Robust Weighted Low-Rank Matrix Approximation

Anders Eriksson and Anton van den Hengel
School of Computer Science
University of Adelaide, Australia
{anders.eriksson, anton.vandenhengel}@adelaide.edu.au

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 generalisation 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 that can be efficiently implemented using existing optimization software. We show the results of experiments on synthetic and real data.

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

$$\min_{U,V} \|\hat{W} \odot (Y - UV)\|_1,$$

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 $\hat{w}_{ij} = 1$ 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,

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

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 non-smooth problem, so many of the standard optimization tools available will not be applicable. The second is that it is a non-convex problem so certificates of global optimality are in general hard to provide. And finally the optimisation process can also be very computationally demanding task when applied to a 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 minimises the $L_1$ norm, by effectively addressing the issues of non-smoothness 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].

2. Previous Work

The subject of matrix approximation has been extensively studied, especially using $L_2$ type norms. A number of different names for the process are used in the literature, such as 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 of the existing methods for least $L_2$-norm factorization see the work of [7]. This
paper also contains a direct quantitative comparison of a number of key methods. Unfortunately, their comparison did 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 matrix factorization has not received as much attention within computer vision as it has in other areas (see [9, 10] for example). This is beginning to be addressed, however. A very good starting point towards a study of more robust methods, however, is the work of [11]. One of the first methods suggested was Iteratively Re-weighted Least Squares which minimizes a weighted $L_2$ norm. The method is unfortunately very sensitive to initialization (see [12] for more detail).

Black and Jepson in [13] 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 in [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 in [14] 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 in [12] 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 [15] also deals with robust low-rank matrix approximations. This excellent work proves that, under certain assumptions in regards 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 by [16] also needs mentioning. Here they apply Branch and Bound and convex under-estimators 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 $V$,

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

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

$$v^*(U) = (G(U)^T G(U))^{-1} G(U)Wy,$$

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

$$\min_u ||Wy - W(V^T \otimes I_m)u||_2^2,$$

with the optimal $u$ given by

$$u^*(V) = (F(V)^T F(V))^{-1} F(V)Wy,$$

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 [12] were motivated by this method.
Continuing with the Wiberg approach, by substituting (4) into equation (5) we see that the optimum of (5) is also the optimum of
\[
\min_U \|Wy - W\text{vec}(UV^*(U))\|_2^2 = \min_U \|Wy - \phi(U)\|_2^2,
\] (7)
where the function \(\phi(U) = W\text{vec}(UV^*(U))\) is introduced to emphasise the fact that (7) represents a non-linear least squares problem in \(U\). It is the application of the Gauss-Newton method [17] 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 \(\phi\) at \(U_k\) and solving the resulting subproblem
\[
\min_\delta \|f(U_k) - \frac{\partial \phi(U_k)}{\partial U}\delta\|_2^2.
\] (8)
If we let \(J_k\) denote the Jacobian \(\frac{\partial \phi(U_k)}{\partial U}\) we can write the solution to (8) as
\[
\delta_k^* = (J_k^T J_k)^{-1} J_k^T Wy,
\] (9)
the well known normal equation. The next iterate is then given by \(U_{k+1} = U_k + \delta_k^*\).

3. Differentiating the \(L_1\) solution of an overdetermined system of linear equations.

In the previous work [18] the derivation of a \(L_1\) 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
\[
\min_{x \in \mathbb{R}^n} \|b - Ax\|_1,
\] (10)
\(b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}, m \geq n\). It can furthermore be assumed, for our purposes and without loss of generality, that \(A\) has full rank.

Note that (10) is equivalent to the linear program
\[
\begin{align*}
\min_t & \quad t^T t \\
\text{s.t.} & \quad -t \leq b - Ax \leq t \\
& \quad x \in \mathbb{R}^n, t \in \mathbb{R}^m.
\end{align*}
\] (11a)
(11b)
(11c)

Now from [19], we have the following theorem and corollary.

**Theorem 3.1.** Let \(X^*\) denote the set of all minimizers of the convex optimization problem
\[
\min_{x \in \mathbb{R}^n} \|b - Ax\|_1.
\] (12)
Then there always exist 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 (12).

For proofs see section 6 of [19]. Here, \(Z(x^*)\) denotes the set of indices for which the residual is zero, \(Z(x^*) = \{i = 1, ..., m \mid [b - Ax^*]_i = 0\}\). 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 (10), or equivalently (11), 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 \(\frac{\partial x^*}{\partial A_{ij}}\) and \(\frac{\partial x^*}{\partial b_i}\).

**Theorem 3.2.** Let \(x^*\) be the minimizer (10). 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 partitions such that
\[
A = \begin{bmatrix} B \\ N \end{bmatrix},
\] (13)
\[
b = \begin{bmatrix} b_B \\ b_N \end{bmatrix}.
\] (14)

Then \(x^*\) is differentiable at \(A, b\) with the partial derivatives given by
\[
\begin{align*}
\frac{\partial x^*}{\partial B} &= -(x^*)^T \otimes B^{-1} \\
\frac{\partial x^*}{\partial N} &= 0 \\
\frac{\partial x^*}{\partial b_B} &= B^{-1} \\
\frac{\partial x^*}{\partial b_N} &= 0.
\end{align*}
\] (15a)
(15b)
(15c)
(15d)

**Proof.** According to theorem 3.1 we have that
\[
b_B - Bx^* = 0,
\] (16)
by corollary 3.2, \(B\) is of full rank, so
\[
x^* = B^{-1}b_B.
\] (17)
Since as $B$ 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 (17) gives (15a)
\[
\frac{\partial x^*}{\partial B} = \frac{\partial}{\partial B} (B^{-1}b_B) = (b_B^T \otimes I_m) \frac{\partial B^{-1}}{\partial B} = (18)
\]

\[
= -(b_B^T \otimes I_m) (B^{-T} \otimes B^{-1}) = -(x^*)^T \otimes B^{-1}. \tag{19}
\]

Equations (15b)-(15d) follows trivially.

\section{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 iterate using the minimizer of said subproblem.

Our starting point for the derivation our generalization of the Wiberg algorithm is the minimization problem
\[
\min_{U,V} f'(U,V) = ||g'(U,V)||_1 = ||\bar{W} \otimes (Y - UV)||_1. \tag{20}
\]

Following the approach of section 2.1 we first note that for fixed $U$ and $V$ it is possible to rewrite the optimization problem (20) as
\[
v^*(U) = \arg\min_v ||Wy - W(I_n \otimes U)v||_1, \tag{21}
\]
\[
u^*(V) = \arg\min_u ||Wy - W(V^T \otimes I_m)u||_1, \tag{22}
\]

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

Substituting (21) into equation (22) we obtain
\[
U^* = \arg\min_U f(U) = \arg\min_U f'(U,V^*(U)) = \arg\min_U ||g'(U,V^*(U))||_1 = \arg\min_U ||g(U)||_1 = \arg\min_U ||Wy - W\text{vec}(UV^*(U))||_1 = \arg\min_U ||Wy - \phi_1(U)||_1, \tag{23}
\]

\[
= \arg\min_U ||Wy - \phi_1(U)||_1, \tag{24}
\]

where $\phi_1(U) = W\text{vec}(UV^*(U))$. Unfortunately, (26) is not a least squares minimization problem so the Gauss-Newton algorithm is not applicable. Nor does $V^*$ 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 (21) 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 non-linear 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
\[
v^* = B^{-1}QWy = (QG(U))^{-1}QWy. \tag{27}
\]

Using (15a) and applying the chain-rule, we obtain
\[
\frac{\partial v^*}{\partial U} = \frac{\partial}{\partial U} (B^{-1}QWy) = ((v^*)^T \otimes B^{-1}) \frac{\partial B}{\partial U} \tag{28}
\]
\[
\frac{\partial B}{\partial U} = \frac{\partial}{\partial U} (QG(U)) = \frac{\partial}{\partial U} (QW(I_n \otimes U)) = \frac{\partial I_n \otimes QW}{\partial U} \tag{29}
\]
\[
= (I_{rn} \otimes QWy) \frac{\partial (I_n \otimes U)}{\partial U} \tag{30}
\]

Here $T_{m,n}$ denotes the $mn \times mn$ matrix for which $T_{m,n} \text{vec}(A) = \text{vec}(A^T)$.

Combining the above expressions we arrive at
\[
J(U) = \frac{\partial \phi_1(U)}{\partial U} = F(V) + G(U) \frac{\partial v^*}{\partial U} = \tag{31}
\]
\[
= \frac{F(V) + G(U) [(v^*)^T \otimes B^{-1}QWy] \frac{\partial (I_n \otimes U)}{\partial U}. \tag{32}
\]

The Gauss-Newton method, in the least squares case, works by linearizing the non-linear part and solving the resultant subproblem. By equation (31) the same can be done for $\phi_1(U)$.

Using (31) we see that the first order Taylor expansion of $Wy - W\text{vec}(UV^*(U))$ (from equation (26)) around $U_k$ is
\[
q_k(\delta) = ||f(U_k) - J(U_k)\delta||_1. \tag{33}
\]

This allows the construction of an approximation to equation (26)
\[
\min_\delta ||f(U_k) - J(U_k)(\delta - u)||_1 \tag{34}
\]

and as in the $L_2$ case this is a linear problem, but now in $\delta$. This minimisation problem may be stated as
\[
\begin{align*}
&\min_{\delta,t} \left[ \begin{array}{c} 0 \\ 1^T \end{array} \right] \left[ \begin{array}{c} \delta \\ t \end{array} \right] = \left[ \begin{array}{c} 0 \\ 1^T \end{array} \right] \left[ \begin{array}{c} \delta \\ t \end{array} \right] \leq \left[ \begin{array}{c} -f(U_k) \\ f(U_k) \end{array} \right] \\
&\text{s.t.} \quad \left[ \begin{array}{cc} -J(U_k) & -I \\ J(U_k) & -I \end{array} \right] \left[ \begin{array}{c} \delta \\ t \end{array} \right] \leq \left[ \begin{array}{c} -f(U_k) \\ f(U_k) \end{array} \right] \\
&\quad \quad \quad ||\delta||_1 \leq \mu_k \\
&\quad \quad \quad \delta \in \mathbb{R}^{mr}, \ t \in \mathbb{R}^{mn}.\tag{35}
\end{align*}
\]

Let $\delta_k^*$ be the minimizer of (35), then the update rule for our proposed method is simply
\[
U_{k+1} = U_k + \delta_k^*. \tag{36}
\]

Note that in (35) we have added the constraint $||\delta||_1 \leq \mu_k$. This is done as a trust region strategy to limit the step sizes.
that can be taken at each iteration to ensure a non-increasing 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.

1: Input: 
   $U_0 \in \mathbb{R}^{m \times r}$, $\mu_0 > 0$, $1 > \eta_2 > \eta_1 > 0$ and $c > 1$
2: $k = 0$.
3: $V_0 = V^*(U_0)$
4: repeat
5:   Compute the Jacobian $\nabla \phi_1 = J(U_k)$ using (31)
6:   Solve the subproblem (35) to obtain $\delta_k^*$
7:   Let $\rho_k = \frac{f(U_k) - f(U_k + \delta_k^*)}{f(U_k) - q(\delta_k^*)}$
8:   if $\rho_k \leq \eta_1$ then
9:     $\mu_{k+1} = \eta_1 ||\delta_k^*||_1$
10: end if
11: if $\rho_k \geq \eta_2$ then
12:   $\mu_{k+1} = c\mu_k$
13: end if
14: if $\rho_k \geq c$ then
15:   $U_{k+1} = U_k + \delta_k^*$
16:   $V_{k+1} = V^*(U_k + \delta_k^*)$
17: $k = k + 1$
18: end if
19: until convergence

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

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. 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 [12], (alternated LP and alternated QP).

All algorithms were implemented in Matlab. Linear and quadratic optimization subproblems were solved using linprog and quadprog respectively.

5.1. 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 were generated. The elements of the measurement matrix $Y$ were drawn from a uniform distribution between $[-1, 1]$. Then $20\%$ of the elements were chosen at random and designated as missing by setting the corresponding entry in the matrix $W$ to 0. In addition, to simulate outliers, uniformly distributed noise over $[-5, 5]$ were added to $10\%$ of the elements in $Y$.

Since the alternated QP method of Ke and Kanade relies on quadratic programing and as such does not scale as well as linear programs we deliberately kept the synthetic problems relatively small, with $m = 7$, $n = 12$ and $r = 3$.

Figure 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 non-increasing 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. Owing 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 from both the AQP and $L_1$-Wiberg algorithms, applied to one of the synthetic problems, can be seen in figure 2. These figures are not intended to show the quality of the final solution,
Figure 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 but rather how it quickly it is obtained by the competing methods.

Figure ?? 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 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 I 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 alternated linear program method.

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

5.2. 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 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% 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 figure 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 figure 5.

5.3. 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 20 gray scale images randomly selected from
Table 1. The averaged results from running 100 synthetic experiments.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error ($L_1$)</td>
<td>4.60</td>
<td>2.29</td>
<td>1.01</td>
</tr>
<tr>
<td>Execution Time (sec)</td>
<td>0.16</td>
<td>93.57</td>
<td>1.51</td>
</tr>
<tr>
<td># Iterations</td>
<td>4.72</td>
<td>177.64*</td>
<td>21.77</td>
</tr>
<tr>
<td># LP/QP solved</td>
<td>9.44</td>
<td>355.28*</td>
<td>24.13</td>
</tr>
<tr>
<td>Time per LP/QP</td>
<td>0.016</td>
<td>0.264*</td>
<td>0.061</td>
</tr>
</tbody>
</table>

* The alternated QP algorithm was terminated after 200 iterations and 400 solved quadratic programs.

5.4. Non-Rigid Motion Recovery

It was shown in [22] that the recovery of the shape of a non-rigidly moving object from an image sequence can be posed as a low-rank matrix approximation problem. Under the assumption that the non-rigid motion can be described as the linear combination of $K$ different shape basis, the tracked points in the sequence will span a $3K$-dimensional subspace.

In this section we use the shark sequence of [22] for experimental validation. This synthetic data set consists of 91 points on a non-rigid shape (a shark) tracked along sequence of 240 frames, under varying camera pose.

In order to obtain more realistic conditions Gaussian noise with a variance of 1 pixel was added. In addition, 20% of the tracked points were labeled missing and 10% 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.

6. 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.

### References


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

Figure 6. Outliers, images not containing faces.

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

Figure 8. $L_2$ norm of the reconstruction error for the 20 face images.


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


