Robust L_1 Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming

Qifa Ke and Takeo Kanade
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213
{qifa.ke, tk}@cs.cmu.edu

Abstract

Matrix factorization has many applications in computer vision. Singular Value Decomposition (SVD) is the standard algorithm for factorization. When there are outliers and missing data, which often happen in real measurements, SVD is no longer applicable. For robustness Iteratively Re-weighted Least Squares (IRLS) is often used for factorization by assigning a weight to each element in the measurements. Because it uses L_2 norm, good initialization in IRLS is critical for success, but is non-trivial. In this paper, we formulate matrix factorization as a L_1 norm minimization problem that is solved efficiently by alternative convex programming. Our formulation 1) is robust without requiring initial weighting, 2) handles missing data straightforwardly, and 3) provides a framework in which constraints and prior knowledge (if available) can be conveniently incorporated. In the experiments we apply our approach to factorization-based structure from motion. It is shown that our approach achieves better results than other approaches (including IRLS) on both synthetic and real data.

1 Introduction

In many vision or other sensor data interpretation problems, measurements or observation data often lie in a lower dimensional subspace within the original high dimensional data space. Such a subspace, especially the linear subspace, has many important applications in computer vision, such as Structure from Motion (SFM) [24], motion estimation [11], layer extraction [14], object recognition [25], and object tracking [4].

Representing each measurement as a d-vector $\mathbf{m}_i$, the $d \times n$ measurement matrix $\mathtt{M}$ is the stack of all such column vectors: $\mathtt{M} = [m_{ij}] = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n]$. The subspace can then be estimated by the following *matrix factorization*:

$$\mathtt{M}_{d\times n} = \mathtt{U}_{d\times k}\mathtt{V}^{\top}_{k\times n} \quad (1)$$

Here d is the dimension of the input data space; n is the number of input data items; and k is the dimension of the linear subspace. The k columns of the matrix $\mathtt{U}$ are the basis of the linear subspace. In the following of this paper, the terms *matrix factorization* and *subspace estimation* are interchangeably used.

With real measurement data that contains noise, the factorization in Eq. (1) can only be approximated. Assuming Gaussian noise in the measurement, the maximum likelihood estimation (MLE) of the subspace ($\mathtt{U}$ and $\mathtt{V}$) is equivalent to minimizing the following L_2 norm cost function:

$$E(\mathtt{U},\mathtt{V}) = \|\mathtt{M} - \mathtt{U}\mathtt{V}^{\top}\|^2_{L_2} = \sum_{i=1}^{d}\sum_{j=1}^{n}(m_{ij} - \mathbf{u}_i^{\top}\mathbf{v}_j)^2 \quad (2)$$

where $\mathbf{u}_i^{\top}$ is the i-th row of $\mathtt{U}$, and $\mathbf{v}_j$ is the j-th column of $\mathtt{V}^{\top}$. For a matrix $\mathtt{A}$, $\|\mathtt{A}\|^2_{L_2} = \sum_i\sum_j a_{ij}^2$. Note that both $\mathbf{u}_i$ and $\mathbf{v}_j$ are k-vectors. The global minimum of Eq. (2) is provided by the Singular Value Decomposition (SVD) algorithm [8]. However, it is known that SVD is sensitive to outliers, and can not handle missing data. Various approaches have been proposed to deal with the above two problems, to cite a few [24, 21, 13, 18, 12, 5, 7, 1, 10]; See [7] for an extensive review. The most often used approach is to replace the squared error function in Eq. (2) with some robust ρ-function such as Geman-McClure function [3], which leads to a weighted L_2 norm cost function where the contribution of each item is weighted according to its fitness to the subspace:

$$E(\mathtt{W},\mathtt{U},\mathtt{V}) = \|\mathtt{W}\otimes(\mathtt{M}-\mathtt{U}\mathtt{V}^{\top})\|^2_{L_2} = \sum_{i=1}^{d}\sum_{j=1}^{n} w_{ij}(m_{ij}-\mathbf{u}_i^{\top}\mathbf{v}_j)^2$$

where $\otimes$ denotes the component-wise multiplication, and $w_{ij} \geq 0$ is the weight ($w_{ij} = 0$ if m_{ij} is missing).

$E(\mathtt{W},\mathtt{U},\mathtt{V})$ is a convex function *only if* $\mathrm{rank}(\mathtt{W}) = 1$, which is the case studied in [12]. In general, we have $\mathrm{rank}(\mathtt{W}) > 1$, and the above cost function $E(\mathtt{W},\mathtt{U},\mathtt{V})$ becomes non-convex [22]. To find an approximated optimal solution, Iteratively Re-weighted Least Squares (IRLS) is often used. Starting from some *initialization* of $(\mathtt{W},\mathtt{U},\mathtt{V})$, IRLS alternatively fixes two unknowns and computes the third one until *converges*. At each iteration, $\mathtt{U}$ or $\mathtt{V}$ is derived by solving a weighted least squares problem respectively, and $\mathtt{W}$ is computed based on the current estimate of $(\mathtt{U},\mathtt{V})$ using the weighting function derived from the robust M-estimator [3]. It is obvious that the weight matrix $\mathtt{W}$ depends on $(\mathtt{U},\mathtt{V})$ which are unknown at the beginning. Trivial and usually adopted initialization of $\mathtt{W}$ is to assign $w_{ij} = 1$ for

all (i, j) except those for missing data. But it often leads to undesired local solutions when there are many outliers. The reason is that IRLS uses L_2 norm metric. While L_2 norm leads to a simple least squares problem for each alternative step in IRLS, it is, however, sensitive to outliers. It is therefore important for IRLS to start with good initial weights that down-weight the outliers at the beginning, which is of course non-trivial since that is indeed the goal.

Croux et al. [6] robustified the factorization by replacing the L_2 norm with L_1 norm. However, their minimization requires the weight w_{ij} to be separable: $w_{ij} = w_i w_j$. In other words, the weight matrix $\mathtt{W} = \mathbf{w}\mathbf{w}^\top$, which is a rank-1 matrix. This formulation can not be used in measurements where missing data and outliers are presented at arbitrary locations in the measurement matrix. If m_{ij} is missing, for example, w_{ij} must be 0 for which w_i or w_j must be 0. If w_i is set to 0, then w_{ij}'s for all j become 0 even their corresponding measurements are inliers.

In this paper, we formulate the robust L_1 norm matrix factorization as alternative convex programs that can be efficiently solved by linear or quadratic programming. Our formulation 1) can handle outliers and missing data that present at arbitrary locations in the matrix, 2) does not require initial weighting as in IRLS, and 3) provides a framework where constraints and prior knowledge (if available) can be conveniently incorporated. In the experiments we apply our approach to factorization-based structure from motion. It is shown that our approach achieves better results than other approaches (including IRLS) on both synthetic and real data.

2 L_1-norm based subspace estimation

In this section, we formulate the subspace estimation as a L_1 norm minimization problem, and then present alternative convex programming to minimizing the L_1 norm.

2.1 MLE via L_1 norm minimization

The subspace estimation problem can be formulated as a maximum likelihood estimation (MLE) problem [23]. In general, the observed datum $\mathbf{m}_j$, a d-dimensional column vector, is the measurement of some unknown variable $\boldsymbol{\mu}_j$:

$$\mathbf{m}_j = \boldsymbol{\mu}_j + \boldsymbol{\varepsilon}_j \qquad j = 1, \cdots, n \tag{3}$$

where $\boldsymbol{\varepsilon}_j$ is the noise in the measurement. $\boldsymbol{\mu}_j$ resides in a k dimensional linear subspace ($k < d$) such that:

$$\boldsymbol{\mu}_j = \mathtt{U}\mathbf{v}_j \tag{4}$$

where $\mathbf{v}_j$ is the projection of $\mathbf{m}_j$ on the subspace defined by the k columns of $\mathtt{U}$.

Assuming the n measurements $\mathbf{m}_{1:n}$ are independent, the log likelihood of the total n measurements is:

$$l(\boldsymbol{\mu}_{1:n}; \mathbf{m}_{1:n}) = \log p(\mathbf{m}_{1:n} \,|\, \boldsymbol{\mu}_{1:n}) = \sum_{j=1}^{n} \log p(\mathbf{m}_j \,|\, \boldsymbol{\mu}_j)$$

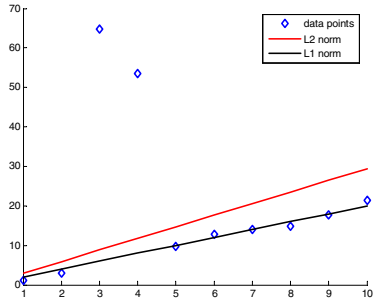


Figure 1. *Fit a line to 10 given data points. The two data points on upper-left are outliers.*

The goal of subspace estimation is thus to find the values of $\boldsymbol{\mu}_j$'s that maximize the likelihood of the measurements $l(\boldsymbol{\mu}; \mathbf{m})$, subject to the condition that these $\boldsymbol{\mu}_j$'s reside in a low dimensional subspace defined by $\mathtt{U}$ in Eq. (4).

If we model the noises $\boldsymbol{\varepsilon}_{1:n}$ by $i.i.d.$ Laplacian distribution, and also assume that the components in the noise vector $\boldsymbol{\varepsilon}_j$ are independent, we have:

$$p(\mathbf{m}_{1:n} \,|\, \boldsymbol{\mu}_{1:n}) \sim \exp\{-\sum_{j=1}^{n} \|\mathbf{m}_j - \boldsymbol{\mu}_j\|_1 / s\} \tag{5}$$

where $\|\mathbf{x}\|_1$ is the L_1 norm of vector $\mathbf{x}$: $\|\mathbf{x}\|_1 = \sum_i |x_i|$, and s is the scale parameter. The maximum log likelihood of the observed data is therefore given by minimizing the following L_1 norm cost function:

$$E_L(\boldsymbol{\mu}) = \sum_{j=1}^{n} \|\mathbf{m}_j - \boldsymbol{\mu}_j\|_1 \tag{6}$$

Substituting Eq. (4) into Eq. (6) and re-written in matrix format:

$$E_L(\mathtt{U}, \mathtt{V}) = \|\mathtt{M} - \mathtt{U}\mathtt{V}^\top\|_{L_1} \tag{7}$$

where $\mathtt{M}$ is the measurement matrix with $\mathbf{m}_j$ its j-th column. For a matrix $\mathtt{A}$, $\|\mathtt{A}\|_{L_1} = \sum_i \sum_j |a_{ij}|$.

It is well known that L_1 norm is much more robust to outliers than L_2 norm. Figure 1 shows a simple example of computing the 1D subspace (the straight line) from ten 2D input data points, two of which are outliers. While L_2 norm gives erroneous line fitting, L_1 norm gives correct result.

2.2 Alternative convex minimization

The L_1-norm cost function defined in Eq. (7) is in general non-convex. But if one of the unknowns, either $\mathtt{U}$ or $\mathtt{V}$, is known, the cost function w.r.t. the other unknown becomes a convex function (see following section) and the global solution to Eq.(7) can be found. This fact suggests a scheme that minimizes the cost function alternatively over $\mathtt{U}$ or $\mathtt{V}$, each time optimizing one argument while keeping the other one fixed. The alternative optimization can be written

as:

$$\mathtt{V}^{(t)} = \arg\min_{\mathtt{V}} \|\mathtt{M} - \mathtt{U}^{(t-1)}\mathtt{V}^\top\|_{L_1} \quad (8a)$$

$$\mathtt{U}^{(t)} = \arg\min_{\mathtt{U}} \|\mathtt{M} - \mathtt{U}\mathtt{V}^{(t)^\top}\|_{L_1} \quad (8b)$$

Here (t) indicates the solution at time t during the iteration. In the following, we show how to solve Eq. (8a) using convex programming, including linear programming and quadratic programming. Eq. (8b) can be solved in the same way.

2.2.1 Linear programming

The cost function of Eq. (8a) can be written as:

$$E(\mathtt{V}) = \|\mathtt{M} - \mathtt{U}^{(t-1)}\mathtt{V}^\top\|_{L_1} = \sum_{j=1}^{n} \|\mathbf{m}_j - \mathtt{U}^{(t-1)}\mathbf{v}_j\|_1 \quad (9)$$

where $\mathbf{m}_j$ is the j-th column of $\mathtt{M}$, $\mathbf{v}_j$ is the j-th column of $\mathtt{V}^\top$. The problem of Eq. (8a) is therefore decomposed into n *independent small* sub-problems, each one optimizing $\mathbf{v}_j$:

$$\mathbf{v}_j = \arg\min_{\mathbf{x}} \|\mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j\|_1 \quad (10)$$

The *global* optimal solution of Eq. (10) can be found by the following linear program (LP):

$$\min_{\mathbf{x},\mathbf{t}} \mathbf{1}^\top\mathbf{t}$$
$$s.t. -\mathbf{t} \le \mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j \le \mathbf{t} \quad (11)$$

where $\mathbf{1}$ is a column vector of ones. In other words, the LP finds the minimum bound $\mathbf{t}$, such that the feasible region defined by $-\mathbf{t} \le \mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j \le \mathbf{t}$ is non-empty.

The computational cost of linear programming depends on the number of unknown variables and linear constraints. In recent years, efficient algorithms have been developed for large scale linear programs. In [2], linear programming has been used in minimizing point-to-line absolute distance (also a L_1 metric) to compute 2D image motions in *real time*. In practice, the complexity of linear programming is known to be quasi-linear in terms of number of constraints. Note that in our case, each of the n sub-problems in Eq. (11) is a small scale linear program. The n sub-problems share the same matrix $\mathtt{U}^{(t-1)}$. The computational cost can therefore be further reduced by sharing intermediate results among the n linear programs.

2.2.2 Quadratic programming

The L_1 norm minimization can also be solved by quadratic programming. To do this, the following Huber M-estimator cost function is used to approximate the L_1 norm [16]:

$$\rho(e) = \begin{cases} \frac{1}{2}e^2, & \text{if } |e| \le \gamma \\ \gamma|e| - \frac{1}{2}\gamma^2, & \text{if } |e| > \gamma \end{cases}$$

where γ is some positive number. Huber M-estimator has similar robustness to the L_1 norm, i.e., it deemphasizes outliers as the L_1 norm does. Each of the n independent sub-problem (Eq. (10)) now becomes:

$$\mathbf{v}_j = \arg\min_{\mathbf{x}} \rho(\mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j) \quad (12)$$

Since Huber M-estimator is a differentiable convex function, Eq. (12) can be converted to a convex quadratic programming (QP) problem whose *global* minimum can be computed efficiently [17]:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{t}} \frac{1}{2}\|\mathbf{z}\|_2^2 + \gamma\mathbf{1}^\top\mathbf{t}$$
$$s.t. -\mathbf{t} \le \mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j - \mathbf{z} \le \mathbf{t} \quad (13)$$

When $\gamma \to 0^+$, the L_1 norm estimator is achieved. Efficient algorithm for Eq. (13) has been developed for large scale quadratic programming using CPLEX software package [17].

2.3 Handling missing data

It is straightforward to handle missing data in our alternative convex programming formulation. In contrast to other approaches (e.g., [24, 20]) that handle missing data by explicitly recovering them, our approach just simply drops the constraint for each missing datum when solving Equation (11) or (13). Once the subspace has been derived, the missing data can be recovered by $\mathtt{U}\mathtt{V}^\top$.

To see the reason, we rewrite Eq. (9) as:

$$E(\mathtt{V}) = \sum_{i=1}^{d}\sum_{j=1}^{n} |m_{ij} - \mathbf{u}_i^\top\mathbf{v}_j| \quad (14)$$

If m_{ij} is missing, then we discard the corresponding cumulative item of $|m_{ij} - \mathbf{u}_i^\top\mathbf{v}_j|$. For the convex programming algorithm, discarding one such item removes one constraint in Eq (11) or Eq (13). In general, the original dimensions d is much larger than the subspace dimension k, which allows large number of missing data in each column of $\mathtt{M}$.

2.4 Algorithm summary

In Fig. 2 we summarize the algorithm for subspace estimation by minimizing the L_1 norm using alternative convex programming. The normalization step is for numerical stability purpose. Once the basis of the subspace $\mathtt{U}$ is obtained, we can use QR-factorization to orthonormalize the basis if desired by the application. In the following we give more details on the initialization step, and on the convergence of the algorithm.

2.4.1 Initialization

Like other iterative algorithms, our algorithm requires the initialization of $\mathtt{U}$ at the beginning. We use a simple random initialization. In practice we find our algorithm is not sensitive to the initialization. Another initialization approach is to first fill each missing datum with its corresponding column-mean, and then apply the SVD algorithm onto the

1. *Initialization:* $\mathtt{U}^{(0)}, \Sigma^0 = \mathtt{I}$.
2. For $t = 1, \cdots,$ *convergence*:

   - $\mathtt{V}^{(t)} = \arg\min_{\mathtt{V}} \|\mathtt{M} - \mathtt{U}^{(t-1)}\Sigma^{(t-1)}\mathtt{V}^\top\|_{L_1}$
   - $\mathtt{U}^{(t)} = \arg\min_{\mathtt{U}} \|\mathtt{M} - \mathtt{U}\Sigma^{(t-1)}\mathtt{V}^{(t)^\top}\|_{L_1}$
   - Normalization: $\begin{cases} \mathtt{N}_v &= diag(\mathtt{V}^{(t)^\top}\mathtt{V}^{(t)}) \\ \mathtt{N}_u &= diag(\mathtt{U}^{(t)^\top}\mathtt{U}^{(t)}) \\ \mathtt{V}^{(t)} &\leftarrow \mathtt{V}^{(t)}\mathtt{N}_v^{-1} \\ \mathtt{U}^{(t)} &\leftarrow \mathtt{U}^{(t)}\mathtt{N}_u^{-1} \\ \Sigma^{(t)} &\leftarrow \mathtt{N}_u\Sigma^{(t-1)}\mathtt{N}_v \end{cases}$

3. Output: $\mathtt{U} \leftarrow \mathtt{U}\Sigma^{1/2}, \quad \mathtt{V} \leftarrow \mathtt{V}\Sigma^{1/2}$

Figure 2. *Algorithm: Subspace estimation via L_1 norm minimization using alternative convex programming. Here* I *in the initialization step is the identity matrix.*

filled matrix to initialize U. We found such approach is not necessary better than the simple random initialization, when there are many outliers and missing data.

2.4.2 Convergence

The cost function $E(\mathtt{U}, \mathtt{V})$ decreases at each alternative minimization step. Since the cost function $E(\mathtt{U}, \mathtt{V})$ is lower bounded (≥ 0), the alternative minimization procedure will converge.

The convergence is achieved when the difference of the parameters between adjacent iterations is small enough. More specifically, the algorithm will stop if for each subspace base, the following holds:

$$\theta(\mathbf{u}_i^{(t)}, \mathbf{u}_i^{(t-1)}) < \alpha$$

Here $\theta(\mathbf{a}, \mathbf{b})$ denotes the angle between the two vectors $\mathbf{a}$ and $\mathbf{b}$; $\mathbf{u}_i$ is the i-th column in U or V; and α is a small positive number.

2.5 Weighted L_1 norm

In many applications, L_1 norm is robust enough since the measurements are bounded, which in turn means the measurement errors are bounded too. In other words, the outliers are usually not strong enough to break the robustness of L_1 norm in many vision applications. For example, in structure from motion, the measurements are the locations of feature points which lie inside the boundary of the images. In face recognition using subspace the measurements are the intensity which are bounded by the maximum intensity value (e.g., 255 in CCD sensors).

We can therefore use L_1 norm minimization to derive a good initial subspace estimation. A good initial weight can then be computed for each element in the measurements, which enables us to use weighted L_1 norm to further reduce the influence of the outliers in order to increase the robustness of our subspace estimation. The weighted L_1 norm is:

$$E(\mathtt{U}, \mathtt{V}) = \|\mathtt{W} \otimes (\mathtt{M}_{d\times n} - \mathtt{U}_{d\times k}\mathtt{V}_{k\times n}^\top)\|_{L_1} \tag{15}$$

Here W contains the weight for each element in the matrix $(\mathtt{M} - \mathtt{U}\mathtt{V}^\top)$, and $\otimes$ denotes the component-wise multiplication. In each alternative minimization step, we have:

$$E(\mathtt{V}) = \sum_{j=1}^{n} \|\mathbf{w}_j \otimes (\mathbf{m}_j - \mathtt{U}^{(t-1)}\mathbf{v}_j)\|_1 \tag{16}$$

Here $\mathbf{w}_j$ is the j-th column of the weight matrix W, with each component computed by:

$$w_{ij} = \frac{1}{c}\exp\left(-\frac{(m_{ij} - \mathbf{u}_i^\top\mathbf{v}_j)^2}{2\sigma^2}\right) \tag{17}$$

Here c is a normalization constant such that $\sum_{i,j} w_{ij} = 1$, and σ can be robustly estimated using median absolute deviation (MAD) [19]. If m_{ij} is missing, then $w_{ij} = 0$. W is recomputed at each iteration. Note that we do not have any special requirement on W, therefore we can handle outliers and missing data presenting at arbitrary locations in M.

The problem in Eq. (16) can be decomposed into n independent small sub-problems:

$$\mathbf{v}_j = \arg\min_{\mathbf{x}} \|\mathbf{w} \otimes (\mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j)\|_1 \tag{18}$$

The above problem can be solved by the following linear program:

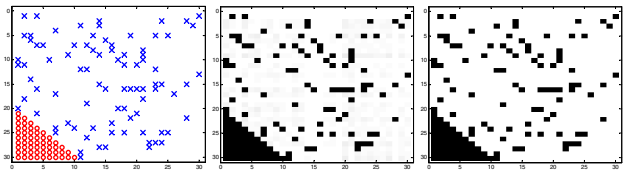
$$\begin{aligned} &\min_{\mathbf{x},\mathbf{t}} \mathbf{w}^\top\mathbf{t} \\ &s.t. -\mathbf{t} \leq \mathtt{U}^{(t-1)}\mathbf{x} - \mathbf{m}_j \leq \mathbf{t} \end{aligned} \tag{19}$$

Compared to equation (11), the only difference is the cost function which is weighted by $\mathbf{w}$. The linear constraints remain the same. Therefore the weighted and un-weighted L_1 norm minimization can be easily integrated together into the algorithm in Fig. 2 with little overhead.

3 Experimental results

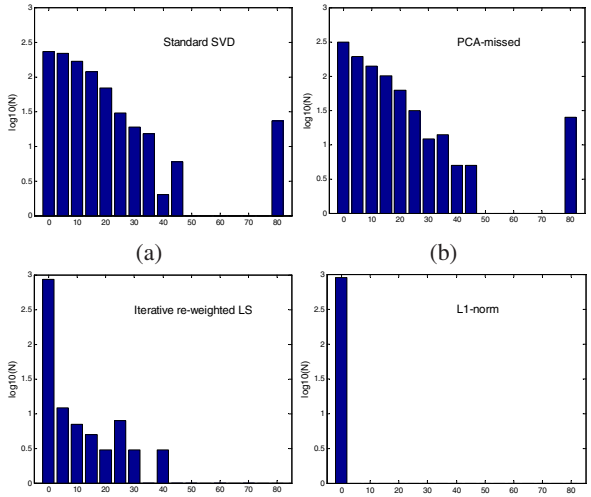
3.1 Synthetic data with ground truth

We use a synthetic 30×30 matrix M (with $\mathrm{rank}(\mathtt{M}) = 3$) to evaluate our algorithm, and to compare with other existing algorithms including: stand SVD, PCA with missing data (PCA-MISS), iteratively re-weighted least squares (IRLS). PCA-MISS assigns weight zero to missing elements and one to others. The IRLS uses the weight function derived from Geman-McClure M-estimator to compute the continuous weights, except for each missing datum the weight is zero. To generate the rank-3 matrix M, we first create a 30×30 matrix A whose elements are randomly drawn from the uniform distribution over $[-100, 100]$. We then apply SVD to A, i.e., $\mathtt{A} = \mathtt{U}\Sigma\mathtt{V}^\top$. The matrix M is then constructed by: $\mathtt{M} = \mathtt{U}_{(:,1:3)}\Sigma_{(1:3,1:3)}\mathtt{V}_{(:,1:3)}^\top$. To simulate the missing data elements, we cut off 55 elements in the



(a) (b) (c)

Figure 3. *Factorizing a synthetic* 30×30 *matrix. (a): Missing data (o) and outliers (×) in synthetic matrix; (b): Initial weights from least* L_1 *norm via alternative convex programming. Darker means lower weights; (c): Final weights after weighted* L_1 *norm minimization.*



(a) (b) (c) (d)

Figure 4. *The histogram of squared errors (in base-10 logarithm) for all elements in* $(\mathtt{M} - \mathtt{U}\mathtt{V}^\top)$*. The last bin is for all errors larger than 80. (a): Standard SVD; (b) PCA with missing data; (c): Iterative re-weighted least squares; (d)* L_1 *norm with convex programming.*

bottom-left corner of matrix. Then we randomly pick up 10% of the elements in $\mathtt{M}$ and transform them into outliers by assigning them a value in the range of $[-2000, 2000]$. Figure 3(a) shows the missing data (o) and outliers (×) in the matrix. The original un-corrupted matrix $\mathtt{A}$ is saved as ground truth for comparison purpose.

We use trivial initialization where $w_{ij} = 0$ for missing data, and $w_{ij} = 1$ for others. We apply our alternative convex programming to minimizing the L_1 norm for rank-3 matrix factorization, from which we re-compute the initial weights for each element (using Eq. (17)), which are shown in Figure 3(b), where darker means lower weights. All of the outliers are correctly assigned near-zero weights. Figure 3(c) shows the final weights after we applied the weighted L_1 norm matrix factorization.

For quantitative comparison, we repeat the same experiment for 20 times, with different matrix $\mathtt{M}$ but same initialization scheme. Our approach converges to the right solution in all of these 20 runs in less than 10 iterations, while the IRLS converges to the right solution only in 7 out of the 20 runs, with an average of about 40 iterations of alternative minimization. We also use reconstruction errors (averaged from the 20 runs) to compare different algorithms. The error matrix is defined as: $\mathtt{E} = \mathtt{M} - \hat{\mathtt{M}}$, where $\mathtt{M}$ is the ground truth and $\hat{\mathtt{M}}$ is its rank-3 approximation $\hat{\mathtt{M}} = \mathtt{U}_{30\times 3}\mathtt{V}_{3\times 30}^\top$. We plot the histogram of the squared errors (elements in $\mathtt{E}$) in Figure 4 (note that the count is in 10-based log scale). As we can see, the reconstruction errors from our algorithm are all near to zero. It finds the true subspace, recovers the true values of the missing data and outliers. PCA-MISS performs marginally better than SVD, but can not recover the correct subspace. IRLS performs better than PCA-MISS, but worse than our approach.

3.2 Application: structure from motion

Structure from motion (SFM) with affine camera model is first formulated by Tomasi and Kanade [24] as a rank-3 matrix factorization problem. In an affine camera, a 3D scene point $\mathbf{X}^j = (X, Y, Z, 1)^\top$ and its projection on the i-th camera imaging plane $\mathbf{x}^j = (x, y)^\top$ is related by a 2×4 affine projection matrix $\mathtt{P}_i$:

$$\mathbf{x}^j = \mathtt{P}_i \mathbf{X}^j$$

Given F images and n scene points, we have:

$$\begin{bmatrix} \mathbf{x}_1^1 & \cdots & \mathbf{x}_1^n \\ \vdots & \ddots & \vdots \\ \mathbf{x}_F^1 & \cdots & \mathbf{x}_F^n \end{bmatrix} = \begin{bmatrix} \mathtt{P}_1 \\ \vdots \\ \mathtt{P}_F \end{bmatrix} \begin{bmatrix} \mathbf{X}^1 & \cdots & \mathbf{X}^n \end{bmatrix}$$
$$\Leftrightarrow \mathtt{M} = \mathtt{P}\mathtt{X} \tag{20}$$

where $\mathtt{P} \in \Re^{2F\times 4}$ and $\mathtt{X} \in \Re^{4\times n}$. When the measurement matrix $\mathtt{M}$ is subtracted by its column-wise mean, the rank of the resulted matrix $\Delta\mathtt{M}$ is reduced to three, and we can obtain the Euclidean camera motion matrix and 3D of the scene points by rank-3 matrix factorization followed by metric enforcement on the camera motion matrix [24]:

$$\Delta\mathtt{M} = \mathtt{R}_{2F\times 3}\mathtt{S}_{3\times n} \tag{21}$$

where $\mathtt{R}$ is the camera motion matrix and $\mathtt{S}$ is the recovered 3D of the scene points.

When there are outliers and missing data, we can not directly compute the column-wise mean of $\mathtt{M}$. Instead, we first use subspace estimation to compute the rank-4 matrix factorization indicated by Eq. (20), which gives us a rank-4 approximation $\hat{\mathtt{M}} = \mathtt{U}_{2F\times 4}\mathtt{V}_{4\times n}^\top$ of the measurement matrix $\mathtt{M}$. We then recovered the motion and 3D from $\hat{\mathtt{M}}$ by subtracting its column-wise mean and then applying Eq. (21).

Even we can correctly estimate the 4-D subspace in Eq. (20), we should be cautious when we try to use Eq. (21) to recover the 3D of some outlier measurement $\mathbf{m}$. The factorization algorithm simply picks the closest point in the subspace to approximate $\mathbf{m}$. In the extreme case, when the

2D feature positions of a scene point are all outliers, such subspace approximation is no long valid, and we simply do not have enough cue to infer its 3D position. But we can use the subspace to detect such *outlier track*. To do so, we compute the re-projection errors of each recovered 3D point across all the frames:

$$e_i^j = \|\mathbf{x}^j - \hat{\mathbf{x}}^j\|_2 = \|\mathbf{x}^j - \mathtt{P}_i\mathbf{X}^j\|_2$$

where e_i^j is the re-projection error of the j-th point on the i-th frame. We assume that the x-component and y-component of a 2D point are independent. e_i^j then follows χ^2 distribution with 2 degrees of freedom. $\mathbf{x}_i^j$ is marked as an outlier if its corresponding e_i^j is outside the 95% confidence interval of the χ^2 distribution. A 3D point is kept only if its corresponding 2D track contains enough inliers. In our experiments, we require a valid 3D point contain at least 5 inliers in its 2D track.

We applied both our approach and IRLS to factorization based SFM for two image sequences from CMU VASC public image database [1]: the *Pingpong ball* sequence and *House* sequence. In order to test the robustness of the algorithms, in the feature tracking phase we intentionally relax the error threshold, which means that some features with larger tracking errors (the intensity residuals) are still kept.

3.2.1 Pingpong ball

We use twenty frames of the Pingpong ball sequence to test our algorithm. The ball underwent a rotational motion while the images were taken. Figure 5(a) and (b) show the first and last frame, with tracked features super-imposed on them as white dots. About 42% of the features are missing in the last frame. Due to specular reflections, some of the feature points become outliers. The arrow in Figure 5(b) shows one of them.

We successfully recovered the 3D of almost all of the feature points appeared in the first frame, except four feature points that are detected and removed as outliers due to specular reflection. Figure 5(c) shows a side view of the 3D points recovered by our approach. The smooth circle-shape contour indicates correct shape recovery. We also applied IRLS to this sequence for comparison purpose. Figure 5(d) plots the mean re-projection errors. As we can see our approach has less re-projection errors. For visualization, Figure 5(e) shows the recovered 3D (texture-mapped) using our approach, and Figure 5(f) shows the same view of texture-mapped 3D recovered by IRLS. Note the jagged surface in (f) due to errors which does not appear in (e).

3.2.2 House

We use 40 frames of the *house* sequence where the house in the image has two visible perpendicular walls. Figure 6(a) and (b) shows the first and last frame in the sequence, with

[1] http://vasc.ri.cmu.edu/idb/

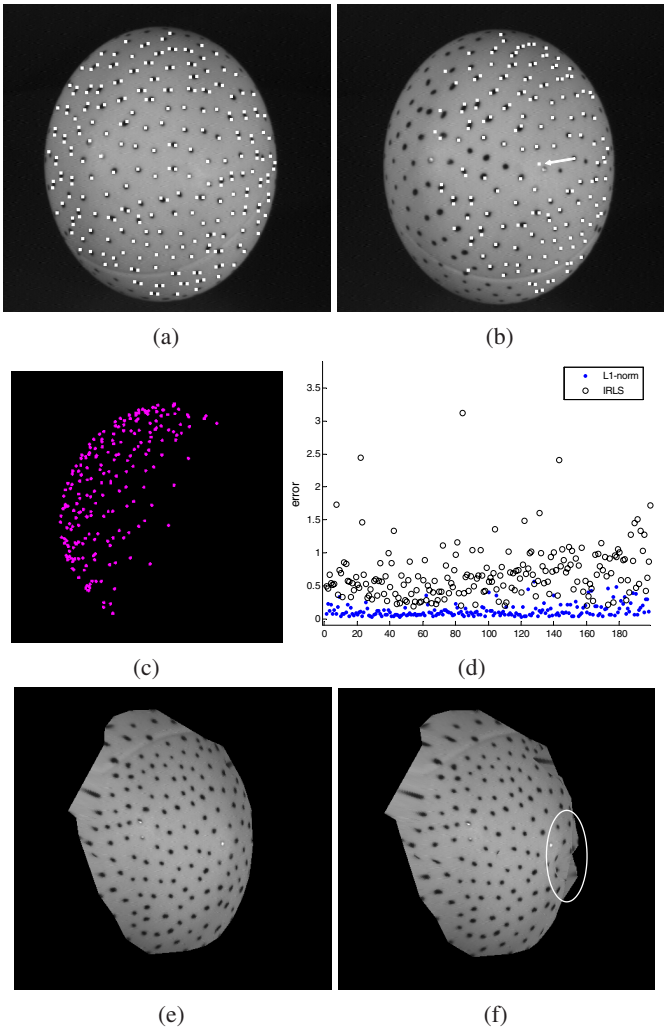
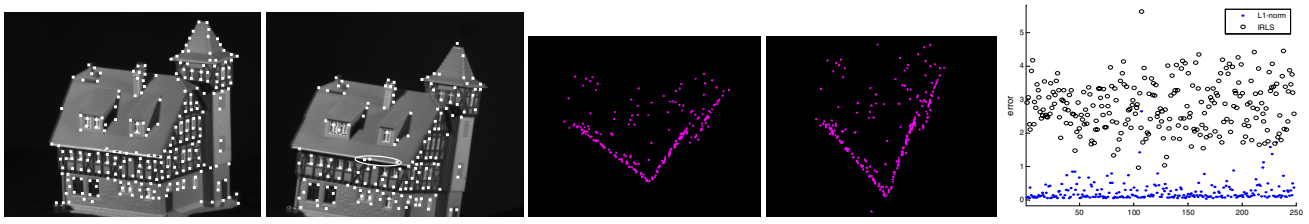


Figure 5. *Structure from motion of the pingpong image sequence. (a): The first frame with all feature points (marked by white dots); (b): The 20th frame where about 42% feature points from the first frame are missed; (c): One view of 3D points recovered by our approach; Note the smooth circle shape boundary; (d): Reprojection errors of all recovered 3D points; (e): Texture-mapped 3D recovered by our approach; (f): 3D recovered by IRLS; Note the jagged surface marked by the eclipse;*

feature points (marked by white dots) superimposed. About 38% percent of the feature points from (a) are missed in (b). The eclipse in (b) shows some points with large tracking errors. Figure 6(c) shows top-down view of the 3D points recovered by our approach. Notice the perpendicular angle between the two walls. Figure 6(d) shows the same view of the 3D points recovered by IRLS, where the two walls are not perpendicular. Figure 6(e) shows the re-projection errors. Our approach contains less errors than that of IRLS.

4 Incorporating prior knowledge

In many applications prior knowledge or constraints about the subspace are available. Our formulation of



(a) (b) (c) (d) (e)

Figure 6. *Structure from motion of the* house *image sequence. (a): The first frame with 248 feature points (marked by white dots); (b): The 40^{th} frame where about 38% of the points from the first frame are missed; (c): Top-down view of 3D points recovered by our approach; Note the perpendicular angle between the two walls; (d): Top-down view of 3D points recovered by IRLS; Note the angle between the two walls is not perpendicular; (e): Re-projection errors of all recovered 3D points. Our approach contains fewer errors than IRLS.*

L_1 norm subspace estimation via alternative convex programming provides a framework where constraints or prior knowledge can be conveniently incorporated.

4.1 Smoothness constraint

Smoothness constraint is often used in motion and shape estimation. For example, in structure from motion via factorization [24], we can enforce the temporal smoothness of camera motion [9] as well as the 3D scene smoothness.

Smoothness is often enforced by penalizing the discontinuities between neighbored elements. Such discontinuity is usually measured by first or second order derivatives. The L_1 norm based cost function with smoothness constraints can be written as:

$$E(\mathtt{U},\mathtt{V}) = \|\mathtt{M}-\mathtt{U}\mathtt{V}^\top\|_{L_1} + \delta\|\mathtt{D}_1\mathtt{U}\|_{L_1} + \eta\|\mathtt{D}_2\mathtt{V}\|_{L_1} \quad (22)$$

Here $\delta \geq 0$ and $\eta \geq 0$ are used to control the smoothness of the solution. $\mathtt{D}_1$ and $\mathtt{D}_2$ are the differentiate matrices. Their values depend on the neighbor relationship among the elements in $\mathtt{M}$. Figure 7 is an example that shows a neighborhood relationship and the corresponding 1st order differentiate matrix $\mathtt{D}$. In the application of structure from motion, the neighborhood relationship can be established by Delaunay triangulation of 2D feature points.

We use alternative minimization to minimize Eq. (22):

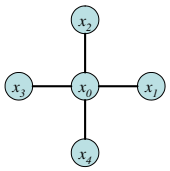
$$\mathtt{V}^{(t)} = \arg\min_{\mathtt{V}} \|\mathtt{M}-\mathtt{U}^{(t-1)}\mathtt{V}^\top\|_{L_1} + \eta\|\mathtt{D}_2\mathtt{V}\|_{L_1} \quad (23a)$$

$$\mathtt{U}^{(t)} = \arg\min_{\mathtt{U}} \|\mathtt{M}-\mathtt{U}\mathtt{V}^{(t)^\top}\|_{L_1} + \delta\|\mathtt{D}_1\mathtt{U}\|_{L_1} \quad (23b)$$

We can decompose the above optimization into smaller independent sub-problems, each sub-problem corresponding to one maximum connected subgraph in the neighborhood graph. We can also solve it as one single problem. For example, Eq. (23a) can be converted to the following linear program:

$$\min_{\{\mathbf{x}_j,\mathbf{t}_j,\mathbf{h}_j\}} \sum_j (\mathbf{1}^\top\mathbf{t}_j + \eta\mathbf{1}^\top\mathbf{h}_j) \quad (24)$$
$$s.t. -\mathbf{t}_j \leq \mathtt{U}^{(t-1)}\mathbf{x}_j - \mathbf{m}_j \leq \mathbf{t}_j$$
$$-\mathbf{h}_j \leq \mathtt{D}\mathbf{x}_j \leq \mathbf{h}_j$$
$$j = 1, ..., n$$

Here $\mathbf{x}_j$ is the j-th column of $\mathtt{V}$. Note that we can enforce different degree of smoothness on different elements by changing $\eta\|\mathtt{D}\mathtt{X}\|_1$ to $\|\mathtt{Y} \otimes (\mathtt{D}\mathtt{X})\|_1$. The matrix $\mathtt{Y}$ specifies the degree of smoothness for each element. We can avoid smoothness constraints at discontinuous locations by assigning their weights to zero in $\mathtt{Y}$. Automatic estimation of $\mathtt{Y}$ is out of the scope of this paper. We can also extend the Eq. (23) to weighted L_1 norm, as done in Section 2.5.



$$\mathtt{D} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

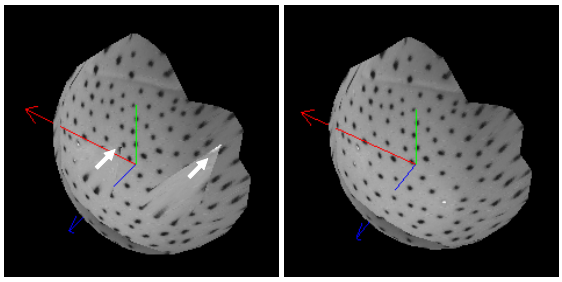
Figure 7. *A 4-connected neighborhood and its corresponding differentiation matrix.* $\|\mathtt{D}\mathtt{X}\|_1 = \Sigma_{i=1}^4|x_0 - x_i|$.

4.2 Non-negative matrix factorization

Non-negative matrix factorization [15] has many applications in computer vision. Adding non-negative constraint is straightforward in our formulation. We can simply add the non-negative constraints $\mathtt{U} \geq 0$ and $\mathtt{V} \geq 0$ to the set of linear constraints in the convex program (e.g., $\mathbf{x} \geq 0$ in Eq. (11),(13), or (24)).

4.3 Experiment: smoothness constraint

When a 2D track contains too many outliers, we would not be able to recover its 3D. As an example, Fig. 8(a) shows the texture-mapped 3D points recovered in Section 3.2.1, including the four outliers that were previously detected by the subspace model. The arrows show the incorrect 3D positions of those outliers. If we can apply prior knowledge, we may still have enough constraints to recover their correct 3D positions. For example, once we know that the surface of the ball is smooth, we can apply the smoothness constraint in the matrix factorization. Fig. 8(b) shows the result after the smoothness constraint is enforced. The position of the four previously detected outliers are now correctly re-



(a) (b)

Figure 8. *Recovering the 3D of some very noisy points using smoothness constraints. (a): Texture-mapped 3D points recovered without enforcing smoothness constraints, including the four very noisy points due to reflection, as indicated by the arrows; (b): Texture-mapped 3D points recovered with smoothness constraints. Note that the 3D of the noisy points are correctly recovered.*

covered. The intuition is that when the re-projection errors are large for some data items, their corresponding weights become very small, and the smoothness penalty plays an important role in determining their 3D positions (in the factorization).

5 Conclusion

We have presented a new subspace estimation algorithm that minimizes the (weighted) L_1 norm based cost function using alternative convex programming. Our algorithm is robust without requiring initial weighting, handles missing data straightforwardly, and provides a framework for incorporating prior knowledge/constraints. Compared to existing approaches that use L_2 norm or weighted L_2 norm (SVD, PCA-MISS, IRLS), our approach achieves better estimations in fewer iterations. One might argue that L_1 norm has lower breakpoint than some other robust estimators (e.g., M-estimators with redescending influence function, see [3]). However, the fact that many applications have bounded measurements makes L_1 norm robust enough to obtain a reasonable solution in practice. Moreover, starting from the solution derived by L_1 norm minimization, we can confidently initialize the weights for each element, which enables us to perform weighted L_1 norm minimization to increase the robustness of the final subspace estimation.

Both the L_1 norm and weighted L_1 norm are minimized by alternative convex programming, including linear programming and quadratic programming. Efficient implementation of linear/quadratic programming in commercial package can achieve quasi-linear complexity. In the future we plan to customize the linear/quadratic programming for our purpose, especially to reduce the computational cost by sharing the information among the decomposed linear/quadratic sub-programs.

Acknowledgements

This research was supported by USAF under grant No. F49620-03-1-0381 managed by Dr. Robert Sierakowski, Chief Scientist AFRL/MN, and Dr. Neal Glassman and Dr. Sharon Heise, Program Directors, AFOSR.

References

[1] H. Aanæs, R. Fisker, K. Åström, and J. M. Carstensen. Robust factorization. *IEEE Trans. PAMI*, 24(9), 2002.

[2] M. Ben-Ezra, S. Peleg, and M. Werman. Real-time motion analysis with linear programming. In *ICCV*, 1999.

[3] M. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–92, 1996.

[4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.

[5] G. Q. Chen. Robust point feature matching in projective space. In *CVPR 2001*, pages 717–722.

[6] C. Croux and P. Filzmoser. Robust factorization of a data matrix. In *COMPSTAT, Proceedings in Computational Statistics*, pages 245–249, 1998.

[7] F. de la Torre and M. Black. A framework for robust subspace learning. *IJCV*, 54(1):117–142.

[8] G. Golub and C. V. Loan. *Mattrix Computation*. Johns Hopkins University Press, 2nd edition, 1989.

[9] A. Gruber and Y. Weiss. Factorization with uncertainty and missing data: Exploiting temporal coherence. In *Advances in Neural Information Processing Systems 16*.

[10] D. Q. Huynh, R. Hartley, and A. Heyden. Outlier correction in image sequences for the affine camera. In *ICCV*, 2003.

[11] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *ICCV*, 1999.

[12] M. Irani and P. Anandan. Factorization with uncertainty. In *ECCV (1)*, pages 539–553, 2000.

[13] D. Jacobs. Linear fitting with missing data. In *CVPR 1997*, pages 206–212.

[14] Q. Ke and T. Kanade. A subspace approach to layer extraction. In *CVPR 2001*, pages I:255–262.

[15] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[16] W. Li and J. Swetits. The linear l_1 estimator and the huber m-estimator. *SIAM J. Optimization*, 8:457–475, 1998.

[17] O. L. Mangasarian and D. R. Musicant. Robust linear and support vector regression. *IEEE Trans. on PAMI*, 22(9):950–955, 2000.

[18] D. D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *ICCV*, pages 696–702, 1998.

[19] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, New York, 1987.

[20] S. Roweis. EM algorithms for pca and spca. In *NIPS*, 1997.

[21] H.-Y. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. on PAMI*, 17(8):854–867.

[22] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proc. of Int. Conf. on Machine Learning*, 2003.

[23] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 6(3):611–622, 1999.

[24] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2), 1992.

[25] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.