A Combinatorial Approach to L_1-Matrix Factorization

Jiang, Fangyuan; Enqvist, Olof; Kahl, Fredrik

Published in:
Journal of Mathematical Imaging and Vision

DOI:
10.1007/s10851-014-0533-0

2015

Noname manuscript No.
(will be inserted by the editor)

A Combinatorial Approach to L_1-Matrix Factorization

Fangyuan Jiang · Olof Enqvist · Fredrik Kahl

Received: 11 March 2014 / Accepted: 21 August 2014

Abstract Recent work on low-rank matrix factorization has focused on the missing data problem and robustness to outliers and therefore the problem has often been studied under the L_1-norm. However, due to the non-convexity of the problem, most algorithms are sensitive to initialization and tend to get stuck in a local optimum. In this paper, we present a new theoretical framework aimed at achieving optimal solutions to the factorization problem. We define a set of stationary points to the problem that will normally contain the optimal solution. It may be too time-consuming to check all these points, but we demonstrate on several practical applications that even by just computing a random subset of these stationary points, one can achieve significantly better results than current state of the art. In fact, in our experimental results we empirically observe that our competitors rarely find the optimal solution and that our approach is less sensitive to the existence of multiple local minima.

Keywords L_1-Matrix Factorization · Robust Estimation · Structure-from-Motion · Photometric Stereo

Fangyuan Jiang
Lund University, Centre for Mathematical Sciences
Box 118 22100 Lund
Tel.: +46-46-222 9735
Fax: +46-46-222 8537
E-mail: fangyuan@maths.lth.se

Olof Enqvist
Chalmers University of Technology
E-mail: olof.enqvist@chalmers.se

Fredrik Kahl
Lund University
Chalmers University of Technology
E-mail: fredrik@maths.lth.se

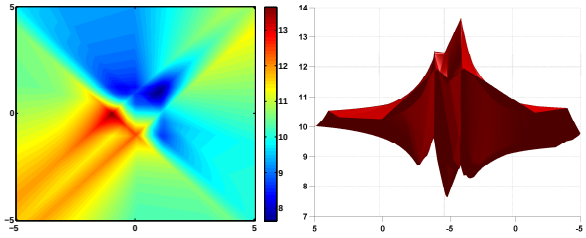


Fig. 1 The L_1-cost for fitting a 1D-subspace to a set of points in $\mathbb{R}^3$ ($m = 3$ and $r = 1$). The cost function is varied over two dimensions of U and then the optimal solution is computed for the other variables. Note that there are at least three local minima.

1 Introduction

Given an observation matrix $W \in \mathbb{R}^{m\times n}$, we are interested in the problem of factorizing W into two low-rank matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$ with $r < \min(m, n)$ such that $W \approx UV$, or effectively solving

$$\min_{U,V} \|W - UV\| . \tag{1}$$

If $\|\cdot\|$ is the standard L_2-norm, then the solution is obtained by computing a Singular Value Decomposition (SVD) of W. However, in many applications W tends to contain missing values and erroneous measurements (outliers) and therefore, a lot of work has been devoted to alternative methods that are more robust to such factors. In this paper, we analyze the factorization problem under the L_1-norm and give new theoretical insights to the problem, which in turn, will lead to a novel approach for solving it.

Factorization plays an important role in many applications in computer vision. Perhaps its most well-known application is in affine structure from motion, first studied in [33]. There are numerous extensions —

also based on factorization — to, for example, independently moving objects [16], projective structure from motion [31], non-rigid structure from motion [3,17] and articulated motion [37]. In photometric stereo, the factorization problem also appears as a subtask and SVD-based methods are common [38,27]. Another important application area is in shape modelling [15]. Here the objective is to compute a compact representation of the training shapes, which can be achieved via factorization. Often, the principal components are computed by an SVD, but this assumes no missing data and only inlier measurements.

Related Work. Missing data in low-rank matrix factorization was originally addressed in [35], then under the L_2-norm. An algorithm independent of initialization was given in [21], but the method is highly sensitive to noise. Still, it is suitable as an initialization method followed by an iterative, refinement technique. Similar approaches to the structure-from-motion problem are studied in [32,22].

An early work that aims for robustness to outliers is [1], using iteratively reweighted least squares to optimize a robust error function. A limitation is that the method requires a good initial solution, which is often difficult to obtain. The theory of robust subspace learning is further developed in [24]. In [5], a damped Newton method is proposed to solve the problem with missing data. In [6], a bilinear model is formulated under L_2 norm with the constraints that the factor matrices should lie in a certain manifold. It is solved via the augmented Lagrange multipliers method. In [23], alternating optimization is proposed for both the Huber-norm and the L_1-norm. Yet another iterative approach is proposed in [18] which can be seen as an extension of [35], but for the L_1-norm. This approach has been further generalized in [30] to handle the projective structure-from-motion problem. In [28], a damping factor is incorporated in the Wiberg method. It also experimentally showed that Newton-family minimization techniques with a damping factor lead to a top global convergence performance. The method in [40] first solves the affine factorization in L_2 norm by adding an extra mean vector in the formulation. Another recent algorithm in [39] constrained U to be column-orthogonal and added a nuclear norm regularizer to V. With the augmented Lagrangian multipliers method, it achieves a fast convergence rate. All of these algorithms are based on local optimization, and hence they risk getting stuck in local minima. The cost function may indeed exhibit several local optima as exemplified in Fig. 1. One noticeable attempt to solve the problem globally optimal is proposed in [10], which uses a branch and bound method. It proves that the globally optimal solution is obtained. However, in practice, it is only restricted to the simple problems for which the number of variables in either U or V is very small. For example, there are only 9 variables in U in one of their experiments.

Alternative approaches for tackling the low-rank factorization or low-rank approximation problems are to minimize a convex surrogate of the rank function, for example, the nuclear norm. In [9], the solution turns out to be very pleasing - only a convex optimization problem needs to be solved. The nuclear norm formulation leads to solving an SDP, which the methods in [8,25] try to solve efficiently. These approaches can handle application problems when the rank is not known a priori, for example, segmentation [11], background modeling [8] and tracking [36]. However, when applied to problems with known rank, the performance of the methods based on the nuclear norm formulation is typically worse compared with the bilinear formulation [7]. These methods assume that the missing data are sparse and the locations of missing data are random. However, for many applications the assumption is in general not fulfilled. For example, in structure from motion the missing data are neither sparse nor randomly located, but rather distributed densely in the off-diagonal blocks. In [29], it is also noted that the convex factorization approach may break down due to violation of the sparsity assumption in structure from motion.

Due to the limitation of the nuclear norm formulation, we only consider comparisons based on the bilinear formulation. More specifically, we consider the L_1-Wiberg method in [18] together with two recent follow-up methods, that is, the Regularized L_1-Augmented Lagrangian Multipliers method (RegL_1-ALM) in [39] and General Wiberg in [30] to be the state-of-the-art for robust factorization with missing data and they will serve as a baseline for our work. In [18], L_1-Wiberg is also compared with three other methods, namely the Alternated Linear Programming (ALP) and Alternated Quadratic Programming (AQP) in [23] and L_2-Wiberg in [35]. However, as L_1-Wiberg gives a fairly large improvement over ALP, AQP and L_2-Wiberg, it is unnecessary here to include those three methods in our comparison. In contrast to all these methods, the our approach does not depend on initialization.

Article Overview. The main contribution of this paper is a new approach that directly tries to estimate the globally optimal solution under the L_1-norm. Contrary to the local methods to which we compare, it does not depend on the initialization. Furthermore, we will show that the same methodology is applicable to the

truncated L_1-norm. In this model, each outlier gets a constant penalty. Another contribution is that in our framework, the affine subspace problem can be handled in a robust manner as well. Setting the translation vector to the mean of the observations, which is customary, is not a good choice in the presence of missing data and outliers.

In Section 2, mathematical problem formulations are given for the cost function under either the L_1-norm or the truncated L_1-norm. In Section 3, various applications are described, from the toy example of line fitting to real vision applications, that is, affine structure from motion and photometric stereo. In Section 4, some results on L_1-projection are given, which constitute an essential part of our framework. In Section 5, the special case when the rank $r = m - 1$ is studied and an optimal algorithm is given. In Section 6, the general case when $r < m$ is discussed and two algorithms are described. In Section 7, the adaption of the algorithms to the cost function under the truncated L_1-norm is briefly described. The experimental results and comparisons are exhibited in Section 8, followed by the conclusion.

2 Problem Formulation

We will be focusing on the L_1-norm, which is defined as

$$\|X\|_1 = \sum_{i,j} |x_{ij}|. \tag{2}$$

Under the L_1-norm, the problem can simply be stated as

$$\min_{U,V} \sum_{i,j} |w_{ij} - \sum_k u_{ik} v_{kj}|. \tag{3}$$

In practice, there may be missing observation entries in the matrix W. This means that the cost function should only be summed over the indices i, j that have measured values in W. We will make no requirement that the full observation matrix is available.

In many applications, one would like that erroneous measurements should have a fixed penalty which can be obtained via truncation of the residual errors. Therefore, we will be considering the following problem as well

$$\min_{U,V} \sum_{i,j} \min(|w_{ij} - \sum_k u_{ik} v_{kj}|, \epsilon), \tag{4}$$

where ϵ is a given truncation value. Hence, the maximum cost for an outlier measurement is ϵ under this model.

To shed further light on the factorization problem, one can view it as the estimation of a low-dimensional subspace. Given data $w_i \in \mathbb{R}^m, i = 1 \ldots, n$, the problem in (1) can be treated as that of finding an optimal subspace

$$S = \{w \in \mathbb{R}^m \mid w = Uv, v \in \mathbb{R}^r\} \tag{5}$$

defined by a matrix $U \in \mathbb{R}^{m \times r}$ such that when all the data is projected onto S, the sum of projection errors $\sum_i \|w_i - Uv_i\|_1$ is minimized. Note that with this formulation we always get a linear subspace containing the origin. In many applications though, one is interested in finding an affine subspace

$$S = \{w \in \mathbb{R}^m \mid w = Uv + t, v \in \mathbb{R}^r\}, \tag{6}$$

defined by $U \in \mathbb{R}^{m \times r}$ and $t \in \mathbb{R}^m$. For observations with no missing entries or outliers, the translational component t is optimally estimated as the mean of the observation vectors under the L_2-norm. This is clearly not a good estimator under the L_1-norm or in the presence of outliers. In analogy to (3), the affine subspace problem can be formulated as

$$\min_{U,V,t} \sum_{i,j} |w_{ij} - t_i - \sum_k u_{ik} v_{kj}|, \tag{7}$$

or in matrix notation, cf. (1),

$$\min_{U,V,t} \left\| W - \begin{bmatrix} U & t \end{bmatrix} \begin{bmatrix} V \\ 1 \ldots 1 \end{bmatrix} \right\|_1. \tag{8}$$

where $W \in \mathbb{R}^{m \times n}$, $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{r \times n}$ and $t \in \mathbb{R}^m$

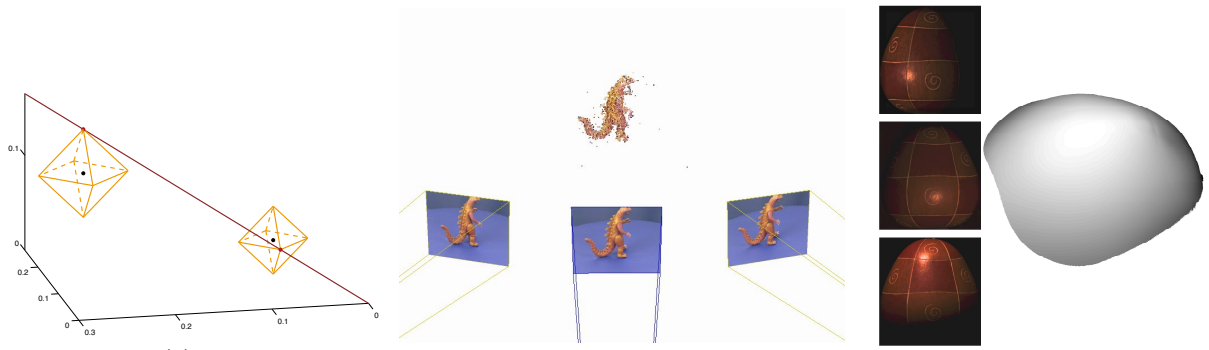
The residual matrix $R \in \mathbb{R}^{m \times n}$, which will be used later, is defined here as

$$R = \left| W - \begin{bmatrix} U & t \end{bmatrix} \begin{bmatrix} V \\ 1 \ldots 1 \end{bmatrix} \right|. \tag{9}$$

In summary, we are considering two different cost functions for the factorization problem, one based on the L_1-norm (3) and one based on the truncated L_1-norm (4), as well as two different versions, one viewed as a subspace estimation problem (5) and one viewed as an affine subspace problem (6). In the next section, we will give several example applications.

3 Applications

Line fitting. Let $w_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ be observed points in the plane ($m = 2$) or in space ($m = 3$). Then, a line through the origin can be parametrized by a direction vector $u \in \mathbb{R}^m$. For each point there should be a parameter $v_i \in \mathbb{R}$ satisfying $w_i \approx uv_i$. In order to



(a) (b) (c)

Fig. 2 (a) Line fitting. L_1-balls are plotted for two 3D points. Note that it is typical that the optimal line goes through the corner of the L_1-ball for exactly two points (which means there is residual error in one axial direction only), whereas the other points (not shown) have errors in two axial directions. (b) Affine structure from motion. Three images of a toy dinosaur and the corresponding 3D reconstruction. (c) Photometric stereo. Three of eight images of a gourd and the corresponding 3D reconstruction viewed from the side. Images are courtesy of [2].

estimate the line parameters, one can solve the factorization problem

$$\min_{u,v} \left\| \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} - u \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \right\|_1 . \tag{10}$$

For a line not necessarily going through the origin, it can be regarded as an affine subspace problem

$$\min_{u,v,t} \left\| \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} - \begin{bmatrix} u & t \end{bmatrix} \begin{bmatrix} v_1 & \dots & v_n \\ 1 & \dots & 1 \end{bmatrix} \right\|_1 . \tag{11}$$

See also Fig. 2(a).

Affine structure from motion. According to the affine camera model [20], a 3D point $v \in \mathbb{R}^3$ is mapped to the image point $w \in \mathbb{R}^2$ by $w = Uv+t$, where $U \in \mathbb{R}^{2\times 3}$ and $t \in \mathbb{R}^2$ encode the orientation and the translation of the camera, respectively. Given image points w_{ij} in image i of 3D point v_j, for $i = 1, \dots, m$ and $j = 1, \dots, n$, this can be written as

$$\begin{bmatrix} w_{11} & \dots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \dots & w_{mn} \end{bmatrix} = \begin{bmatrix} U_1 & t_1 \\ \vdots & \vdots \\ U_m & t_m \end{bmatrix} \begin{bmatrix} v_1 & \dots & v_n \\ 1 & \dots & 1 \end{bmatrix} .$$

This is the basis for the famous Tomasi-Kanade factorization algorithm [33] which first estimates the translation t_i by computing the mean of the observations in the corresponding rows, and then applies SVD to the (reduced) observation matrix in order to recover U and V. We will treat it as an affine subspace problem. See Fig. 2(b) for an example.

Photometric Stereo. Assuming an orthographic camera viewing a Lambertian surface illuminated by a distant light source $v \in \mathbb{R}^3$, the image intensity w of a surface element is given by

$$w = u^T v,$$

where $u \in \mathbb{R}^3$ is the (unnormalized) surface normal. The length $\|u\|$ gives the albedo of the surface element. By varying the light source directions (and keeping the camera fixed), and by considering several image intensities, we end up in the factorization problem (1). The measurement matrix $W \in \mathbb{R}^{m\times n}$ contains the intensities of m pixels in n images, $U \in \mathbb{R}^{m\times 3}$ the albedos and the surface normals of the m pixels and $V \in \mathbb{R}^{3\times n}$ the n light sources. Given the normals, it is possible to estimate a depth map by integration [38]. In Fig. 2(c), some example images are given together with a 3D reconstruction based on the truncated L_1-minimization (see experimental section). Note that in this example the surface is highly specular and do not concur with the Lambertian model at the specularities.

4 L_1-Projections

In this section, we will give some general results concerning L_1-projections.

Theorem 1. *For a given point $w \in \mathbb{R}^m$ and a given r-dimensional affine subspace S defined by a matrix $U \in \mathbb{R}^{m\times r}$ and $t \in \mathbb{R}^m$, the L_1-projection of w onto S occurs only along $m - r$ directions.*

This is equivalently saying that the remaining r directions are error free, that is, r components of the residual vector $w - Uv$ are always zero. The above result is well-known [26,4] and can be formally proved using linear

programming theory. However, it should intuitively be clear that the theorem is true. Writing the cost function explicitly,

$$\min_v \sum_{i=1}^{m} |w_i - t_i - \sum_{k=1}^{r} u_{ik} v_k|, \tag{12}$$

we see that it is a piecewise linear function of the v_k's. Furthermore, as the column vectors u_k, $k = 1, \ldots, r$, form a basis for an r-dimensional subspace they are all linearly independent. Hence the cost tends to infinity as $u \to \infty$ and the minimum must be attained at a corner point, that is, where the derivative is not defined in any direction. So, at least r elements are zero in the residual vector at optimum.

Assume, for a while, that we know the positions of the r zeros of Theorem 1. Since each zero gives a linear constraint on v we could easily compute the L_1-projection from this information. And even if the zero positions are unknown, this technique can be useful if an exhaustive search over the possible positions is performed[1]. A natural question is whether a similar approach can be used to solve the full problem.

5 Hyperplane Fitting

If the dimension of the subspace $r = m - 1$ then we are dealing with a hyperplane. According to Theorem 1, the projection of a given point $w \in \mathbb{R}^m$ onto a hyperplane occurs along a single direction. Moreover, this direction depends only on the hyperplane - not on the point w. This result is a direct consequence of Theorem 2.1 in [26], but for clarity, we state it as a theorem.

Theorem 2. *Given a set of points $w_k \in \mathbb{R}^m$ and an $(m-1)$-dimensional affine subspace S, there exist optimal L_1-projections of w_k onto S such that all occur along a single axis.*

If we know this axis, then we can solve for the hyperplane using linear programming (LP). Hence optimal hyperplane fitting can be solved as a series of m LP problems. Another option is indicated by the following theorem.

Theorem 3. *For an optimal affine hyperplane, there will be $m-1$ rows of zeros and one row with m zero elements in the residual matrix R in (9). Provided we know the positions of these zeros, the hyperplane can be solved for in closed-form.*

[1] This is not the most efficient way of computing an L_1-projection.

Proof. According to Theorem 2, all the points will be projected along a single direction. This means that the residual matrix R will have $m-1$ rows of zeros. Without loss of generality, we assume the top $m-1$ rows of R are zeros, which gives a partition of $R = [\mathbf{0}, \hat{r}^T]^T$ where $\hat{r}$ is a row vector. Applying the same partition to W, U and t leads to

$$\begin{bmatrix} \mathbf{0} \\ \hat{r} \end{bmatrix} = \begin{bmatrix} \tilde{W} \\ \hat{w} \end{bmatrix} - \begin{bmatrix} \tilde{U} & \tilde{t} \\ \hat{u} & \hat{t} \end{bmatrix} \begin{bmatrix} V \\ 1 \ldots 1 \end{bmatrix}. \tag{13}$$

Note the partition of $R \in \mathbb{R}^{m \times n}$ yields a zero matrix $\mathbf{0} \in \mathbb{R}^{(m-1) \times n}$ and a row vector $\hat{r} \in \mathbb{R}^n$.

There exists a coordinate ambiguity in the factorization as we can always reparametrize $[U, t]$ using a matrix $Q = \begin{bmatrix} \tilde{Q} & \tilde{q} \\ \mathbf{0} & 1 \end{bmatrix}$ since we have

$$\begin{bmatrix} U & t \end{bmatrix} \begin{bmatrix} V \\ 1 \ldots 1 \end{bmatrix} = \begin{bmatrix} U & t \end{bmatrix} Q Q^{-1} \begin{bmatrix} V \\ 1 \ldots 1 \end{bmatrix}. \tag{14}$$

This means we can always reparametrize (13) such that $\tilde{U} = I$ and $\tilde{t} = \mathbf{0}$. The reparametrization gives the solution $V = \tilde{W}$,

$$\begin{bmatrix} \mathbf{0} \\ \hat{r} \end{bmatrix} = \begin{bmatrix} \tilde{W} \\ \hat{w} \end{bmatrix} - \begin{bmatrix} I & \mathbf{0} \\ \bar{u} & \bar{t} \end{bmatrix} \begin{bmatrix} \tilde{W} \\ 1 \ldots 1 \end{bmatrix}. \tag{15}$$

The remaining cost $\|\hat{r}\|_1$ is now a function of $\bar{u}$ and $\bar{t}$ which is piecewise linear. If the columns of $\tilde{W}$ span $\mathbb{R}^m$ then the cost tends to infinity as $\| \begin{bmatrix} \bar{u} & \bar{t} \end{bmatrix} \|_1 \to \infty$. Hence the piecewise linear cost $\|\hat{r}\|_1$ attains its minimum at a corner point with m zeros in the residual vector and if we know the zero positions, then we can estimate the unknowns in $\bar{u}$ and $\bar{t}$ by solving linear equations.

If on the other hand the columns of $\tilde{W}$ do not span $\mathbb{R}^m$, then the complete data matrix W has to lie in a subspace of R^m. So, for the optimal hyperplane, all residuals are zero. Using m of these we can compute an optimal hyperplane.

□

As an example, consider the case of line fitting to a set of points $\{w_i, i = 1, \ldots, n\}$ in $\mathbb{R}^2$. For an optimal line, the residual matrix $R \in \mathbb{R}^{2 \times n}$ has a full row of zeros in either x or y coordinates.

Note that the final estimation of $\bar{u}$ and $\bar{t}$ could be solved more efficiently using LP, see Algorithm 1. However, this does not generalize to truncated L_1-norm. In that case, one needs to do an exhaustive search based on Theorem 3 or a random search as will be described later.

Algorithm 1 Optimal hyperplane fitting (HF)

Given an observation matrix W, solve for the optimal affine subspace (U^*, t^*) and the projection matrix V^*

1. Initialize the best error $\epsilon^* = \infty$
2. For $i = 1$ to m
3. Set the index set $\mathcal{P}$ for row partition as
4. $\mathcal{P} = \{1, 2, \ldots, m\} \backslash \{i\}$
5. Let $\tilde{W} = W_{\mathcal{P}}$ and $\hat{w} = W_{\{i\}}$ in (15)
6. Solve $\min_{\bar{u}, \bar{t}} \|\hat{r}\|_1$ in (15) using LP
7. Calculate the L_1-error ϵ
8. If $\epsilon < \epsilon^*$
9. $U^* = U, t^* = t, V^* = V$ and $\epsilon^* = \epsilon$
10. return $U^*, t^*, V^*, \epsilon^*$

6 The General Case

A subspace defined by $U \in \mathbb{R}^{m \times r}$ has $d = (m - r)r$ degrees of freedom (mr parameters defined up to an $r \times r$ coordinate transformation). Similarly, an affine subspace has $d = (m - r)(r + 1)$ degrees of freedom. For example, when $r = m - 1$ as in the previous section, there are only m degrees of freedom of the affine subspace, and these m unknowns can be determined in closed-form from the m extra zeros in the residual matrix (Theorem 3).

In the general case ($r < m - 1$), there may be fewer zeros in the residual matrix than necessary to solve directly for the subspace. Moreover, even with sufficiently many zeros, the structure of the residual matrix might not allow us to linearly solve for the parameters. Despite these facts, similar ideas can be used to achieve state-of-the-art results and very often to find an optimal L_1-factorization. The basis will be the following type of points:

– A point (U, t) representing an affine subspace in parameter space is a *principal stationary point* if the residual matrix has d extra zeros for the optimal V.

By *extra* here is of course meant the additional zeros to the r zeros present in every column of the residual matrix according to Theorem 1. Note that when $r = m - 1$, then there are always $d = m$ extra zeros and hence all optimal subspaces U^* to the L_1-factorization problem are principal stationary points (Theorem 3).

Empirically, we have made the following two observations concerning L_1-optimal factorizations:

- In practice, the optimal subspace for L_1-factorization is often a principal stationary point.
- Even if the optimal subspace is not a principal stationary point, there is often a principal stationary point which is close to the optimal one.

How Common Are Principal Stationary Points? To give some insight into this question we considered a low-dimensional problem in order for brute-force search to be applicable. More precisely, we considered fitting of a r-dimensional subspace in $\mathbb{R}^m$.

	m=3	m=4	m=5
r=1	98.4%	96.6%	96.0%
r=2	-	92.0%	94.0%
r=3	-	-	95.0%

Table 1 Percentage of principal stationary points being optimal.

To generate the data, we first randomly generate r orthonormal basis u_i for $i = 1, 2, \cdots, r$ in $\mathbb{R}^m$, which constitutes the columns of ground truth subspace U. The data is generated on the subspace using a linear combination of the basis, $x_j = \sum_{i=1}^{j} a_i u_i$ where the coefficients a_i are uniformly drawn from $[-1, 1]$. Gaussian noise from $\mathcal{N}(0, 0.02)$ are added to all the points. And 10% data are regarded as outliers by a random perturbation uniformly drawn from $[-1, 1]$.

Grid search is performed in the parameter space of U. We fix the top $r \times r$ block of U to be identity, and search for the remaining $(m - r)r$ variables of U. Since the ground truth of elements of U is generated between $[-1, 1]$. We perform the search in the slightly larger range of $[-2, 2]$ by dividing it into equally-sized intervals, with the length of each interval being 0.1. We estimate V by L_1-projection and refining the best solution using the method from [18]. The number of zeros in the residual matrix of the solution was counted to evaluate whether it was a principal stationary point or not.

Due to the exponentially increasing cost for the grid search with the number of variables in U, we only consider the test on low-dimensional problems. More specifically, the problems we tested are the cases with $r = 1$ for $m = 3, 4, 5$, $r = 2$ for $m = 4, 5$, and $r = 3$ for $m = 5$. We skipped the hyperplane-fitting case ($r = m - 1$) here. We run 2000 random problems for $r = 1$ cases and 200 random problems for $r = 2, 3$ cases due to the exponentially increasing time complexity. The percentage of the principal stationary point being the global optimal is summarized in Tab. 1

From Tab. 1, we observe that for more than 90% of random problems we tested on estimating the different low-dimensional subspaces, the principal stationary points are the globally optimal solutions. This observation motivates the following algorithms that consider only principal stationary points. Note the cost function for one example of the problem $m = 3$, $r = 1$ is plotted in Fig. 1.

6.1 Searching for Principal Points

Our approach to general subspace fitting is based on searching for principal stationary points, that is, points that have d extra zeros in the residual matrix, allowing us to solve directly for the parameters. This suggests that to estimate the subspace U we only need to consider a subset of columns with d extra zeros in the residual matrix. Once the subspace U is estimated, the projection coefficient V for the remaining columns of W can be solved either by a linear program or with the approach discussed in Section 4.

We first focus on estimating a subspace U. A zero at position (i, j) in the residual matrix gives a bilinear equation in the unknowns t_i, u_{ik} and v_{kj}

$$w_{ij} - t_i - \sum_{k=1}^{r} u_{ik} v_{kj} = 0. \tag{16}$$

By Theorem 1, every column yields at least r such equations, but looking for principal stationary points we can assume that there are d extra zeros corresponding to the d degrees of freedom of the subspace. Hence we consider a subset of at most d columns and assume that there are $dr + d$ zeros in the corresponding residual matrix. For small problems an exhaustive search over the possible positions of these zeros might be tractable, but to handle larger problems, a randomized algorithm is necessary.

In principle, this approach can be viewed as applying RANSAC [19] to low-rank matrix factorization, although our motivation was quite different. Just as in RANSAC a minimal set of data points are assumed to have zero error and this assumption is used to find the model parameters, and then, the obtained parameters are evaluated on all data to measure the goodness of fit. Either we repeat this exhaustively for every possible minimal subset or for a fixed number of random subsets. More on this later.

6.2 Exhaustive Search

For small-sized problems it is tractable to search the space of all possible positions for the $d + dr$ zeros of a principal stationary point. For each possible zero pattern, we need to solve a set of $d+dr$ degree-2 polynomial equations, which can be expensive and might yield up to 2^{d+dr} solutions. Fortunately, the structure of these polynomial equations, for example, all the quadratic terms are bilinear, yields much fewer solutions. In fact, for low-dimensional problems, such as line fitting in $\mathbb{R}^3$, there is a unique and simple closed form solution, rendering a much more efficient algorithm.

Algorithm 2 Exhaustive Search (ES)

Given an observation matrix W, solve for the optimal affine subspace (U^*, t^*) and the projection matrix V^*.

1. Initialize the best error $\epsilon^* = \infty$
2. Generate all the column subsets $\{\mathcal{I}_i\}$ of size d
3. Generate all the residual patterns $\{R_j\}$ of size $\mathbb{R}^{m \times d}$
4. For each column subset $\mathcal{I}_i$
5. For each residual pattern R_j.
6. Compute U, t and $V_{\mathcal{I}_i}$ in closed form using $W_{\mathcal{I}_i}$, R_j
7. Compute the projection $V_{\{1,2,\ldots,n\}\setminus\mathcal{I}_i}$
8. Compute the L_1-error ϵ
9. If $\epsilon < \epsilon^*$
10. $U^* = U, t^* = t, V^* = V$ and $\epsilon^* = \epsilon$
11. return $U^*, t^*, V^*, \epsilon^*$

Note that when generating the residual patterns, one should first make sure each column has at least r zeros, which follows from Theorem 1. Then d extra zeros should be arranged such that each row has at least r zeros, which followed by applying Theorem 1 to the transpose of the measurement matrix W. This makes sure that each row of U can be determined from the equation system.

6.3 Random Search

The exhaustive search algorithm quickly becomes infeasible with growing problem size, both because the number of possible zero patterns grows exponentially and because solving the system of quadratic equations gets increasingly more expensive, as for general d and r, there is no simple closed-form solution.

Simple patterns of zeros. To work around these problems, we propose a random search algorithm only considering especially simple residual patterns. More precisely, we restrict the search to patterns that lead to linear equations and can hence be solved extremely fast. Note that even these simple patterns abound and in practice, we can always find such patterns despite missing entries as long as the factorization problem is well-posed.

Here we describe a family of general residual pattern that can be solved linearly. The first assumption is that (possibly after some permutations on rows and columns) we have a $(r + 1) \times r$ zeros in the top left of the residual matrix. Based on the degree of freedom in the subspace U, a coordinate system can always be chosen such that the first r rows of U become an identity

matrix, for example the top r rows, that is,

$$U = \begin{bmatrix} I_{r\times r} \\ \mathbf{u}_{r+1}^T \\ \mathbf{u}_{r+2}^T \\ ... \\ \mathbf{u}_m^T \end{bmatrix}. \tag{17}$$

Now, from the definition of R in (9), we can immediately obtain the first r columns of V

$$\tilde{W} - I[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r] = \mathbf{0}, \tag{18}$$

where $\tilde{W}$ is the top left $r \times r$ sub-matrix of W. Then the last row of the $(r+1)\times r$ zeros in R gives us r linear constraint on $\mathbf{u}_{r+1}^T$

$$\mathbf{w}_{r+1}^T - \mathbf{u}_{r+1}^T[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r] = \mathbf{0}, \tag{19}$$

where $\mathbf{w}_{r+1}^T$ are the elements of W corresponding to the last row of the zero block in R. As $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_r}$ are already known from (18), we can compute $\mathbf{u}_{r+1}^T$ linearly from (19). (We have r linear equations and r unknowns in u_{r+1}^T.)

To solve for another row of U, we need at least r zeros in that row of R. They can be either in new columns or in columns already used to compute $\mathbf{u}_{r+1}^T$. Let us assume that after permutations, the zeros are in columns $r+1$ to $2r$.[2] This yields

$$\mathbf{w}_{r+2}^T - \mathbf{u}_{r+2}^T[\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_{2r}] = \mathbf{0}. \tag{20}$$

However, since the $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_{2r}$ are unknown, we first need to compute them. We can use the fact that the $\mathbf{u}_1^T$, $\mathbf{u}_2^T$, ... $\mathbf{u}_{r+1}^T$ are all known. This means for the column $\mathbf{v}_i \in \mathbb{R}^r$ of V to be solvable, we need at least r zeros in the top $r+1$ rows of i^{th} column of R. Take solving $\mathbf{v}_{r+1}$ for example, assume $n_1, n_2, ..., n_r$ are the indices of rows, where those r zeros locates in the $(r+1)^{th}$ column of R. Taking the corresponding elements of W, that is, the rows $n_1, n_2, ..., n_r$ of the column $r+1$, we form a vector $\tilde{\mathbf{w}}$. Taking the corresponding rows of U, that is, the rows $n_1, n_2, ..., n_r$ of U, we form a sub-matrix $\tilde{U}$ of size $r \times r$. Then $\mathbf{v}_{r+1}$ is computed from

$$\tilde{\mathbf{w}} - \tilde{U}\mathbf{v}_{r+1} = \mathbf{0}. \tag{21}$$

Note this is similar to (18) except we have to solve the column of V separately as the zero position for each column might be varied now. Other columns, $\mathbf{v}_{r+2}, \ldots, \mathbf{v}_{2r}$ are solved in the same way. With $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_{2r}$ solved, we can compute $\mathbf{u}_{r+2}^T$ using (20).

So to solve $\mathbf{u}_{r+2}^T$, we required a block of size $(r+2)\times r$, inside which the top $r+1$ rows contain at least r zeros in each column, and the last row contain only zeros. Note that apart from this the position of the non-zero residuals in this block is arbitrary. One example for $r = 3$ is

$$R = \begin{bmatrix} 0 & 0 & 0 & \cdot & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & \cdot & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & \cdot & ... \\ \cdot & \cdot & \cdot & 0 & 0 & 0 & ... \\ ..... & & & & & & ... \end{bmatrix}. \tag{22}$$

The remaining rows $\mathbf{u}^T$ of U can be solved sequentially in the same way. It is worth noting that in general, this will not compute all columns in V but only up to d of them. We will use $\mathcal{J}$ to denote the columns which are computed directly in this way. The remaining columns of V can be found using L_1-projections since U is already known.

This also leads to a simple strategy to handle missing data. When a random residual pattern is generated in Algorithm 3, we simply restrict it such that the zeros in R cannot be placed in a position corresponding to a missing element of W. Fig. 3 shows an example residual pattern generated randomly for the structure from motion application. The zero positions are close to the main diagonal since that is where we have observed data.

Fig. 3 A random generated residual pattern for $m = 20$ and $r = 3$ case. Each black patch is a zero in R.

Adaptive Sampling. To generate the residual pattern described above, we sample $d + dr$ elements from observation W, corresponding to the zeros in R. As noted earlier, this is basically the RANSAC approach to matrix factorization although motivated in a completely different way. Consequently, we can use any of the alternative sampling strategies that have been proposed for RANSAC. For example, guided-MLESAC [34] proposes an algorithm for maximum likelihood estimation by RANSAC, which estimates the inlier probability of each match based on the proximity of matched features. PROSAC [12] measures the similarity of correspondences, and uses sequential thresholding to form a sequence of progressively larger set of top-ranked correspondences. It is based on the mild assumption that correspondences with high similarity are more likely to

[2] If one or more are in the first r columns it only makes things easier as v_1 to v_r are already known.

be inliers. We chose the following PROSAC-like sampling strategy.

We initialize the probability of sampling w_{ij} of W to be $p_{ij} > 0$ if w_{ij} is observable, otherwise, we set $p_{ij} = 0$. In each iteration when a better solution is found, we check the residual $r_{ij} = w_{ij} - \mathbf{u}_i^T \mathbf{v}_j$, if r_{ij} is large, then we lower the probability p_{ij} of picking up w_{ij} in the next iteration, otherwise we increase p_{ij}. In practice, the strategy helps us to find a better solution using fewer sampling steps.

Algorithm 3 Random Search (RS)

Given an observation matrix W and a max iterations N, solve for the optimal affine subspace (U^*, t^*) and the projection matrix V^*.

1. Initialize the best error $\epsilon^* = \infty$
2. Initialize the probabily $p_{ij} = 1$ for $i = 1, \ldots, m, j = 1, \ldots, n$
3. While $k \leq N$
4. Randomly generate a simple residual pattern s.t. element (i, j) is included with probability $\approx p_{ij}$
5. Compute U, t and $V_{\mathcal{J}}$ linearly as described in text
6. Compute $V_{\{1,2,\ldots,n\}\setminus\mathcal{J}}$ using L_1-projection
7. Compute the L_1-error ϵ^i
8. If $\epsilon^k < \epsilon^{k-1}$
9. Update the probability p_{ij} based on $|W - UV|$.
10. If $\epsilon^k < \epsilon^*$
11. $U^* = U, t^* = t, V^* = V$ and $\epsilon^* = \epsilon$
12. return $U^*, t^*, V^*, \epsilon^*$

7 Truncated L_1-Factorization

Perhaps somewhat surprisingly, most of the results we have presented generalize easily to the truncated L_1-norm; cf. (4). A brief sketch of the proof is as follows. Consider an optimal factorization with respect to the truncated L_1-norm. Now divide the measurements into inliers — having a residual smaller than ϵ — and outliers — having a residual larger than ϵ. Now apply Theorems 2 and 3 to the inliers only.

Algorithms 2 and 3 (but not Algorithm 1) can be used to optimize the truncated L_1-norm. The only required modification is to evaluate solutions using the *truncated* L_1-norm rather than the standard L_1-norm.

8 Experiments

All experiments are run on a Macbook Pro with a quadcore CPU and 8GB RAM.

Line fitting. We first test our exhaustive search method for affine line fitting in $\mathbb{R}^3$. The purpose of this experiment is to investigate the local minima problems. More quantitative results are given in the following experiments. In this case, all the principal stationary points can be solved for in closed form. For each experiment, 20 3D points with coordinates in $[-1, 1]$ are generated on a line and perturbed with Gaussian noise with standard deviation 0.1. In addition, we perturb 80% of the points with uniform noise in $[-1, 1]$ to be outliers. 100 random examples are tested using both our exhaustive search algorithm with L_1-norm and L_1-Wiberg from [18].

From Fig. 4, we can see that our method performs better as a fairly large portion of errors (brown) fall into the interval between 0 and 0.1 while most errors for L_1-Wiberg (grey) lie between 0.1 and 0.2. In every single instance, our algorithm performs better (around 50% of the cases) or equally well compared to the L_1-Wiberg algorithm. This means that the L_1-Wiberg gets stuck in local optima roughly half of the instances. We have made similar observations for other settings by varying the inlier/outlier ratio and the dimensions, both for affine and non-affine cases. Running time is 2s for our methods and 0.03s for L_1-Wiberg.

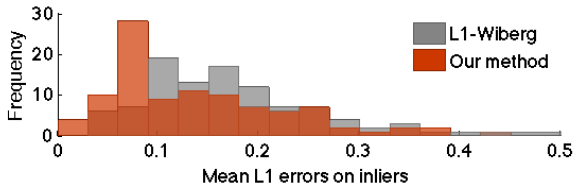


Fig. 4 Results on the affine line fitting in $\mathbb{R}^3$.

Affine Structure from Motion. We also tried N-view structure from motion using the Oxford dinosaur sequence. For each instance we use 300 points in N consecutive views, where N varies from 2 to 10. This creates a data matrix of size $2N \times 300$ with up to 75% missing data. Outliers with uniform noise on $[-50, 50]$ are added to 10% of the tracked points.

The 2-view SfM is exactly a hyperplane fitting problem, so we use Algorithm 1. For problems with more than two views, we use Algorithm 3 with 10^6 iterations. Running times are up to 3.8mins using our parallel OpenMP implementation in C. As a comparison, we use the C++ implementation of L_1-Wiberg in [18], General Wiberg in [30] and Matlab code of the Regularized L_1-Augmented Lagrange Multiplier method (RegL_1-ALM) in [39]. In our experiments, L_1-Wiberg converges in no more than 50 iterations, which takes up to 30 s, while General Wiberg exhibits slower convergence. We set the maximum number of iterations to 500, which results in run-times up to 10.4mins. In a few cases, it does

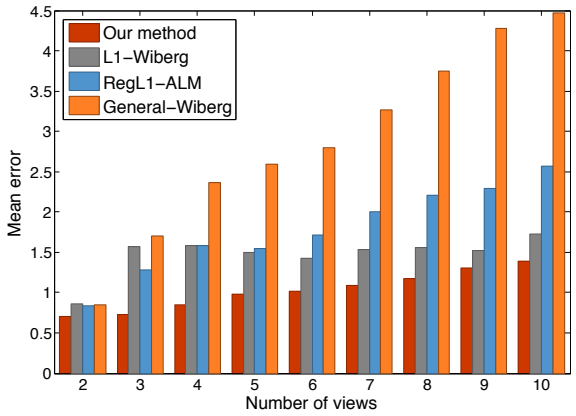


Fig. 5 Results on Affine Structure from Motion with varying number of views. The L_1-Wiberg, General Wiberg and RegL_1-ALM run with truncated-SVD or all zeros initialization. Errors are given in pixels.

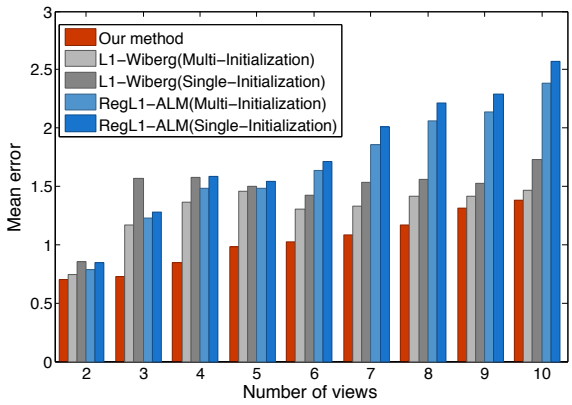


Fig. 6 Results on Affine Structure from Motion with varying number of views. L_1-Wiberg and RegL_1-ALM are runned with single or multiple random initializations. Errors are given in pixels.

not even converge. The RegL_1-ALM converges in 0.3 s. L_1-Wiberg and General Wiberg are initialized using the truncated SVD, while the RegL_1-ALM is initialized with all zeros in U and V. All follows the settings in the original papers.

For each $N = 2, 3, \ldots, 10$, all possible instances with N consecutive views were tested. The Mean Absolute Error (MAE) of inliers for each N is shown in Fig. 5. The MAE of inliers is defined as

$$\text{MAE} = \frac{1}{|\mathcal{I}|} \sum_{(i,j)\in\mathcal{I}} |w_{ij} - \sum_{k=1}^{r} u_{ik} v_{kj}| \tag{23}$$

where $\mathcal{I}$ is the set of inliers, and $|\mathcal{I}|$ is the cardinality of the set. We can see that our method clearly achieves lower error in all the experiments. As the dimension goes up, the percentage of missing data also rises, which heavily affects the performance for General Wiberg, but also for RegL_1-ALM.

To compare the methods with the same running time, we run the L_1-Wiberg and RegL_1-ALM with multiple random initializations. The General Wiberg is excluded in this comparison due to its weak performance both in accuracy and complexity. To generate the random initialization, we first scale the data matrix W so that all the elements are around $[-1, 1]$. Then the elements of U and V are sampled uniformly from the interval $[-1, 1]$. We run the L_1-Wiberg with 10 different random initializations (including truncated SVD) and RegL_1-ALM with 1000 random initializations (including setting U and V to zeros). This leads to roughly the same running time for the three methods. The comparison is given in the Fig. 6. It turns out that trying multiple random initializations leads to some improvement for both L_1-Wiberg and RegL_1-ALM. But the improvement is minor, especially for the RegL_1-ALM method, considering it runs with more random starts. It also verifies, as claimed in [39], that it is hard to find a better solution using random initializations compared with the solution by setting U and V to be all zeros. One possible reason is that the algorithm first solves U with respect to V. In vision application, the number of elements in V is usually very large, leading to a huge search space of V. So 1000 random initializations on V might be relatively very few, from which it is hard to find a better solution. As seen from Fig. 6, the other two methods do not gain much by trying different starting point given the same amount of running time. Our method still achieves lower errors.

To further examine the quality of our solutions, we run the random search algorithm 10 times, each with 10000 iterations, to check if the same optimal is obtained. Taking an example of 3-view SfM, we plot the result in Fig. 7. We found that different random searches, although not leading to exactly the same solution, give very similar low errors. In this case, our random search has achieved lower error than our competitors in no more than 100 iterations.

We also tested on the 3-view data with different outlier ratios. The data setup is the same as above, but we add varied percentage of outliers to the data from 10% up to 50%. For our method, we use the random search with truncated L_1-norm. All the methods are affected by the increasing outliers, see Fig. 8. The chance of our method to sample an outlier-free minimal set is decreased with increasing outlier ratio. Still we achieve smaller median errors of inliers.

Photometric Stereo. In [2], a set of 102 images (size 588×552 pixels) of a gourd was used to estimate the

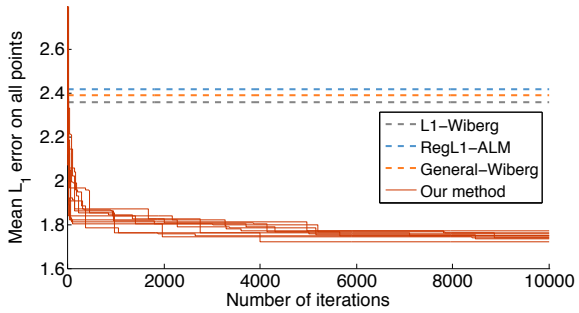


Fig. 7 Quality of our solutions. The random search is run for 10 times with each red solid curve representing the current best error vs. the number of iterations. The grey, blue and orange dotted lines are the results for the other three methods.

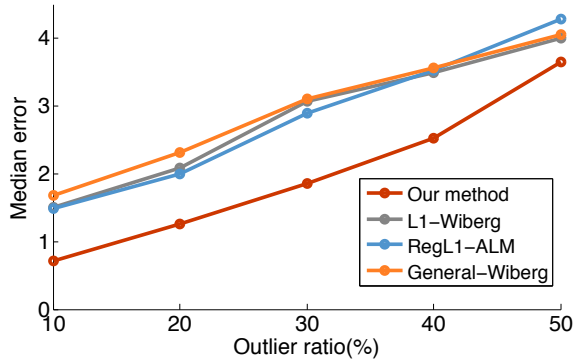


Fig. 8 Results on 3-view Affine Structure from Motion with varying outlier ratios. Errors are given in pixels.

3D shape and the surface reflectance properties using a sophisticated data-driven photometric stereo model. In contrast, we use a standard Lambertian model and a subset of eight random images to demonstrate that it is still possible to obtain a good estimate of 3D surface shape by using the truncated L_1-norm; see Fig. 2(c). Deviations from the Lambertian model such as specularities are handled by the robust choice of norm.

As the number of pixels is usually very large in the experiment, that is, of order 10^6, the computation of the L_1-projection, that is, estimate V given U for all the points in each iteration in Algorithm 3 are quite time consuming. Here we adopt a strategy that in each iteration, we compute the surface normal v for only a subset of points. In our experiment, we define this subset as the set of points on the down-sampled image with the down-sampled factor 4. It turned out the error on this subset of points is a good approximation of the error of all the points. And it gives a large speed up in the algorithm. Note that when estimating U we randomly sample the data from the original image. And the final solution is still for the image of the original size.

As a baseline, we compare with the RegL_1-ALM in [39]. We also tried the L_1-Wiberg of [18] and General Wiberg of [30], but none of those was possible to run on such a large problem. For our method, the truncation threshold is set to $\epsilon = 0.05$. It should be noted that each iteration of our method takes 0.3 s, and we use 15000 iterations. The running time for RegL_1-ALM is just 35 s. To make a fair comparison with respect to the running time, we also use the multiple random initializations for RegL_1-ALM and pick out the best solution. Here we run it with 1000 different random starting points, which gives roughly the same running time.

The result of our 3D shape estimate is given in Fig. 2(c) and the detected specularities are shown in Fig. 9. Visual inspection shows that our solution basically captures the correct saturated points. The RegL_1-ALM has very similar visual results so we omit them here. If we calculate the average truncation error per pixel, our method achieves a mean absolute error of 0.0049 while the mean absolute error of RegL_1-ALM is 0.0052.

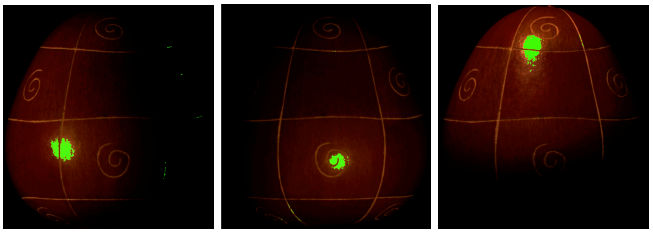


Fig. 9 Results on photometric stereo. Three of eight input images where points with absolute residuals above $\epsilon = 0.05$ are marked in green.

9 Conclusions

We have presented an alternative way of solving the factorization problem under the L_1-norm which also applies to the truncated L_1-norm. The method is independent of initialization, trivially parallelizable and as our empirical investigation of low-dimensional problems show, often the optimal solution is obtained. Compared to iterative methods based on local optimization, the quality of our solution is significantly better in terms of lower error. Our experimental results demonstrated that the local minima problem is not satisfactorily solved by the iterative methods.

The method we propose in the paper resembles the standard RANSAC algorithm in [19] in many ways. Therefore, many of the developments for RANSAC [13,

14] could also be applied to our framework as well. For example, it may be worthwhile to develop a guided search strategy for finding principal stationary points. This is left for future work.

References

1. H. Aanaes, R. Fisker, K. Åström, and J. Carstensen. Robust factorization. *Trans. Pattern Analysis and Machine Intelligence*, 2002.
2. N. Alldrin, T. Zickler, and D. Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *Conf. Computer Vision and Pattern Recognition*, 2008.
3. C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Conf. Computer Vision and Pattern Recognition*, 2000.
4. J. Brooks and J. Dulá. The L1-norm best-fit hyperplane problem. *Applied Mathematics Letters*, 2013.
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.
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.
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.
8. E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of ACM*, 2009.
9. E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.
10. M. K. Chandraker and D. J. Kriegman. Globally optimal bilinear programming for computer vision applications. In *Conf. Computer Vision and Pattern Recognition*, 2008.
11. B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan. Multi-task low-rank affinity pursuit for image segmentation. In *Int. Conf. Computer Vision*, pages 2439–2446, 2011.
12. O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *Conf. Computer Vision and Pattern Recognition*, pages 220–226, 2005.
13. O. Chum and J. Matas. Optimal randomized ransac. *Trans. Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008.
14. O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Pattern Recognition*, pages 236–243. Springer, 2003.
15. T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Trans. Pattern Analysis and Machine Intelligence*, 2001.
16. J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *Int. Journal Computer Vision*, 1998.
17. Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. In *Conf. Computer Vision and Pattern Recognition*, 2012.
18. A. Eriksson and A. Hengel. Efficient computation of robust weighted low-rank matrix approximations using the L_1 norm. *Trans. Pattern Analysis and Machine Intelligence*, 2012.
19. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.
20. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.
21. D. Jacobs. Linear fitting with missing data for structure-from-motion. *Computer Vision and Image Understanding*, 2001.
22. F. Kahl and A. Heyden. Affine structure and motion from points, lines and conics. *Int. Journal Computer Vision*, 33(3):163–180, 1999.
23. 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.
24. F. D. la Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1-3):117–142, 2003.
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.
26. O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 1997.
27. D. Miyazaki and K. Ikeuchi. Photometric stereo under unknown light sources using robust SVD with missing data. In *Int. Conf. Image Processing*, 1999.
28. 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.
29. C. Olsson and M. Oskarsson. A convex approach to low rank matrix approximation with missing data. In *Scandinavian Conf. on Image Analysis*, 2011.
30. D. Strelow. General and nested wiberg minimization. In *Conf. Computer Vision and Pattern Recognition*, 2012.
31. P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conf. Computer Vision*, 1996.
32. J.-P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy. Algorithms for batch matrix factorization with application to structure-from-motion. In *Conf. Computer Vision and Pattern Recognition*, Minneapolis, USA, 2007.
33. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. Journal Computer Vision*, 1992.
34. B. Tordoff and D. W. Murray. Guided sampling and consensus for motion estimation. In *European Conf. Computer Vision*, pages 82–98, Copenhagen, Denmark, 2002.
35. T. Wiberg. Computation of principal components when data are missing. In *Proc. Second Symp. Computational Statistics*, 1976.
36. F. Xiong, O. I. Camps, and M. Sznaier. Dynamic context for tracking behind occlusions. In *European Conf. Computer Vision*, pages 580–593, 2012.
37. 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.
38. A. Yuille and D. Snow. Shape and albedo from multiple images using integrability. In *Conf. Computer Vision and Pattern Recognition*, 1997.
39. 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.

40. Y. Zheng, S. Sugimoto, S. Yan, and M. Okutomi. Generalizing wiberg algorithm for rigid and nonrigid factorizations with missing components and metric constraints. In *Conf. Computer Vision and Pattern Recognition*, pages 2010–2017, 2012.