the rows given by $\mathtt{I}$. Similarly we have the overlapping constraints for V_1 and $H^{-1} V_2$ as

$$H V_1(:, \mathtt{J}_{12}) = V_2(:, \mathtt{J}_{12}), \tag{9}$$

where $\mathtt{J}_1$ and $\mathtt{J}_2$ denotes the indices of overlapping columns in V_1 and V_2 respectively.

If we have enough constraints from (8) and (9), H can be solved linearly. Two examples are if there are at least K overlapping rows or K overlapping columns. For the cases where the overlap doesn't give sufficiently many constraints, we need some extra constraint outside W_1 and W_2 to solve for the transformation matrix H.

4. Building blocks

In this section we will describe the basic components that are used in our matrix factorization method. Our full system will be described Section 5.

4.1. Initializing solutions

We use RANSAC to find small initial solutions. We choose a sub-problem to problem 5, by choosing a sub-matrix W_i of W. We further randomly select a minimal number of measurements, represented by $W_m < W_i$. These index-matrices have corresponding measurement matrices X_i and X_m. Even though we have chosen a minimal subset of measurements, this need not represent a well posed minimal problem. In order to check this, we perform Henneberg reductions as described in Section 3.2. If W_m indeed represents a minimal problem, we can, if we have a solver for

this case, solve the corresponding matrix factorization of X_m. This gives a minimal solution (U_m, V_m). We can now look at how well this solution matches the other measurements in W_i by looking at the residuals $(U_m V_m - X_i) \odot W_i$. Repeating this process gives a set of initial good solutions.

4.2. Extending solutions

If we have a solution, represented by (U_i, V_i) we can minimally extend this solution row- or columnwise using Henneberg-1 extensions, for every column (or row) that has at least K measurements. For every such column a, we randomly select K rows that are represented in the corresponding index sub-matrix W_i, and use the Henneberg-1 extension to find the new column v_a so that $\bar{V}_i = \begin{bmatrix} V_i & v_a \end{bmatrix}$. To handle outliers we check how many of the measurements that fit this new $\bar{U}_i$. If we have a substantial enough number of inliers we keep this solution, otherwise we repeat the process a number of times.

4.3. Glueing solutions

If we have two solutions, represented by (U_i, V_i) and (U_j, V_j) we can, depending on the overlap of the two solutions, glue these solutions into one using the methods described in Section 3. Basically this can be done if there is enough information to estimate the $K \times K$ transformation matrix H so that $U_i H$ and $H^{-1} V_i$ are given in the same coordinate frame as U_j and V_j. Using a randomly selected minimal set of measurements to estimate H gives a new solution (U_k, V_k) where U_k is the union of $U_i H$ and U_j, and V_k is the union of $H^{-1} V_i$ and V_j. To handle outliers we check how many inliers we get for this new solution. Again, if we have a substantial enough number of inliers we keep this solution, otherwise we repeat the process a number of times.

4.4. Refining solutions

Since we use minimal solvers, and extend these solutions iteratively, we need to refine our solutions in order to avoid error propagation. We non-linearly refine our solutions, by minimizing (5) iteratively using Gauss Newton descent. We handle the truncation, at each step, by only optimizing over the inlier set, and then updating the inlier set using the new estimate of U and V. Since the error on the inlier set is quadratic in U and V the derivatives with respect to U and V are easily obtained.

5. Sampling scheme

Using the building blocks from the previous section, we can now describe our full sampling scheme. The basic idea behind our method is that we will have several solutions competing against each other. These solutions will expand and merge, but at all steps we will try to be robust against outliers and inlier errors, so that we do not propagate errors. We do this by random sampling at all instances.

We assume that we have four functions available: INIT, EXTEND, GLUE and REFINE, that do initialization, extensions, glues and non-linear refinement respectively as described in Sections 4.1-4.4. We start by initializing a number of seed solutions. For each solution i, we have U_i and V_i. We then, for each of these seed solutions, attempt to extend it and refine it. If two solutions overlap, we try to glue them together. We repeat this procedure until we have at least one solution that covers the whole data matrix X. Sometimes the errors of a solution will grow during this process, and we remove solutions that have a residual norm larger than some fixed threshold. This means that we could end up with an empty solution set. In this case we re-initialize a number of seed solutions. The steps of our procedure are summarized in Algorithm 1. There

Algorithm 1 Matrix factorization sampling scheme

Given an $M \times N$ data matrix X with index matrix W,
initialize a solution set S = INIT,
$(S = \{S_1, S_2, \ldots, S_n\}, S_i = (U_i, V_i))$.
while no S_i is of size $M \times N$ **do**
  for all i **do**
    S_i = EXTEND S_i (column-wise)
    S_i = EXTEND S_i (row-wise)
    S_i = REFINE S_i
    if $\|(U_i V_i - X_i) \odot W_i\|_F > C$ **then**
      $S = S \setminus S_i$
    end if
  end for
  for all overlapping S_i, S_j **do**
    S_k = GLUE S_i, S_j
    $S = S \cup S_k$
  end for
  if $S = \emptyset$ **then**
    Initialize n new seed solutions S = INIT.
  end if
end while

is of course no guarantee that Algorithm 1 will converge, but given a well posed problem and relevant parameter settings, it is our experience that it will. We terminate after a fixed number of iterations or when we have at least one solution that covers the whole measurement matrix. We will in the experimental section validate this, on both synthetic and real data. There are a number of parameters that need to be set in order for the algorithm to work properly. The most important one is the error bound, *i.e.* the reprojection error that differentiates between an inlier and an outlier. In order to increase the robustness, we have introduced an absolute threshold on the number of inliers for the EXTEND, and GLUE functions. For each row and column the num-

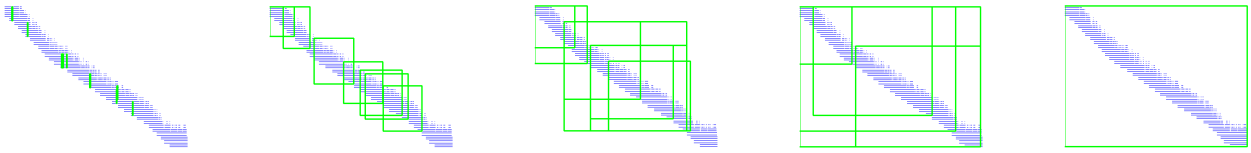

Figure 1. Results from running Algorithm 1 on the Dinosaur experiment (See Section 6.1 for details). Measured data points in blue, and the extent of the solutions are depicted as green boxes. The figure shows from left to right how the solution set evolves.

ber of inliers should exceed this threshold. We have used the rank plus a small integer as threshold. We also need to set the number of RANSAC iterations for each of these functions, but this will mainly affect the total running time. If we assume structured data, it makes sense to smooth the index matrix W with a two-dimensional Gaussian, to obtain W_{sm}. We then sample initialization matrices guided by W_{sm} as a probability measure. Sampled positions are removed from W_{sm} to avoid multiple initial points. In [16] they do a manual block subdivision (with relatively large blocks *e.g.* 50×50). This means that the model that is fitted is very large (*e.g.* $50K + 50K - K^2$ parameters for a rank K problem) algorithm. In our approach there is no need for any explicit block subdivision, and the initial models that are fitted are much smaller. This gives a much more tractable and robust algorithm.

6. Experiments

We have applied our method to a number of different matrix factorization problems, in order to show the usefulness of it in terms of robustness, accuracy and speed.

We have compared our results to a number of state-of-the-art methods for low rank matrix factorization, namely the method of Larsson *et al*., [24], the method of Jiang *et al*., [16], the Truncated Nuclear Norm Regularization (TNNR-ADMM) [15], OptSpace [21], the damped Wiberg algorithm using the implementation of Okatani *et al*. [29], and the L^1Wiberg algorithm, [11], using the C++ implementation from [33]. In the results we have included the relevant comparisons, in order to make the tables and graphs more readable. It has been previously reported (in [16]) that the methods of [15] and [21] perform much worse for structured data patterns with high amounts of missing data. This is also our experience end hence we have omitted these results. Most of the methods do not handle outliers, and in the absence of outliers the Wiberg algorithm gives best accuracy. For this reason we mainly focus on comparison with the L^1- and L^2-versions of the Wiberg algorithm in the synthetic experiments in Section 6.3. All tests were conducted on a desktop computer running Ubuntu, with an Intel Core i7 3.6 GHz processor. The Matlab implementation of our method is available at https://github.com/hamburgerlady/miss-ranko.

6.1. Affine structure from motion

Given a full data matrix and an affine camera model it is possible to solve the structure from motion problem using factorization. The standard way of doing this is by first removing the centroid of each point in the images. This leads to a rank 3 factorization problem. When there is missing data, we can neither use SVD for factorization nor remove the centroid since this is not known. We can still write the problem as a rank 4 matrix factorization problem with missing data, see *e.g.* [34]. An affine camera is of the form

$$P_i = \begin{bmatrix} A_i \\ B_i \\ 0\ 0\ 0\ 1 \end{bmatrix}. \tag{10}$$

If we collect the N (homogeneous) 3D points in the $4 \times N$ matrix V, one usually writes the rank 4 problem as $UV = X$, where U is the $2M \times 4$ matrix containing the stacked camera rows A_i and B_i, and the $2M \times N$ matrix X contains the x and y coordinates of image points. However, solving this rank 4 matrix factorization problem ignores the fact that the last row of the camera matrices should be equal to $[0\,0\,0\,1]$, and that the last coordinate of each homogeneous point should be equal to one. In our model we include this constraint by simply adding a row in the measurement matrix with just ones. We have found this to be a very powerful constraint, since we know it should hold even for missing data, and we use it throughout our pipeline. If the constraint is fulfilled, we can simply upgrade to the affine model by $U^a = UH$ so that the last row of U^a is equal to $[0\,0\,0\,1]$. We have run Algorithm 1 on the well known Dinosaur sequence. This is a sequence that contains very little outliers, but a large amount of missing data (88%). Even though the underlying structure from motion can be (and has been) solved using a multitude of methods, the Dinosaur sequence works well as a benchmark problem for matrix factorization. We have compared our results with to those of Larsson *et al*., [24]. In their work they also did experiments on using the nuclear norm and we have included these results here. In Jiang *et al*., [16] it was reported that the Truncated Nuclear Norm Regularization (TNNR-ADMM) [15]

Dataset	Algorithm			
	[16]	[24]	[29]	*Proposed*
Dino ∼300 pts	99	73.3	28.0	**17.0**
Dino ∼2000 pts	-	-	144.1	**48.45**
Linnaeus ∼2000 pts	-	-	580000	**380.5**
Linnaeus ∼4000 pts	-	-	-	**543.4**

Table 1. Comparison of results from the Dinosaur and Linnaeus reconstruction experiments. The table shows the Frobenius errors using the proposed method compared to the Nuclear norm minimization, the method of Larsson *et al*. [24], and the damped Wiberg algorithm [29].

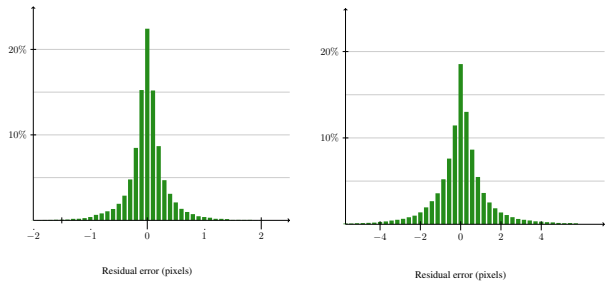

Figure 2. Left: Histogram of the final (non-truncated) residuals from the Dinosaur experiment using our proposed method. The results are from the larger dataset, with approximately 2000 3D points. Right: Histogram of the final (non-truncated) residuals from the Linnaeus experiment using our proposed method.

and OptSpace [21] failed to recover the 2D tracks from the Dinosaur sequence. This is also our experience, as we consistently failed to recover a reasonable solution using these methods. In addition we have also run the damped Wiberg algorithm using the implementation of Okatani *et al*. [29]. In Table 1 the Frobenius norm of the final factorizations are shown. The average running time for the Wiberg algorithm was 144 seconds, compared to around 3 seconds for our method. We have also run our method on a larger point set. For this set we didn't have access to results from [24]. We have run our method multiple times, and we always end up in the same optimum. Here the running time for the Wiberg algorithm was 4087 seconds, compared to 5.4 seconds for our method. A histogram of the residuals (in pixels) from our reconstruction is shown in Figure 2. Using the calibration of the projective camera model, we can upgrade our affine reconstruction to an orthographic. The resulting calibrated affine reconstruction is shown to the right in Figure 3. In a second experiment, we recorded a sequence of a statue of Carl Linnaeus. We extracted Harris corner points, and tracked these using the *Kanade-Lucas-Tomasi* (KLT) tracker [26]. This resulted in a sequence with 86 images

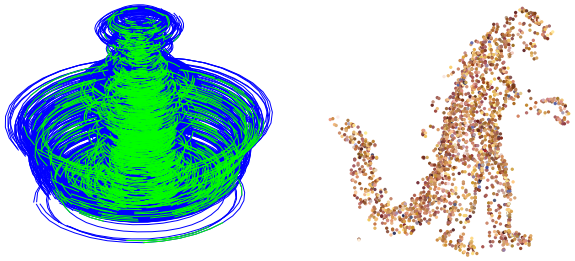

Figure 3. Results from the Dinosaur experiment, with approximately 2000 points. Left: Measured tracks (green) and reconstructed tracks (blue). Right: The calibrated affine reconstruction.

Figure 4. The figure shows three frames of the Linnaeus sequence. Also shown are the tracked Harris points (orange circles) and the reprojected points (white dots) using the proposed method. For visibility a random subset of 250 points are shown

and a total of 3977 3D points. This sequence also contains closed loops, *i.e*. points were tracked from the last frames to the first. Three frames of the sequence can be seen in Figure 4. The results of running our algorithm on this sequence can be seen in Table 1. The running time for the full dataset was 39.8 s. The extracted points (orange circles) and the reprojected points (white dots) can be seen in Figure 4. The sequence constains approximately 1% outliers. As for the Dinosaur sequence, the Truncated Nuclear Norm Regularization (TNNR-ADMM) [15] and OptSpace [21] failed to recover reasonable 2D tracks from the Linnaeus sequence. We were not able to run the damped Wiberg algorithm on the full dataset, but the results for a subset of around half the points can be seen in Table 1. The running time for our method on the smaller Linnaeus sequence was 10.4 s compared to 1068 s for the Wiberg algorithm. We didn't have access to results from [24].

6.2. Linear shape basis estimation

If we have non-rigid structures in a scene, a linear shape basis can be used to model the deformations. The underlying assumption is that the non-rigid deformation of the object can be represented as a linear combination of a set of

Dataset	Algorithm			
	[24]	[16]	[29]	*Proposed*
Book	0.3522	**0.1740**	**0.1740**	**0.1740**
Hand	0.8613	0.6891	**0.2802**	**0.2802**
Book-10%	8.0436	0.1772	**0.1534**	**0.1534**
Hand-10%	1.5495	0.7297	**0.2634**	**0.2634**

Table 2. Comparison of the result on linear shape basis estimation using the *Book* and *Hand* dataset. The second experiment contains an additional 10% missing data. The table shows the Frobenius errors using the proposed method compared to the methods of [24], [16] and the damped Wiberg algorithm [29].

shapes. Typically the size of the shape basis is much smaller than either the number of frames, or the tracked points, so the measurement matrix containing the point tracks can be factorized into a coefficient matrix and a shape basis matrix.

For our experiments we used the datasets from [24], *Book* and *Hand*. In these experiments the image points are tracked using a standard *Kanade-Lucas-Tomasi* (KLT) tracker [26]. Due to occlusions, the tracker fails after a number of frames for a subset of points, which leads to missing data. To compare with the results in [16] we use the same setup as they do, using a subset of 42 frames with 60 tracked points from the *Book* and 38 frames with 203 points from the *Hand* dataset. We then find rank-3 and rank-5 factorizations of the two datasets respectively. We ran our algorithm, and also the Wiberg algorithm using the implementation in [29]. The results can be seen in Table 2. The Wiberg algorithm was initialized randomly. Our method and the Wiberg minimization achieve the same optima, which are slightly better than the other methods. The reason is probably that the Wiberg and our method finds the same optimum– that we believe is the global one – since the set is practically outlier free. The other methods do not in this case find an equally good optimum. For these smaller problems the Wiberg algorithm works well, but for larger problems it becomes intractable in terms of running time, as described in Section 6.1.

6.3. Performance tests

We have conducted a number of synthetic tests to test the performance of our method. The basic setup was done by randomly drawing two matrices $U_{M\times K}$ and $V_{K\times N}$. The product $X_0 = UV$ was then perturbed with Gaussian noise, *i.e.* $X = X_0 + \epsilon_{ij}$ with $\epsilon_{ij} \in N(0, \sigma)$. A band diagonal matrix $W_{M\times N}$ was used to prescribe which measurements were available. The bandwidth b for the matrix W (with entries w_{ij}) is defined using $w_{ij} = 0$ for $j < i-b$ or $j > i+b$. Finally, a certain percentage of the seen measurements were replaced with entries drawn randomly from a uniform distribution, to simulate gross outliers. In a first experiment we tested the sensitivity to the proportion of outliers in the measurement matrix. We used a 300×300 measurement matrix X, with bandwidth 20. The entries were approximately between minus one and plus one, with Gaussian noise with standard deviation $1e-3$. A certain percentage of the measurements were then replaced with gross outliers. We ran Algorithm 1 and compared the results with the L^1-Wiberg algorithm, [11], using the C++ implementation from [33]. The results can be seen to the left, in Figure 5. We have also constructed an oracle ground truth solution to compare the results. This solution was attained by running the non-linear refinement, using the ground truth inlier set, and the ground truth U and V as starting solution. This will in general give a better optimum than U and V. One can see that both the tested algorithms perform on par. The average running times are depicted in Figure 5. The middle graph shows both algorithms in the same plot, and the right hand plot shows a magnification of the running times for the proposed algorithm. The L^1-Wiberg algorithm has very unattractive time complexity in terms of the size of the input measurement matrix, and we failed to run it for larger sizes than 200×200. Our method works well for moderate amounts of outliers, but as the outlier percentage increases, the RANSAC initialization will take longer and longer time, and for this test the break-down point for our method was around 20%. In a second experiment we used a similar setup, but instead of varying the outlier rate, we varied the size of the input measurement matrix X. Here we used a fixed outlier ratio of 5%. In Figure 6 the results can be seen. We show two versions of our algorithm, with and without using the GLUE step. The left of the figure shows the truncated L^2-error for the two versions, compared with the oracle ground truth solution, obtained in the same way as in the previous experiment. The right shows the average running times, as a function of the number of rows (equal to the number of columns) in the measurement matrix. One can see that for smaller problems, there is no need for the GLUE. For larger problems using the GLUE leads to (in this case) near linear time complexity. As can be seen from the error plot, using the GLUE method doesn't lead to any error accumulation. We have also investigated our algorithm's dependence on the bandwidth of the measurement matrix. Using a similar setup as in the previous experiments, we constructed measurement matrices with varying bandwidth. We did not include any outliers in the data, and compared our results with the damped Wiberg algorithm. The results can be seen in Figure 7. Both algorithms give final norms very close to the oracle solution, but for smaller bandwidths the Wiberg algorithm has worse convergence. For very small bandwidths our algorithm becomes unstable, and the RANSAC loops will take excessive amounts of time. In this case, the breakdown point for our algorithm was for a bandwidth around 5. This will depend on the rank

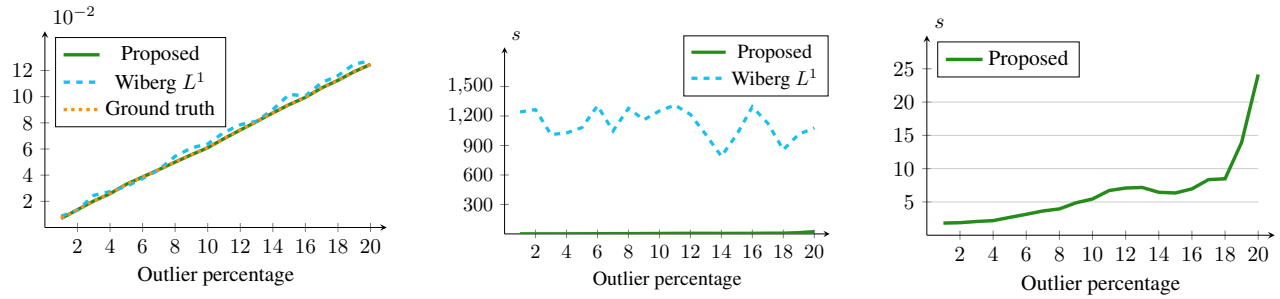

Figure 5. Results from the outlier test, as described in Section 6.3. Left: The graph shows the mean L^1error for the proposed method compared to the L^1Wiberg algorithm, [11, 33] and the oracle ground truth solution, as functions of the outlier rate. Middle: The average running times, as functions of the outlier rate, for the proposed method compared to the L^1Wiberg algorithm, [11, 33]. Right: A magnification of the timing results for the proposed method.

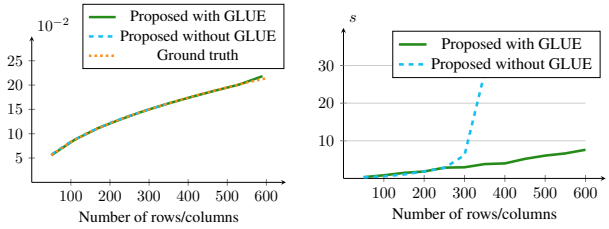

Figure 6. Timing results from the size test, as described in Section 6.3. The left graph shows the truncated L^2-error, as functions of the image size, for the proposed method (with and without using GLUE) compared to the oracle ground truth optimum. The right graph shows the average running times for the proposed method, with and without using GLUE.

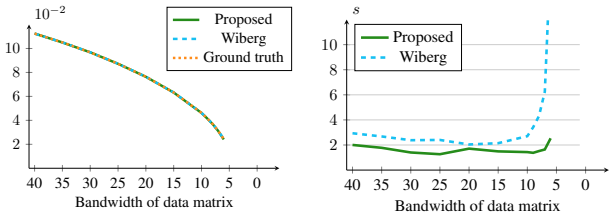

Figure 7. Timing results from the missing data test, as described in Section 6.3. The left graph shows the L^2-error, as functions of the bandwidth of the data, for the proposed method compared to the damped Wiberg algorithm and oracle ground truth optimum. The right graph shows the average running times for the proposed method, compared to the damped Wiberg algorithm.

of the solution, which in this case was 4.

Algorithm 1 is highly parallelizable. We have conducted a simple timing experiments using our Matlab implementation. We simply change the `for`-loop to Matlabs `parfor`-loop. This runs the extensions and refinement of the initial solutions in parallel, but not the initialization. The initialization could of course also be run in parallel, but for most of our conducted experiments the initialization takes a smaller fraction of the total running time. The average running times, using different number of parallel cores, are shown in Table 3. We get slightly less than linear speed-ups.

	Number of cores			
Dataset	1	2	3	4
Book	0.793s	0.470s	0.382s	0.369s
Hand	28.6s	16.1s	12.2s	10.3s
Dino ∼2000 points	5.38s	3.61s	3.02s	2.19s

Table 3. Timings for the linear shape basis estimation using the *Book* and *Hand* dataset. The results are based on running *parfor* in Matlab with different number of cores on an Intel Core i7 3.6 GHz processor.

7. Conclusion

We have in this paper presented a method for doing robust low rank matrix factorization for structured data patterns. It can handle very large amounts of missing data, and moderate amounts of outliers. It gives results on par, or better than, using the L^1-Wiberg algorithm or the damped L^2-Wiberg algorithm, with substantial speed-up. The presented method is also trivially parallelizable. Future work includes investigating two interesting properties of our method, that we have not exploited in any detail in this paper. Firstly we have the ability to solve for multiple models in the data. For instance, if we have two rigid motions in a structure from motion sequence, we could – at least in theory – run our algorithm and find the two solutions corresponding to the two motions. Another property, that easily can be incorporated in our framework, is the ability to add additional constraints on the solution space, *e.g.* for the affine structure from motion setting we can constrain the pair-wise rows of U to be orthogonal to model a scaled orthographic camera.

References

[1] H. Aanaes, R. Fisker, K. Åström, and J. Carstensen. Robust factorization. *Trans. Pattern Analysis and Machine Intelligence*, 2002. 1

[2] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision*, 72(3):239–257, 2007. 1

[3] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, USA, 1987. 2

[4] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Conf. Computer Vision and Pattern Recognition*, 2000. 1

[5] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 316–322, Washington, DC, USA, 2005. IEEE Computer Society. 1

[6] A. D. Bue, J. M. F. Xavier, L. de Agapito, and M. Paladini. Bilinear modeling via augmented lagrange multipliers (balm). *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(8):1496–1508, 2012. 1

[7] R. Cabral, F. D. la Torre, J. P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *Int. Conf. Computer Vision*, 2013. 1

[8] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of ACM*, 2009. 1

[9] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008. 1

[10] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. 1

[11] A. Eriksson and A. Van Den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l 1 norm. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 771–778. IEEE, 2010. 1, 5, 7, 8

[12] R. Garg, A. Roussos, and L. Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1272–1279. IEEE, 2013. 1

[13] N. Guilbert and B. A. Batch recovery of multiple views with missing data using direct sparse solvers. In *British Machine Vision Conference*, 2003. 1

[14] L. Henneberg. *Die graphische Statik der starren Systeme*, volume 31. BG Teubner, 1911. 3

[15] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2117–2130, 2013. 5, 6

[16] F. Jiang, M. Oskarsson, and K. Åström. On the minimal problems of low-rank matrix factorization. In *Conf. Computer Vision and Pattern Recognition*, 2015. 1, 2, 5, 6, 7

[17] C. Julià, A. D. Sappa, F. Lumbreras, J. Serrat, and A. López. An iterative multiresolution scheme for sfm with missing data. *Journal of Mathematical Imaging and Vision*, 34(3):240–258, 2009. 1

[18] F. Kahl, A. Heyden, and L. Quan. Projective reconstruction from minimal missing data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(4):418–424, 2001. 1

[19] J. Karlsson and K. Åström. Mdl patch correspondences on unlabeled images with occlusions. In *CVPR, Computer Vision and Pattern Recognition*, 2008. 1

[20] Q. Ke and T. Kanade. Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *Conf. Computer Vision and Pattern Recognition*, 2005. 1

[21] R. H. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. *CoRR*, abs/0901.3150, 2009. 5, 6

[22] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering mathematics*, 4(4):331–340, 1970. 3

[23] V. Larsson and C. Olsson. Convex envelopes for low rank approximation. In *EMMCVPR 2015*, 2015. 1

[24] V. Larsson, C. Olsson, E. Bylow, and F. Kahl. Rank minimization with structured data patterns. In *ECCV*, 2014. 1, 5, 6, 7

[25] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of a corrupted low-rank matrices. *CoRR*, abs/1009.5055, 2009. 1

[26] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 6, 7

[27] L. W. Mackey, M. I. Jordan, and A. Talwalkar. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142, 2011. 1

[28] D. Nistér, F. Kahl, and H. Stewénius. Structure from motion with missing data is NP-hard. In *Int. Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007. 2

[29] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *Int. Conf. Computer Vision*, pages 842–849, 2011. 1, 5, 6, 7

[30] C. Olsson and M. Oskarsson. A convex approach to low rank matrix approximation with missing data. In *Scandinavian Conf. on Image Analysis*, 2011. 1

[31] M. Oskarsson, K. Åström, and N. C. Overgaard. Classifying and solving minimal structure and motion problems with missing data. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 628–634. IEEE, 2001. 1

[32] M. Oskarsson, K. Åström, and A. Torstensson. Prime rigid graphs and multidimensional scaling with missing data. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 750–755, Aug 2014. 1

[33] D. Strelow. General and nested wiberg minimization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1584–1591. IEEE, 2012. 5, 7, 8

[34] G. Wang, J. S. Zelek, Q. J. Wu, and R. Bajcsy. Robust rank-4 affine factorization for structure from motion. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 180–185. IEEE, 2013. 5

[35] T. Wiberg. Computation of principal components when data are missing. In *Proc. Second Symp. Computational Statistics*, 1976. 1

[36] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *Trans. Pattern Analysis and Machine Intelligence*, 2008. 1

[37] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust L_1-norm. In *Conf. Computer Vision and Pattern Recognition*, 2012. 1