# Efficient Computation of Robust Weighted Low-Rank Matrix Approximations Using the $L_1$ Norm

Anders Eriksson and Anton van den Hengel

**Abstract**—The calculation of a low-rank approximation to a matrix is fundamental to many algorithms in computer vision and other fields. One of the primary tools used for calculating such low-rank approximations is the Singular Value Decomposition, but this method is not applicable in the case where there are outliers or missing elements in the data. Unfortunately, this is often the case in practice. We present a method for low-rank matrix approximation which is a generalization of the Wiberg algorithm. Our method calculates the rank-constrained factorization, which minimizes the $L_1$ norm and does so in the presence of missing data. This is achieved by exploiting the differentiability of linear programs, and results in an algorithm can be efficiently implemented using existing optimization software. We show the results of experiments on synthetic and real data.

**Index Terms**—Low-rank matrix approximation, $L_1$-minimization.

## 1 Introduction

We are concerned with the problem of identifying two low-rank factors of a matrix in the situation where some of the elements of the matrix are unknown. The problem may be stated in terms of the following optimization problem:

$$\min_{U,V} \|\hat{W} \odot (Y - UV)\|, \tag{1}$$

where $Y \in \mathbb{R}^{m \times n}$ is a matrix containing measurements, and the unknown factor matrices are $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$. We let $\hat{w}_{ij}$ represent an element of the matrix $\hat{W} \in \mathbb{R}^{m \times n}$ such that $\hat{w}_{ij}$ is 1 if $y_{ij}$ is known, and 0 otherwise. In the general statement of the problem, $\|\cdot\|$ can be any matrix norm, but in this work we consider the 1-norm,

$$\|A\|_1 = \sum_{i,j} |a_{ij}|, \tag{2}$$

in particular.

The calculation of a low-rank factorization of a matrix is a fundamental operation in many computer vision applications. It has been used in a wide range of problems, including structure-from-motion [1], polyhedral object modeling from range images [2], layer extraction [3], recognition [4], and shape from varying illumination [5].

In the case where all of the elements of $Y$ are known, the singular-value decomposition may be used to calculate the best approximation as measured by the $L_2$ norm. It is often the case in practice, however, that some of the elements of $Y$ are unknown. It is also common that the noise in the elements of $Y$ is such that the $L_2$ norm is not the most appropriate. In this case, the $L_1$ norm is often used to reduce sensitivity to the presence of outliers in the data. Unfortunately, it turns out that introducing missing data and using the $L_1$ norm makes the problem (1) significantly more difficult to solve. The first problem is that it is a nonsmooth problem, so many of the standard optimization tools available will not be applicable. The second is that it is a nonconvex problem, so certificates of global optimality are in general hard to provide. And finally, the optimization process can also be a very computationally demanding task when applied to real-world applications where the number of unknowns may be very large.

In this paper, we present a method that efficiently computes a low-rank approximation of a matrix in the presence of missing data, which minimizes the $L_1$ norm by effectively addressing the issues of nonsmoothness and the computational requirements. Our proposed method should be viewed as a generalization of one of the more successful algorithms for the $L_2$ case, the Wiberg method [6].

### 1.1 Notation

All of the variables used in this paper are either clearly defined or should otherwise be obvious from the context in which they appear. Additionally, $I_n$ denotes an $n \times n$ identity matrix, $\odot$ and $\otimes$ are the Hadamard and Kronecker products, respectively. Upper case roman letters denote matrices and lower case ones vectors and scalars. We also use the convention that $v = \mathrm{vec}(V)$, a notation that will be used interchangeably throughout the remainder of this paper.

## 2 Previous Work

The subject of matrix approximation has been extensively studied, with much of the focus being on methods exploiting the $L_2$ norm. These methods have had a number

• *The authors are with the Department of Computer Science, The University of Adelaide, North Terrace, SA 5005, Australia. E-mail: {anders.eriksson, anton.vandenhengel}@adelaide.edu.au.*

*Manuscript received 16 Dec. 2010; revised 9 Dec. 2011; accepted 30 Apr. 2012; published online 22 May 2012.*
*Recommended for acceptance by T. Darrell, D. Hogg, and D. Jacobs.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMISI-2010-12-0960.*
*Digital Object Identifier no. 10.1109/TPAMI.2012.116.*

of different names, including principal component analysis, subspace learning, and matrix or subspace factorization. In this paper, we describe the problem in terms of the search for a pair of matrices specifying a low-rank approximation of a measurement matrix, but the approach is equally applicable to any of these equivalent problems.

For an outstanding survey of many existing methods of least $L_2$-norm factorization see the work of [7]. This paper also provides a direct quantitative comparison of a number of key methods, but unfortunately this does not include the Wiberg algorithm [6]. This method, which was first proposed more than 30 years ago, has been largely misunderstood or neglected by computer vision researchers, an issue which was addressed in the excellent work of [8], effectively reintroducing the Wiberg algorithm to the vision community. It was also shown there that on many problems the Wiberg method outperforms many of the more recent methods.

The subject of robust low-rank matrix approximation has not received as much attention within the computer vision literature as it has in other areas (see [9], [10], for example). This is beginning to be addressed, however. A very good starting point toward a study of more robust methods, however, is the work of [11]. One of the first methods suggested within the computer vision literature was Iteratively Reweighted Least Squares, which minimizes a weighted $L_2$ norm [12]. The method is unfortunately very sensitive to initialization (see [13] for more detail).

Black and Jepson [14] describe a method by which it is possible to robustly recover the coefficients of a linear combination of known basis vectors that best reconstructs a particular input measurement. This might be seen as a robust method for the recovery of $V$ given $U$ in our context. De La Torre and Black [11] present a robust approach to Principal Component Analysis which is capable of recovering both the basis vectors and coefficients which is based on the Huber distance.

Croux and Filzmoser [15] suggested the $L_1$ norm as a method for addressing the sensitivity of the $L_2$ norm to outliers in the data. The approach they proposed was based on a weighting scheme which applies only at the level of rows and columns of the measurement matrix. This means that if an element of the measurement matrix is to be identified as an outlier, then its entire row or column must also be so identified.

Ke and Kanade [13] present a factorization method based on the $L_1$ norm which does not suffer from the limitations of the Croux and Filzmoser approach and which is achieved through alternating convex programs. This approach is based on the observation that, under the $L_1$-norm, for a fixed $U$, the problem (1) can be written as a linear problem in $V$, and vice versa. A succession of improved matrix approximations can then be obtained by solving a sequence of such linear programs, alternately fixing $U$ and $V$. It was also shown there that one can solve for the Huber-norm, an approximation to the $L_1$-norm, in a similar fashion, with the difference that each subproblem becomes a quadratic problem. Both of these formulations result in convex subproblems for which efficient solvers exist; however, this does not guarantee that global optimality is obtained for the original problem in the $L_1$-norm.

The recent work of [16] also deals with robust low-rank matrix approximations. This excellent work proves that, under certain assumptions in regard to the structure of the error matrix, the noise levels, and the rank of the underlying matrix, a low-rank matrix corrupted by a sparse-error matrix can be recovered exactly with very high likelihood. However, these assumptions are not guaranteed to hold in practical applications, in which case the recovered matrices can be entirely incorrect.

The work in [17] also needs mentioning. Here, the authors apply Branch and Bound and convex underestimators to the general problem of bilinear problems, which includes (1), both under $L_1$ and $L_2$ norms. This approach is provably globally optimal, but is, in general, very time consuming and, in practice, only useful for small scale problems.

### 2.1 The Wiberg Algorithm

As previously mentioned, the Wiberg algorithm is a numerical method developed for low-rank matrix approximation using the $L_2$-norm in the case where some of the data is missing. This section provides a brief description of the underlying ideas behind this method in an attempt to motivate some of the steps taken in the derivation of our generalized version to come.

The Wiberg algorithm is based on the observation that, for a fixed $U$, the $L_2$-norm version of (1) becomes a linear, least-squares minimization problem in $V$:

$$\min_v \|Wy - W(I_n \otimes U)v\|_2^2, \tag{3}$$

where $W = \mathrm{diag}(\hat{w})$. The closed-form solution for the optimal is

$$v^*(U) = (G(U)^T G(U))^{-1} G(U) W y, \tag{4}$$

where $G(U) = W(I_n \otimes U)$. Similarly, for a fixed $V$, (1) becomes a linear least-squares minimization problem in $U$:

$$\min_u \left\|Wy - W\left(V^T \otimes I_m\right)u\right\|_2^2, \tag{5}$$

with the optimal $u$ given by

$$u^*(V) = (F(V)^T F(V))^{-1} F(V) W y, \tag{6}$$

where $F(V) = W(V^T \otimes I_m)$.

Here, it should be mentioned that alternatively fixing $U$ while updating $V$, and vice versa (using (4) and (6)), was one of the earliest algorithms for finding matrix approximations in the presence of missing data. This process is known as the Alternated Least Squares (ALS) approach. The disadvantage of this approach, however, is that it has in practice been shown to converge very slowly (see [7], for example). The alternated LP and QP approaches of [13] were motivated by this method.

Continuing with the Wiberg approach, by substituting (4) into (5) we see that the optimum of (5) is also the optimum of

$$\min_U \|g(U)\|_2^2 = \min_U \|Wy - W\mathrm{vec}(UV^*(U))\|_2^2, \tag{7}$$

where the function $g(U) = Wy - W\mathrm{vec}(UV^*(U))$ is introduced to emphasize the fact that (7) represents a nonlinear least-squares problem in $U$. It is the application of the Gauss-Newton method [18] to the above problem that results in the Wiberg algorithm. The difference between the Wiberg algorithm and ALS may thus be interpreted as the fact that the former effectively computes Gauss-Newton

updates while the latter carries out exact cyclic coordinate minimization.

As such, the Wiberg algorithm generates a sequence of iterates $U_{k+1}$ by repeatedly calculating a first-order Taylor expansion of $g$ at $U_k$ and solving the resulting subproblem:

$$\min_{\delta} \|g(U_k) + \frac{\partial g(U_k)}{\partial U}\ (u_k - \delta)\|_2^2. \tag{8}$$

If we let $J_k$ denote the Jacobian $\frac{\partial g(U_k)}{\partial U}$, we can write the solution to (8) as

$$\delta_k^* = \left(J_k^T J_k\right)^{-1} J_k^T W y, \tag{9}$$

the well-known normal equation. The next iterate is then given by

$$U_{k+1} = U_k + \delta_k^*. \tag{10}$$

## 3 Differentiating the $L_1$ Solution of an Overdetermined System of Linear Equations

In the previous work [19], the derivation of an $L_1$ version of the Wiberg algorithm was based on the differentiation of linear programs in canonical form. Here, we present a slightly different derivation resulting in a simpler and more computationally efficient formulation.

This section deals with the sensitivity of the $L_1$ solution of an overdetermined system of linear equations with respect to changes in the coefficients. The problem we are considering is the following:

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_1, \tag{11}$$

$b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $m \geq n$. It can furthermore be assumed, for our purposes and without loss of generality, that $A$ has full rank.

Note that (11) is equivalent to the linear program

$$\begin{aligned} &\min 1^T t \\ &\text{s.t.} -t \leq b - Ax \leq t \\ &\quad x \in \mathbb{R}^n, \quad t \in \mathbb{R}^m. \end{aligned} \tag{12}$$

Now, from [20] we have the following theorem and corollary.

**Theorem 3.1.** *Let $X^*$ denote the set of all minimizers of the convex optimization problem*

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_1. \tag{13}$$

*Then, there always exists a solution $x^* \in X^*$ such that $b - Ax^*$ has at least $n$ entries equal to zero.*

**Corollary 3.1.** *The submatrix of $A$ consisting of the rows of $A$ corresponding to $Z(x^*)$ must have rank $n$ for some solution $x^*$ to 13.*

For proofs see [20, Section 6]. Here, $Z(x^*)$ denotes the set of indices for which the residual is zero, $Z(x^*) = \{i = 1, \ldots, m | [b - Ax^*]_i = 0\}$. We also include an additional corollary which follows trivially from the above.

**Corollary 3.2.** *Let $Z_n(x^*)$ denote all subsets of $Z(x^*)$ containing $n$ elements. Then, there exists a $z \in Z_n(x^*)$ such that the submatrix consisting of the corresponding rows of $A$ is of full rank.*

Assuming that a minimizer $x^*$ of the linear program (11), or equivalently (12), has been obtained using some optimization algorithm, we are interested in how this minimizer changes as we alter the coefficients of the linear equation system. That is, we wish to compute the partial derivatives $\partial x^*/\partial A_{ij}$ and $\partial x^*/\partial b_i$.

**Theorem 3.2.** *Let $x^*$ be the minimizer (11). We further assume that the $n$-by-$n$ submatrix of Corollary 3.2, denoted $B$, is unique. Reordering the rows of $A$ and $b$, if necessary, there exists one or more partitions such that*

$$A = \begin{bmatrix} B \\ N \end{bmatrix}, \tag{14}$$

$$b = \begin{bmatrix} b_B \\ b_N \end{bmatrix}. \tag{15}$$

*Then, $x^*$ is differentiable at $A$, $b$ with the partial derivatives given by*

$$\frac{\partial x^*}{\partial B} = -(x^*)^T \otimes B^{-1}, \tag{16}$$

$$\frac{\partial x^*}{\partial N} = 0, \tag{17}$$

$$\frac{\partial x^*}{\partial b_B} = B^{-1}, \tag{18}$$

$$\frac{\partial x^*}{\partial b_N} = 0. \tag{19}$$

**Proof.** According to Theorem 3.1, we have that

$$b_B - Bx^* = 0; \tag{20}$$

by Corollary 3.2, $B$ is of full rank, so

$$x^* = B^{-1} b_B. \tag{21}$$

Since $B$ is a smooth bijection from $\mathbb{R}^n$ onto itself, it follows that $x^*$ is differentiable with respect to the coefficients in $A$ and $b$. Differentiating (21) gives (16):

$$\begin{aligned} \frac{\partial x^*}{\partial B} &= \frac{\partial}{\partial B}(B^{-1} b_B) \\ &= \left(b_B^T \otimes I_m\right) \frac{\partial B^{-1}}{\partial B} \\ &= -\left(b_B^T \otimes I_m\right)(B^{-T} \otimes B^{-1}) \\ &= -(x^*)^T \otimes B^{-1}. \end{aligned}$$

Equations (17), (18), and (19) follow trivially. □

## 4 The $L_1$-Wiberg Algorithm

In this section, we present the derivation of a generalization of the Wiberg algorithm to the $L_1$-norm. We follow a similar approach to the derivation of the standard Wiberg algorithm above, that is, by rewriting the problem as a

function of $U$ only, then linearizing it, solving the resulting subproblem, and updating the current iteration using the minimizer of said subproblem.

Our starting point for the derivation, our generalization of the Wiberg algorithm, is the minimization problem

$$\begin{aligned}\min_{U,V} f'(U,V) &= \|g'(U,V)\|_1 \\ &= \|\hat{W} \odot (Y - UV)\|_1.\end{aligned} \tag{22}$$

Following the approach of Section 2.1, we first note that for fixed $U$ and $V$ it is possible to rewrite the optimization problem (22) as

$$v^*(U) = \arg\min_v \|Wy - W(I_n \otimes) v\|_1 \tag{23}$$

and

$$u^*(V) = \arg\min_u \|Wy - W(V^T \otimes I_m)u\|_1 \tag{24}$$

linear problems in $V$ and $U$, respectively.

Substituting (23) into (24), we obtain

$$\begin{aligned}U^* &= \arg\min_U f(U) \\ &= \arg\min_U f'(U, V^*(U))\end{aligned} \tag{25}$$

$$\begin{aligned}&= \arg\min_U \|g'(U, V^*(U))\|_1 \\ &= \arg\min_U \|Wy - W\mathrm{vec}(UV^*(U))\|_1 \\ &= \arg\min_U \|g(U)\|_1.\end{aligned} \tag{26}$$

Unfortunately, (26) is not a least-squares minimization problem, so the Gauss-Newton algorithm is not applicable. Note also that $V^*$ does not have an easily differentiable, closed-form solution, but the results of the previous section allow us to continue in a similar fashion.

Let $V^*(U)$ denote the optimal solution of (23) and $z$ is the corresponding set of $n$ indices of Corollary 3.2. Assuming that the prerequisites of Theorem 3.2 hold, then $V^*(U)$ is differentiable and we can compute the Jacobian of the nonlinear function $\phi_1(U)$. Denote by $Q$ the $n$-by-$mn$ matrix obtained by removing the rows not corresponding to indices in $z$ from the identity matrix $I_{mn}$. Then, we can write

$$v^* = G_B^{-1} QWy = (QG)^{-1} QWy. \tag{27}$$

Using (16) and applying the chain rule, we obtain

$$\frac{\partial v^*}{\partial U} = \frac{\partial}{\partial U}\left(G_B^{-1} QWy\right) = -\left((v^*)^T \otimes G_B^{-1}\right)\frac{\partial G_B}{\partial U} = -G_B^{-1}\frac{\partial G_B}{\partial U} v^* \tag{28}$$

and

$$\begin{aligned}&\frac{\partial G_B}{\partial U} v = \frac{\partial}{\partial U}(G_B v) = \frac{\partial}{\partial U}(QW(I_n \otimes U)v) \\ &= \frac{\partial}{\partial U}\left(QW\left(V^T \otimes I_m\right)u\right) = QW\left(V^T \otimes I_m\right) = QF = F_B.\end{aligned} \tag{29}$$

Combining the above expressions, we see that

$$J(U) = \frac{\partial g}{\partial U} = F + G\frac{\partial v^*}{\partial U} = F - GG_B^{-1}F_B. \tag{30}$$

The Gauss-Newton method, in the least-squares case, works by linearizing the nonlinear part and solving the resulting subproblem. By (30) the same can be done for $g(U)$.

Using (30), the first-order Taylor expansion of $g$ results in the following approximation of $f$ (from (25)) around $U_k$:

$$f(\delta) \approx q_k(\delta) = \|g(U_k) + J(U_k)\ \delta\|_1. \tag{31}$$

This allows the construction of an approximation to (26):

$$\min_\delta \|g(U_k) + J(U_k)\delta\|_1, \tag{32}$$

and as in the $L_2$ case this is a linear problem, but now in $\delta$. This minimization problem may be stated as

$$\begin{aligned}&\min_{\delta,t} \begin{bmatrix}0 & 1^T\end{bmatrix}\begin{bmatrix}\delta \\ t\end{bmatrix} \\ &\text{s.t.} \begin{bmatrix}-J(U_k) & -I \\ J(U_k) & -I\end{bmatrix}\begin{bmatrix}\delta \\ t\end{bmatrix} \le \begin{bmatrix}-g(U_k) \\ g(U_k)\end{bmatrix} \\ &\quad \|\delta\|_1 \le \mu_k \\ &\quad \delta \in \mathbb{R}^{mr}, \quad t \in \mathbb{R}^{mn}.\end{aligned} \tag{33}$$

Let $\delta_k^*$ be the minimizer of (33), then the update rule for our proposed method is again given by

$$U_{k+1} = U_k + \delta_k^*. \tag{34}$$

Note that in (33), we have added the constraint $\|\delta\|_1 \le \mu_k$. This is done as a trust region strategy to limit the step sizes that can be taken at each iteration to ensure a nonincreasing sequence of iterates. See below for details on how the step length $\mu_k$ is handled. We are now ready to present our complete $L_1$-Wiberg method in Algorithm 1.

**Algorithm 1.** $L_1$-Wiberg Algorithm.

**Input:**
$U_0 \in \mathbb{R}^{m\times r}$, $\mu_0 > 0$, $1 > \eta_2 > \eta_1 > 0$ and $c > 1$
$k = 0$.
$V_0 = V^*(U_0)$
**repeat**
  Compute the Jacobian $\nabla\phi_1 = J(U_k)$ using (30)
  Solve the subproblem (33) to obtain $\delta_k^{*\dagger}$
  Let $\rho_k = \frac{\Delta f_k}{\Delta q_k} = \frac{f(U_k) - f(U_k + \delta_k^*)}{f(U_k) - q(\delta_k^*)}$
  **if** $\rho_k \le \eta_1$ **then**
    $\mu_{k+1} = \eta_1 \|\delta_k^*\|_1$
  **end if**
  **if** $\rho_k \ge \eta_2$ **then**
    $\mu_{k+1} = c\mu_k$
  **end if**
  **if** $\rho_k \ge \epsilon$ **then**
    $U_{k+1} = U_k + \delta_k^*$
    $V_k + 1 = V^*(U_k + \delta_k^*)$
    $k = k + 1$
  **end if**
**until** convergence

Typical parameter values used were $\mu_0 = 1$, $\eta_1 = \frac{1}{4}$, $\eta_2 = \frac{3}{4}$, $\epsilon = 10^{-3}$, and $c = 2$.

$^\dagger$ If $\delta_k^* = 0$ and $(U_k, V_k)$ is not a stationary point of (1) then simply chose a different $z \in Z$ and repeat 6.

Proper initialization is a crucial issue for any iterative algorithm and can greatly affect its performance. Obtaining this initialization is highly problem dependent; for certain applications good initial estimates of the solution are readily available and for others finding a sufficiently good $U_0$ might be considerably more demanding. In this work, we either initialized our algorithm randomly or through the rank-r truncation of the singular-value decomposition of $\hat{W} \otimes Y$.

## 5 Proof of Convergence

In this section, we show under what conditions Algorithm 1 converges to a stationary point of (22). The starting point of this proof is the work on nonsmooth optimization by [21], [22], [23], [24]. There, the problem investigated is the unconstrained minimization of nonsmooth, composite functions of the form $h(f(x))$, where $h$ is a nonsmooth convex function from $\mathbb{R}^m$ to $\mathbb{R}$ and $f$ is a continuously differentiable function $\mathbb{R}^n$ to $\mathbb{R}^m$. They propose an iterative trust region algorithm to solve this problem. At each iteration the following convex subproblem is solved:

$$\min_{\|\Delta\| \leq \tau_k} \phi_k(\Delta) = h(f(x_k) + \nabla f(x_k)\Delta) + \Delta^T B_k \Delta, \tag{35}$$

where $B_k$ is some symmetric $n \times n$ matrix. The updating of $x_k$ and $\mu_k$ are handled in the same manner as in Algorithm 1.

It was shown in the above work that under certain conditions, most significantly that $\|B_k\|$ is uniformly bounded, then the above algorithm will have an accumulation point $x^*$ which is a stationary point of (1); see [23] for details.

Since here, in subproblem (26), the function $g(U) = Wy - \mathrm{vec}(UV^*(U))$ is not differentiable everywhere, we cannot apply this result directly. However, we are still able to show objective convergence to a stationary point of the proposed algorithm in the following theorem.

**Theorem 5.1.** *Let $\{U_k, V_k\}_{k=1}^{\infty}$, with $V_k = V^*(U_k)$, denote the sequence produced by Algorithm 1. Then, $f'(U_k, V_k)$ will converge to a stationary point of (1).*

**Proof.** As mentioned above, since $g(U)$ is not differentiable everywhere we cannot apply the result of [21], [22], [23], [24] directly. However, we know that the function $f'(U, V)$ of (22) is a smooth function, and consequently any convergent sequence of symmetric matrices $B_k$ will generate a sequence of iterates $\{U_k, V_k\}_{k=1}^{\infty}$ whose function values converge to a stationary point of (22). If we can show that there exists such a sequence $\{B_k\}_{k=1}^{\infty}$ that produces the same sequence of iterates $\{U_k, V_k\}_{k=1}^{\infty}$ as Algorithm 1, then it must follow that Algorithm 1 has objective convergence to a stationary point of (22).

With $h(.) = \|.\|_1$, let $\partial h(u_k, v_k)$ denote the subdifferential of $h$ at $(u_k, v_k)$. Also, let $O$ denote the set $\{i = 1, \ldots, mn | g_i(U_k) = 0,\ i \notin z\}$ and $N$ is its complement. Then, (35) becomes

$$\min_{\|\Delta\| \leq \tau_k} \phi_k(\Delta) = \|g(U_k) + [\,F \quad G\,]\Delta\|_1 + \Delta^T B_k \Delta. \tag{36}$$

With first-order optimality conditions given by

$$\begin{bmatrix} F^T \\ G^T \end{bmatrix} \Lambda + 2B_k\Delta + \nu\xi = 0, \tag{37}$$

$\Lambda \in \partial h(u_k + \Delta_u, v_k + \Delta_v)$, $\Delta = \left[\begin{smallmatrix}\Delta_u \\ \Delta_v\end{smallmatrix}\right]$, $\xi \in \partial h(\Delta_u, \Delta_v)$, and $\nu \geq 0$ a Lagrangian multiplier. If $\Delta$ is bounded away from zero, then the above system equations is underdetermined and linear in the entries of $B_k$ and there must exist a bounded and symmetric matrix $B_k$ such that (37) holds. Specifically, there exist a bounded $B_k$ such that the solution of (36) fulfills $\left[\begin{smallmatrix}u_{k+1} \\ v_{k+1}\end{smallmatrix}\right] = \left[\begin{smallmatrix}u_k \\ v_k\end{smallmatrix}\right] + \Delta^*$.

Due to the scale ambiguity of $f'(U, V)$ we can, without loss of generality and for the purpose of this proof, assume that $\|U_k\| \leq 1$, then the level set $\mathcal{L} = \{U, V \mid \|U_k\| \leq 1, f'(U, V) \leq f'(U_0, V_0)\}$ is compact and, consequently, the sequence $\{U_k, V_k\}_{k=1}^{\infty}$ is bounded. It was shown in [23], [24] that $\mu \to \bar{\mu} > 0$, i.e., that the trust region constraints are not active for $k$ greater than some $k_0$. Then, if $\Delta = 0$, by the first-order conditions of (23) and (33), there must exist a $\bar{\lambda} = \left[\begin{smallmatrix}\bar{\lambda}_B \\ \bar{\lambda}_O\end{smallmatrix}\right] \in \partial h(\bar{U}, \bar{V})$ such that

$$\left(F_O^T - F_B^T G_B^{-T} G_O^T\right)\bar{\lambda}_O + \left(F_N^T - F_B^T G_B^{-T} G_N^T\right) s_N(\bar{U}, \bar{V}) = 0, \tag{38}$$

$$G_B^T \bar{\lambda}_B + G_O^T \bar{\lambda}_O + G_N^T s_N(\bar{U}, \bar{V}) = 0, \tag{39}$$

where $s_N(U, V) = \mathrm{sgn}(\mathrm{vec}(UV))$ for the indices in $N$. By elimination, we obtain

$$\begin{bmatrix} F_B^T & F_O^T & F_N^T \\ G_B^T & G_O^T & G_N^T \end{bmatrix} \begin{bmatrix} \bar{\lambda}_B \\ \bar{\lambda}_O \\ s_N(\bar{U}, \bar{V}) \end{bmatrix} = 0. \tag{40}$$

Then,

$$\Lambda = \begin{bmatrix} \bar{\lambda}_B \\ \bar{\lambda}_O \\ s_N(\bar{U}, \bar{V}) \end{bmatrix}$$

is a solution to (37) with $B_k = \bar{B} = 0$, and it follows that there exists a sequence of symmetric matrices $B_k$ for which $\|B_k\|$ is bounded and such that (36) produces the same sequence of iterates $\{U_k, V_k\}_{k=1}^{\infty}$ as Algorithm 1.

Finally, as the sequence $\{f'(U_k, V_k)\}_{k=1}^{\infty}$ is convergent by construction, objective convergence to a stationary point follows. □

## 6 Experiments

In this section, we present a number of experiments carried out to evaluate our proposed method. These include real and synthetic tests.

We have evaluated the performance of the $L_1$-Wiberg algorithm against other available methods, including those of Ke and Kanade in [13] (alternated LP and alternated QP).

We did also look into the method of [16], but as this approach failed to produce any sensible results in the real-world applications of Sections 6.2, 6.3, and 6.4, we omitted such a comparison altogether.

All algorithms were implemented in Matlab. Linear and quadratic optimization subproblems were solved using the package MOSEK.

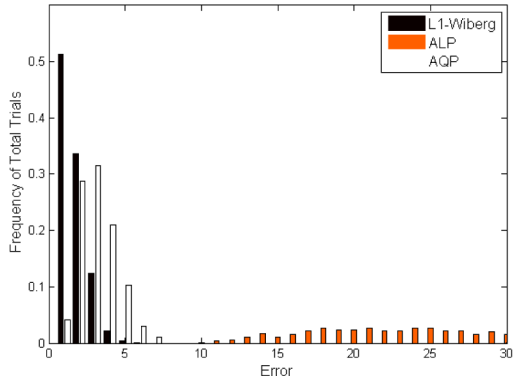



Fig. 1. A histogram representing the frequency of different magnitudes of error in the estimate generated by each of the methods.

### 6.1 Synthetic Data

The aim of the experiments in this section was to empirically obtain a better understanding of the following properties of each of the tested algorithm: resulting error, rate of convergence, execution time, and the computational requirements.

For the synthetic tests, a set of randomly created measurement matrices was generated. The elements of each measurement matrix $Y$ were drawn from a uniform distribution in the range $[-1,\ 1]$. A set of 20 percent of the elements of $Y$ wsd chosen at random (again from a uniform distribution) and designated as missing by setting the corresponding entry in the matrix $\hat{W}$ to 0. In addition, to simulate outliers, uniformly distributed noise over $[-5,\ 5]$ was added to a randomly selected 10 percent of the elements in $Y$. Three different problem sizes were considered: $(m = 10,\ n = 15, r = 5)$, $(m = 25,\ n = 50, r = 7)$, and $(m = 40,\ n = 150, r = 9)$.

Fig. 1 shows a histogram of the error produced by each algorithm on 100 synthetic matrices, created as described above. It can be seen in this figure that our proposed method clearly outperforms the other two. But what should also be noted here is the poor performance of the alternated linear program approach. Even though it can easily be shown that this algorithm will produce a nonincreasing sequence of iterates, there is no guarantee that it will converge to a local minima. This is what we believe is actually occurring in these tests. The alternated linear program converges to a point that is not a local minima, typically after only a small number of iterations. Due to its poor performance, we have excluded this method from the remainder of experiments.

Next, we examine the convergence rate of the algorithms. A typical instance of the error convergence from both the AQP and $L_1$-Wiberg algorithms, applied to one of the synthetic problems, can be seen in Fig. 2. These figures are not intended to show the quality of the final solution, but rather how it quickly it is obtained by the competing methods.

Fig. 3 depicts the performance of the algorithms in 100 synthetic tests and is again intended to show convergence rate rather than the final error. Note the independent scaling of each set of results and the fact that the $Y$-axis is on a logarithmic scale. Again it is obvious that the $L_1$-Wiberg algorithm significantly outperforms the alternated quadratic programming approach. It can be seen that

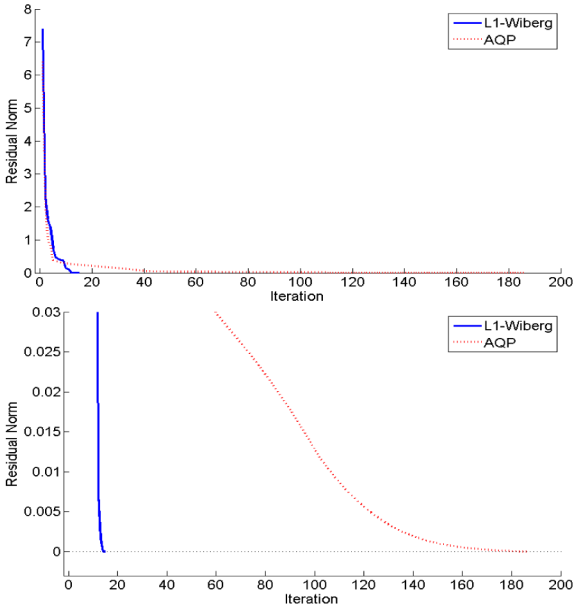



Fig. 2. Plots showing the norm of the residual at each iteration of two randomly generated tests for both the $L_1$ Wiberg and alternated quadratic.

the latter method has a tendency to flatline, that is, to converge very slowly after an initial period of more rapid progress. This is a behavior that has also been observed for alternated approaches in the $L_2$ instance, see [7].

Table 1 summarizes the same set of synthetic tests. What should be noted here is the low average error produced by our method, the long execution time of the alternated quadratic program approach, and the poor results obtained by the alternated linear program method.

The results of these experiments, although confined to smaller scale problems, do indeed demonstrate the promise of our suggested algorithm.

### 6.2 Structure from Motion

Next, we present an experiment on a real-world application, namely structure from motion. We use the well-known dinosaur sequence, available from http://www.robots.ox.ac.uk/~vgg/, containing projections of 319 points tracked over 36 views. Now, finding the full 3D-reconstruction of this scene can be posed as a low-rank matrix approximation

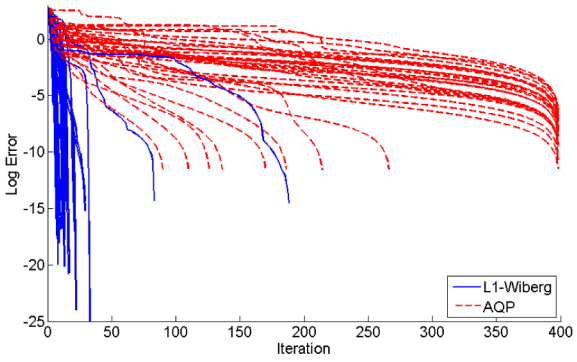



Fig. 3. A plot showing the convergence rate of the alternated quadratic programming and $L_1$-Wiberg algorithms over 100 trials. The error is presented on a logarithmic scale.

TABLE 1
The Averaged Results from Running 100 Synthetic Experiments on Three Different Problem Sizes

| $m = 10,\ n = 15,\ r = 5$ | Alt. LP [13] | Alt. QP [13] | Wiberg $L_1(Alg.1)$ |
|---|---|---|---|
| Error $(L_1)$ | 71.62 | 4.53 | 3.08 |
| Execution Time (sec) | 0.42 | 7.85 | 1.17 |
| # Iterations | 8.17 | 180.17 | 24.78 |
| # LP/QP solved | 16.33 | 360.33 | 61.97 |

| $m = 25,\ n = 50,\ r = 7$ | Alt. LP [13] | Alt. QP [13] | Wiberg $L_1(Alg.1)$ |
|---|---|---|---|
| Error $(L_1)$ | 328.66 | 23.23 | 11.67 |
| Execution Time (sec) | 8.11 | 68.12 | 18.58 |
| # Iterations | 9.69 | 201.95 | 32.05 |
| # LP/QP solved | 19.38 | 403.90 | 80.13 |

| $m = 40,\ n = 150,\ r = 9$ | Alt. LP [13] | Alt. QP [13] | Wiberg $L_1(Alg.1)$ |
|---|---|---|---|
| Error $(L_1)$ | 1484.78 | 153.51 | 71.55 |
| Execution Time (sec) | 4.64 | 1987.71 | 203.81 |
| # Iterations | 26.03 | 286.11 | 93.6 |
| # LP/QP solved | 52.06 | 572.22 | 205.92 |

* The alternated QP algorithm was terminated after 400 iterations and 800 solved quadratic programs.

task. In addition, as we are considering robust approximation in this work, we also included outliers to the problem by adding uniformly distributed noise $[-50,\ 50]$ to 10 percent of the tracked points.

We applied our proposed method to this problem, initializing it using truncated singular-value decomposition as described in the previous section. For comparison, we also include the result from running the standard Wiberg algorithm. Attempts to evaluate the AQP method on this on the same data were abandoned when the execution time exceeded several hours.

The residual for the visible points of the two different matrix approximations is given in Fig. 4. The $L_2$-norm approximation seems to be strongly affected by the presence of outliers in the data. The error in the matrix approximation appears to be evenly distributed among all the elements of the residual. In the $L_1$ case this does not seem to occur. Instead, the reconstruction error is concentrated to a few elements of the residual. The root mean square error of the inliers only was 2.029 for the $L_2$-Wiberg algorithm and 0.862 for the $L_1$-Wiberg algorithm. Total execution times were 3 min, 2 sec and 17 min, 44 sec, respectively. The resulting reconstructed scene can be seen in Fig. 5.

### 6.3 Eigenfaces

Eigenfaces is a classical tool for analyzing images of human faces; the earliest application was presented in [4]. Given a number of training images, the eigenface method finds the $K$-dimensional linear subspace that best describes this data. If one disregards any orthogonality constraint, this task can be viewed as a low-rank matrix approximation problem.

In this section, we show how using our proposed method allows for the construction of a more eigenface decomposition. We used 30 grayscale images randomly selected from the CBCL data set [26], rescaled to a size of 38-by-38 pixels. The dimension of the subspace was set to $K = 5$. To simulate outliers we also included three additional images, significantly different from those of human faces, Fig. 6, into the training set.

Algorithm 1 was then applied to find the best rank-2, $L_1$ approximation to the resulting 1,444-by-33 matrix of face images. Fig. 7 shows the resulting reconstructed face images using the $L_1$ formulation as well as the standard $L_2$-based eigenface method.

The effect of the outlier, nonface images can clearly be observed by the poorer performance of the standard eigenface method. The actual reconstruction error for each of the images can be seen in Fig. 8.

### 6.4 Nonrigid Motion Recovery

It was shown in [27] that the recovery of the shape of a nonrigidly moving object from an image sequence can be posed as a low-rank matrix approximation problem. Under the assumption that that the nonrigid motion can be described as the linear combination of $K$ different shape

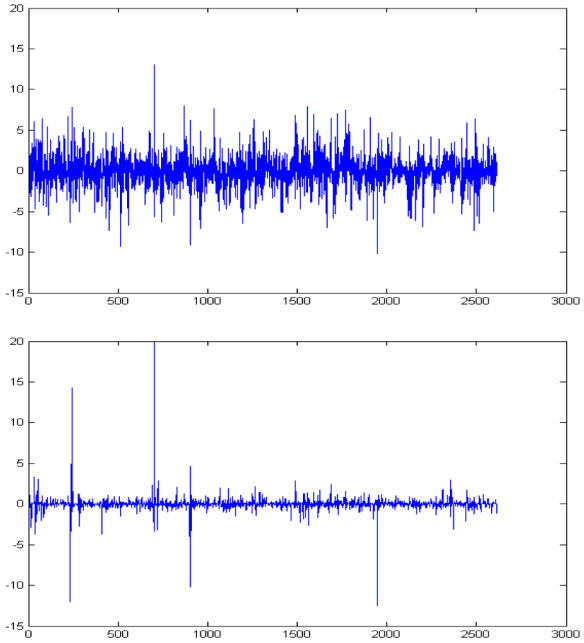



Fig. 4. Resulting residuals using the standard Wiberg algorithm (top) and our proposed $L_1$-Wiberg algorithm (bottom).

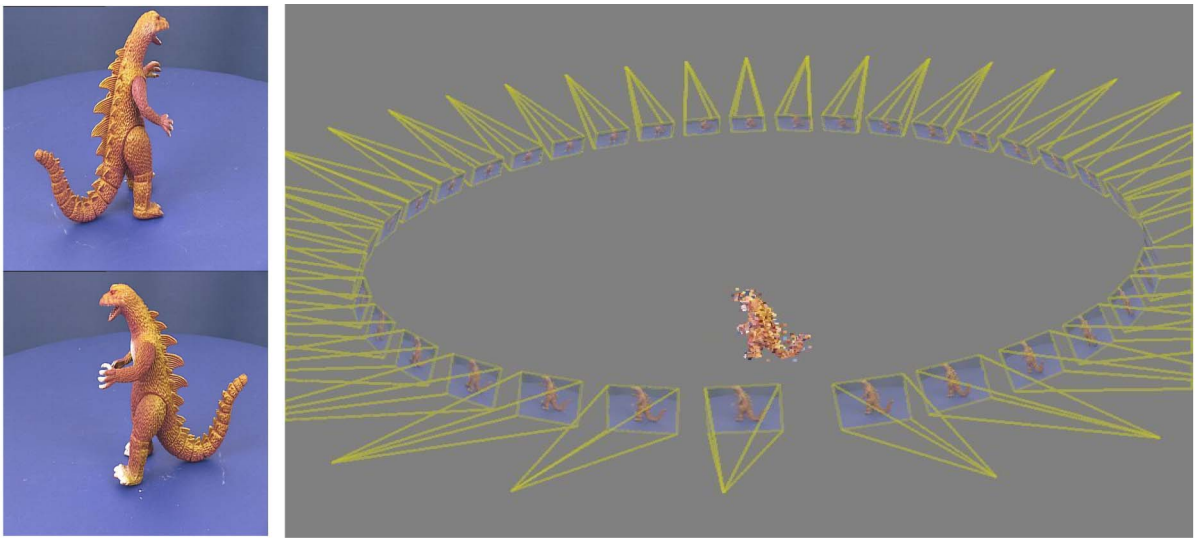


Fig. 5. Images from the dinosaur sequence and the resulting reconstruction using our proposed method. (Visualization tool courtesy of [25].)

basis, the tracked points in the sequence will span a $3K$-dimensional subspace.

In this section, we use the shark sequence of [27] for experimental validation. This synthetic data set consists of 91 points on a nonrigid shape (a shark) tracked along sequence of 240 frames, under varying camera pose.

Fig. 6. Outliers, images not containing faces.

Fig. 7. Left: Original images. Middle: Reconstructed image using $L_1$-norm. Right: Reconstructed image using $L_2$-norm.

In order to obtain more realistic conditions Gaussian noise with a variance of 3 pixels was added. In addition, 20 percent of the tracked points were labeled missing and 10 percent of the tracked points were replaced by gross outliers, drawn from a uniform distribution $[-25, 25]$. The number of shape basis was set to $K = 2$.

The resulting shape reconstructions using the $L_1$ and $L_2$ approximations can be seen in Fig. 9. The actual reconstruction error for a subset of the 240 frames can be seen in Fig. 10. Here, it can be seen that the $L_1$ clearly outperforms the $L_2$ formulation.

## 7 Conclusion

In this paper, we have studied the problem of low-rank matrix approximation in the presence of missing data. We have proposed a method for solving this task under the robust $L_1$ norm which can be interpreted as a generalization of the standard Wiberg algorithm. We have also shown through a number of experiments on both synthetic and real world data that the $L_1$-Wiberg method proposed is both practical and efficient and performs very well in comparison to other existing methods.

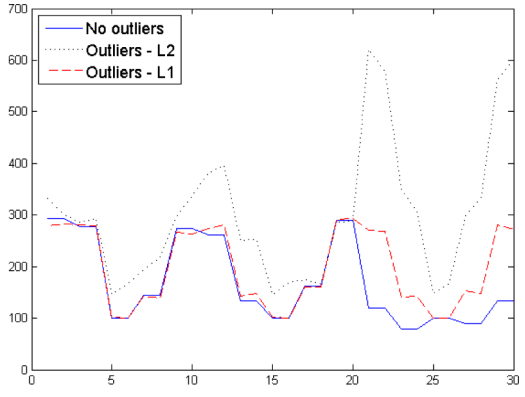



Fig. 8. Reconstruction error for the 30 face images.

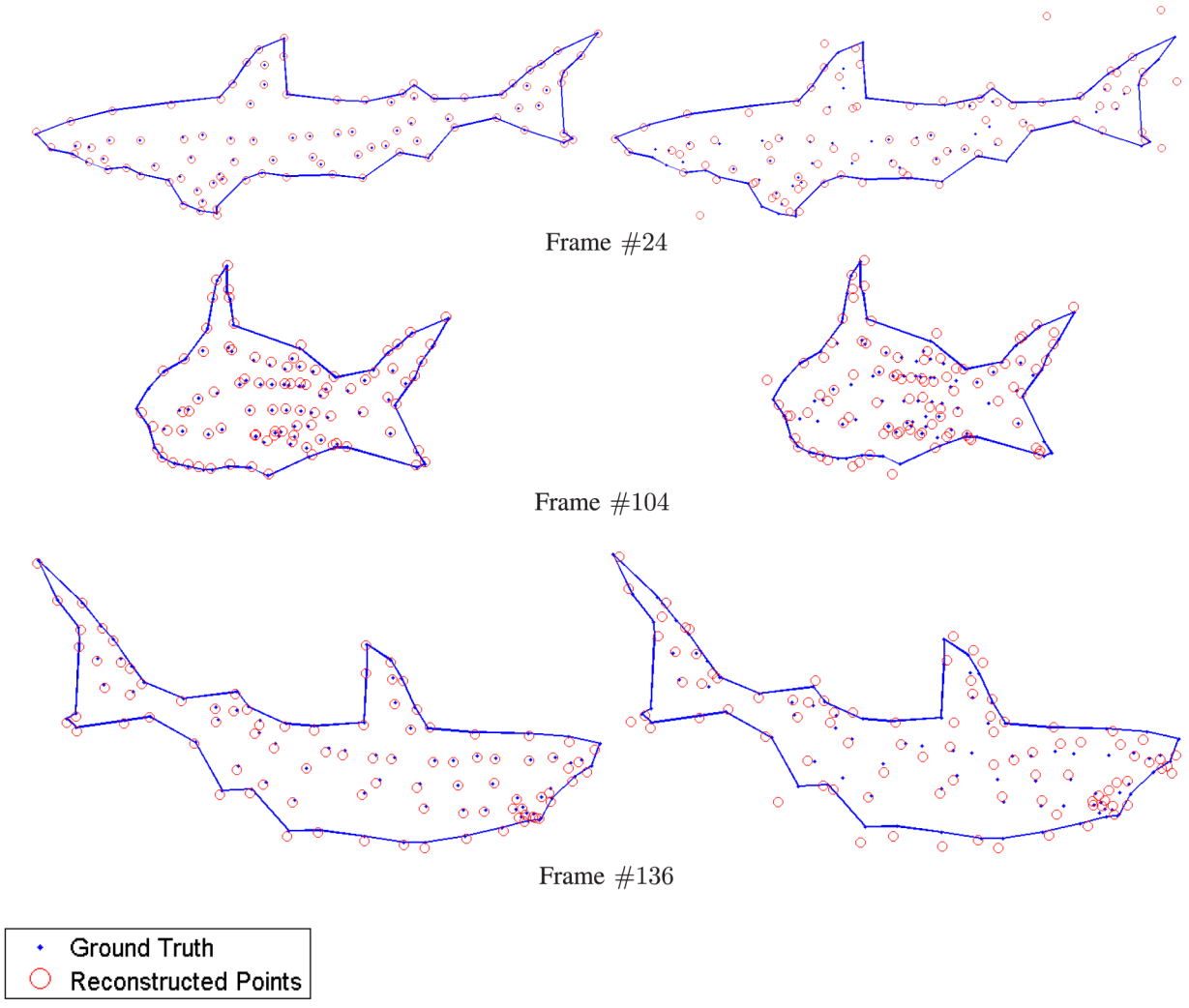



Fig. 9. Nonrigid shape reconstruction results for three different frames of the shark sequence. Left column: $L_1$ Approximation. Right column: $L_2$ Approximation.

## ACKNOWLEDGMENTS

The authors would like to thank Carl Olsson and Fredrik Kahl of Lund University for the use of their visualization tool. This research was supported under the Australian Research Council's Discovery Projects funding scheme (project DP0988439).

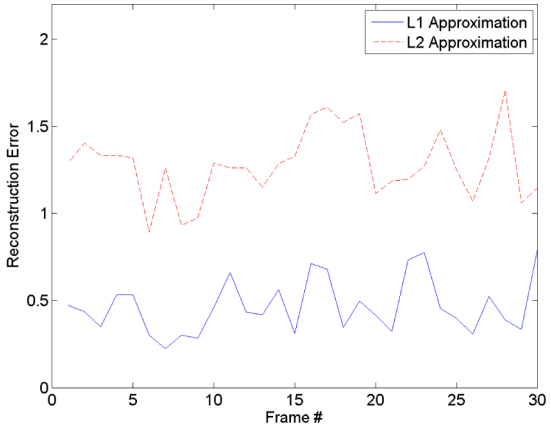



Fig. 10. Average reconstruction error for 30 frames of the shark sequence using both the $L_1$ and $L_2$ formulations.

## REFERENCES

[1] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *Int'l J. Computer Vision*, vol. 9, no. 2, pp. 137-154, 1992.

[2] H.-Y. Shum, K. Ikeuchi, and R. Reddy, "Principal Component Analysis with Missing Data and Its Application to Polyhedral Object Modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 854-867, Sept. 1995.

[3] Q. Ke and T. Kanade, "A Subspace Approach to Layer Extraction," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 2001.

[4] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

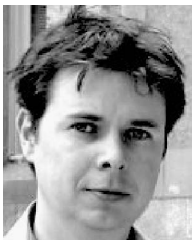
[5] H. Hayakawa, "Photometric Stereo under a Light Source with Arbitrary Motion," *J. Optical Soc. of Am. A*, vol. 11, no. 11, pp. 3079-3089, 1992.

[6] T. Wiberg, "Computation of Principal Components When Data Are Missing," *Proc. Second Symp. Computational Statistics*, pp. 229-236, 1976.

[7] A.M. Buchanan and A.W. Fitzgibbon, "Damped Newton Algorithms for Matrix Factorization with Missing Data," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 316-322, 2005.

[8] T. Okatani and K. Deguchi, "On the Wiberg Algorithm for Matrix Factorization in the Presence of Missing Components," *Int'l J. Computer Vision*, vol. 72, no. 3, pp. 329-337, 2007.
[9] P. Baldi and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima," *Neural Networks*, vol. 2, no. 1, pp. 53-58, 1989.
[10] E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *J. Math. Biology*, vol. 15, pp. 267-273, 1982.
[11] F. De La Torre and M.J. Black, "A Framework for Robust Subspace Learning," *Int'l J. Computer Vision*, vol. 54, nos. 1-3, pp. 117-142, 2003.
[12] H. Aanaes, R. Fisker, K. Astrom, and J.M. Carstensen, "Robust Factorization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1215-1225, Sept. 2002.
[13] Q. Ke and T. Kanade, "Robust $L_1$ Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, pp. 739-746, 2005.
[14] M.J. Black and A.D. Jepson, "Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Int'l J. Computer Vision*, vol. 26, pp. 329-342, 1998.
[15] C. Croux and P. Filzmoser, "Robust Factorization of a Data Matrix," *Proc. Computational Statistics*, pp. 245-249, 1998.
[16] E.J. Candès, X. Li, Y. Ma, and J. Wright, "Robust Principal Component Analysis?" *Computing Research Repository*, vol. abs/0912.3599, 2009.
[17] M.K. Chandraker and D.J. Kriegman, "Globally Optimal Bilinear Programming for Computer Vision Applications," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.
[18] A. Bjorck, *Numerical Methods for Least Squares Problems*. SIAM, 1995.
[19] 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," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.
[20] G.A. Watson, *Approximation Theory and Numerical Methods*. Wiley, 1980.
[21] R. Fletcher, "A Model Algorithm for Composite Nondifferentiable Optimization Problems," *Math. Programming Study*, vol. 17, pp. 67-76, 1982.
[22] R. Fletcher, *Practical Methods of Optimization*. second ed., Wiley-Interscience, 1987.
[23] Y. Yuan, "Some Properties of Trust Region Algorithms for Nonsmooth Optimization," technical report, 1983.
[24] Y. Yuan, "On the Superlinear Convergence of a Trust Region Algorithm for Nonsmooth Optimization," *Math. Programming*, vol. 31, no. 3, pp. 269-285, 1985.
[25] C. Olsson and F. Kahl, "Generalized Convexity in Multiple View Geometry," *J. Math. Imaging and Vision*, vol. 38, pp. 35-51, 2010.
[26] M. Alvira and R. Rifkin, "An Empirical Comparison of SNoW and SVMs for Face Detection," A.I. memo 2001-004, Center for Biological and Computational Learning, MIT, 2001.
[27] L. Torresani, A. Hertzmann, and C. Bregler, "Learning Non-Rigid 3D Shape from 2D Motion," *Proc. Neural Information Processing Systems*, 2003.

**Anders Eriksson** received the Msc degree in electrical engineering and the PhD degree in mathematics from Lund University, Sweden, in 2000 and 2008, respectively. Currently, he is a senior research associate at the University of Adelaide, Australia. His research interests include optimization theory and numerical methods applied to the fields of computer vision and machine learning.



**Anton van den Hengel** received the bachelor of mathematical science degree, bachelor of laws degree, master's degree in computer science, and the PhD degree in computer vision from the University of Adelaide in 1991, 1993, 1994, and 2000, respectively. He is the founding director of the Australian Centre for Visual Technologies (ACVT).