General and Nested Wiberg Minimization

Dennis Strelow
Google
Mountain View, CA
strelow@google.com

Abstract

Wiberg matrix factorization breaks a matrix Y into low-rank factors U and V by solving for V in closed form given U, linearizing $V(U)$ about U, and iteratively minimizing $||Y - UV(U)||_2$ with respect to U only. This approach factors the matrix while effectively removing V from the minimization. Recently Eriksson and van den Hengel extended this approach to L_1, minimizing $||Y - UV(U)||_1$. We generalize their approach beyond factorization to minimize an arbitrary function that is nonlinear in each of two sets of variables. We demonstrate the idea with a practical Wiberg algorithm for L_1 bundle adjustment. We also show that one Wiberg minimization can be nested inside another, effectively removing two of three sets of variables from a minimization. We demonstrate this idea with a nested Wiberg algorithm for L_1 projective bundle adjustment, solving for camera matrices, points, and projective depths.

We also revisit L_1 factorization, giving a greatly simplified presentation of Wiberg L_1 factorization, and presenting a successive linear programming factorization algorithm. Successive linear programming outperforms L_1 Wiberg for most large inputs, establishing a new state-of-the-art for for those cases.

1. Introduction

Matrix factorization breaks a matrix Y into low-rank factors U and V by minimizing $||Y - UV||$ with respect to U and V. Wiberg[14] approached the L_2 version of this problem by solving for V given U in closed form, linearizing $V(U)$ about U, and iteratively minimizing $||Y - UV(U)||_2$ with respect to U only, rather than U and V simultaneously. This approach minimizes the original objective function while eliminating V from the minimization, improving convergence[7][8]. Recently Eriksson and van den Hengel[2] showed that the Wiberg approach could be extended to L_1, minimizing $||Y - UV(U)||_1$, using linear programming. They showed that their Wiberg approach outperformed the previous state-of-the-art for L_1 factorization, Ke and Kanade's[6] alternative convex programming method.

In this paper we generalize the Wiberg approach beyond factorization to minimize an arbitrary nonlinear function of two sets of variables, $f(U, V)$. Our general Wiberg minimization can be used for L_1 minimization, L_2 minimization, or maximum likelihood estimation. In this paper we focus on the most complex case, L_1, generalizing Eriksson and van den Hengel's method. We demonstrate the idea with a practical Wiberg algorithm for L_1 bundle adjustment, which we demonstrate on a real image sequence with about 700 images and 10,000 points.

Our general Wiberg minimization works by solving for V iteratively rather than in closed form. Since it is found iteratively, V can itself be split into two sets of variables found using Wiberg minimization. This results in a nested Wiberg minimization that can effectively minimize with respect to three sets of variables. We demonstrate this idea with an L_1 Wiberg algorithm for projective (uncalibrated) bundle adjustment, solving for camera matrices, points, and projective depths.

Our main contributions are general and nested Wiberg minimization; general and nested Wiberg minimization for the hardest case, L_1; and L_1 bundle adjustment and L_1 projective bundle adjustment using both Wiberg and successive linear programming, which minimizes with respect to all of the variables simultaneously. We also include some contributions to L_1 matrix factorization: a greatly simplified presentation of L_1 Wiberg factorization, which should make it more accessible; and a successive linear programming algorithm for L_1 factorization, which is more practical than L_1 Wiberg for large inputs, establishing a new state-of-the-art for for those cases.

Table 1 shows general Wiberg's place in the space of optimization problems.

2. Related work

Wiberg[14] presented an L_2 factorization algorithm for matrices with missing data, which solved for one set of variables V in terms of the other U, linearized V about U, and then minimized with respect to U only. Okatani *et*

	linear in U or V		
	minimize L_2	minimize L_1	MLE
simultaneous	✓	✓	✓
alternating	✓	✓	✓
Wiberg	Wiberg 1976	Eriksson 2010	this work

	nonlinear in both U and V		
	minimize L_2	minimize L_1	MLE
simultaneous	✓	✓	✓
alternating	✓	✓	✓
Wiberg	this work	this work	this work

Table 1. Optimization problems in two sets of variables U, V, and possible approaches for solving them. "✓" indicates standard algorithms like Levenberg-Marquardt, successive linear programming, or expectation-maximization. General Wiberg ("this work") greatly extends the applicability of the Wiberg approach.

al.[7][8] showed that Wiberg factorization converged better than minimizing with respect to U and V simultaneously with Levenberg-Marquardt and other algorithms, and argued that Wiberg's method had been neglected by the computer vision community.

Recently, Eriksson and van den Hengel[2] extended this approach to L_1 matrix factorization using linear programming. Their method outperformed Ke and Kanade's alternating convex programming algorithms[6], establishing a new state-of-the-art for L_1 factorization. Using Eriksson and van den Hengel's development, we generalize their method beyond factorization to minimize functions that are nonlinear in each of two sets of variables. We also compare their factorization against a stronger baseline than they considered, which minimizes with respect to all of the variables simultaneously. This experiment is analogous to Okatani and Deguchi's[7] L_2 experiment with Wiberg and Levenberg-Marquardt.

Wiberg's method was an application of Ruhe and Wedin's[12] more general work on separable nonlinear minimization that solved for a V in terms of U and then minimized with respect to U only. Ruhe and Wedin recognized that this approach would be advantageous when V breaks down into small independent problems given U, which happens in all the problems in this paper. But, their analysis and experiments focused on least squares objective functions linear in V. In even earlier work, Richards[11] described a separable method for maximum likelihood estimation, but similarly demonstrated it only on a least squares problem linear in V. In contrast, we consider more general functions that can be nonlinear in both U and V. For L_1, there is no previous work analogous to Ruhe and Wedin or Richards.

The Wiberg approach contrasts with alternating least squares and similar methods, which alternate between solving for one set of unknowns while holding the other fixed. Alternating methods sometimes converge well, but they can also converge very slowly[7] or fail to converge "catastrophically"[10]. For this reason, we've bypassed alternating methods as baseline algorithms, instead minimizing with respect to all of the unknowns simultaneously. Since we're considering L_1 in this paper, we've used successive linear programming[1], which often converges quadratically.

3. Wiberg L_1 Factorization

In this section we present Eriksson and van den Hengel's Wiberg L_1 factorization (subsections 3.1-3.4), an alternative algorithm that minimizes with respect to all of the unknowns simultaneously using successive linear programming (3.5), and a qualitative analysis of L_1 Wiberg (3.6).

Our presentation of Wiberg L_1 factorization is equivalent to the original but simpler, and should be more accessible. Part of the simplification is an algorithmic change – solving for the columns of V separately rather than solving for V as a whole – which produces the same solution while greatly simplifying the math.

Throughout the paper we'll use derivatives of matrices and derivatives with respect to matrices. Fackler's notes[3] are a good guide to matrix derivatives. Most cases we'll encounter can be handled by flattening the matrices to vectors by column, and then using the normal rules for vector derivatives.

The general and nested Wiberg minimizations in Sections 4 and 5 below rely heavily on the development in this section.

3.1. Linear Programming and Derivatives

Linear programming solves the problem:

$$\min_x c^{\mathrm{T}} x, \text{ s.t. } Ax \leq b, x \geq 0 \tag{1}$$

(1) is the problem's canonical form, which we'll use in Sections 3.2 and 3.3 below. We'll solve this problem using the simplex method.

To find the derivatives of the linear program solution x, we'll also need to understand the slack form used by the simplex method, which converts the inequalities $Ax \leq b$ to equalities by introducing nonnegative slack variables s:

$$\min_x c^{\mathrm{T}} x, \text{ s.t. } [A\ I][x; s] = b, x \geq 0, s \geq 0 \tag{2}$$

The optimal solution $[x; s]$ will include basic components x_B, which can be nonzero; and nonbasic components x_N, which will be zero. The columns of $[A\,I]$ corresponding to x_B are the basis B, and $Bx_B = b$. Then since $x_B = B^{-1}b$, it can be shown that:

$$\frac{dx_B}{dB} = -x_B^T \otimes B^{-1} \tag{3}$$

$$\frac{dx_B}{db} = B^{-1} \tag{4}$$

where $\otimes$ is the Kronecker product. Some simple rearranging (e.g., inserting zero derivatives corresponding to elements of the nonbasic components, dropping derivatives of the slack variables) then converts these derivatives to dx/dA and dx/db.

3.2. Linear L_1 Minimization

We can minimize the L_1 residual of an overdetermined linear system,

$$\min_y ||d - Cy||_1 \tag{5}$$

using linear programming. Since the linear programming problem (1) requires $x \geq 0$, we'll split y into the difference of two nonnegative terms, $y = y^+ - y^-$. Then, let t_i be the L_1 residual of individual row i of $d - Cy$:

$$t_i = |d_i - C_i(y^+ - y^-)| \tag{6}$$

Converting (6) to two inequalities we have:

$$C_i(y^+ - y^-) - t_i \leq d_i \tag{7}$$
$$-C_i(y^+ - y^-) - t_i \leq -d_i \tag{8}$$

Then, the optimal y^+, y^-, t_i can be found with the linear program:

$$\min_{y^+,y^-,t} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y^+ \\ y^- \\ t \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} C & -C & -I \\ -C & C & -I \end{bmatrix} \begin{bmatrix} y^+ \\ y^- \\ t \end{bmatrix} \leq \begin{bmatrix} d \\ -d \end{bmatrix} \tag{10}$$

The objective function in (9) just says to minimize the sum of the individual L_1 errors. We can get the derivative of $[y^+\ y^-\ t]$ with respect to the coefficient matrix and right-hand side of (10) as described in Section 3.1 above. Since y is a simple function of y^+, y^- and the coefficient matrix and right-hand-side are simple functions of C, d, we can then get dy/dC and dy/dd from those derivatives with some simple algebra and rearranging.

3.3. Nonlinear L_1 Minimization

We can minimize the L_1 norm of a nonlinear function using the linear minimization in Section 3.2 iteratively. Suppose we have errors between predictions $f(x)$ and observations y:

$$\text{error}(x) = y - f(x) \tag{11}$$

and we want to minimize:

$$\min_x ||\text{error}(x)||_1 \tag{12}$$

Given an estimate x, compute a new estimate $x + \delta_x$ with:

$$\min_{\delta_x} ||\,\text{error}(x) - \frac{df(x)}{dx}\delta_x||_1 \tag{13}$$

and repeat until convergence. (13) is a linear L_1 minimization and can be solved as described in 3.2. This iteration is successive linear programming[1] applied to L_1 minimization, and will be used by all of the algorithms in this paper.

The step will often increase rather than decrease the objective function, preventing convergence, because the δ_x that minimizes (13) may be outside the region where the linearization $df(x)/dx$ is accurate. An adaptive trust region helps ensure convergence by limiting the step to a finite region near the current estimate, and adaptively determining what the size of that region should be. The trust region's role is similar to the adaptive damping factor λ that Levenberg-Marquardt adds to Gauss-Newton.

To limit the step size $||\delta_x||_1$ in (13) to some trust region size μ, we augment (10) to be:

$$\begin{bmatrix} C & -C & -I \\ -C & C & -I \\ I & I & 0 \end{bmatrix} \begin{bmatrix} y^+ \\ y^- \\ t \end{bmatrix} \leq \begin{bmatrix} d \\ -d \\ \mu \end{bmatrix} \tag{14}$$

We also need to adapt μ to a useful step size around the current estimate. If the most recent δ_x decreases the objective function, we increase μ to 10μ assuming that the best trust region is no smaller than the current μ. If δ_x does not decrease the objective function, then δ_x is outside the region where the linearization is valid, and we decrease μ to $||\delta_x||_1/10$ and recompute.

3.4. Wiberg L_1 Factorization

Suppose we have an observation matrix Y, that observations $Y_{r_1,c_1}, Y_{r_2,c_2}, \ldots, Y_{r_k,c_k}$ are present, and that the other observations are missing. Then, low-rank matrix factorization minimizes:

$$||[Y_{r_1,c_1}\ \ldots\ Y_{r_k,c_k}]^{\mathrm{T}} - [u_{r_1}v_{c_1}\ \ldots\ u_{r_k}v_{c_k}]^{\mathrm{T}}||_1 \tag{15}$$

with respect to low-rank factors U and V, where u_r is row r of U and v_c is column c of V. Eriksson and van den Hengel showed that Wiberg's approach could be used to minimize (15) by combining the linear and nonlinear L_1 minimizations in the previous sections.

First, hold U fixed and solve for each column v_c of V, by minimizing

$$||[Y_{r_{i_1},c}\ \ldots\ Y_{r_{i_n},c}]^{\mathrm{T}} - [u_{r_{i_1}}; \ldots; u_{r_{i_n}}]v_c||_1 \tag{16}$$

with respect to v_c. This is a linear minimization, so v_c and dv_c/dU can be found as described in Section 3.2 above.

Then, rewrite (15) as a function of U only:

$$||[Y_{r_1,c_1}\ \ldots\ Y_{r_k,c_k}]^{\mathrm{T}} - [u_{r_1}v_{c_1}(U)\ \ldots\ u_{r_k}v_{c_k}(U)]^{\mathrm{T}}||_1 \tag{17}$$

and minimize it iteratively with respect to U using (13). To do this, we need the errors $y - f(x)$, which are just the vector in (17), and the derivative of the predictions with respect

to U. In this case, the individual predictions are $u_{r_i} v_{c_i}(U)$ and their derivatives are:

$$\frac{du_{r_i} v_{c_i}(U)}{dU} = \frac{\partial u_{r_i} v_{c_i}(U)}{\partial U} + \frac{\partial u_{r_i} v_{c_i}(U)}{\partial v_{c_i}} \frac{dv_{c_i}}{dU} \tag{18}$$

where

$$\frac{\partial u_{r_i} v_{c_i}(U)}{\partial u_{r_i}} = v_{c_i}^{\mathrm{T}} \tag{19}$$

and the partial derivative with respect to other components of U is zero; and

$$\frac{\partial u_{r_i} v_{c_i}(U)}{\partial v_{c_i}} = u_{r_i} \tag{20}$$

Solving for the v_c independently produces the same results as solving for all of V in one linear program (as in [2]) while simplifying the method.

3.5. Simultaneous L_1 Factorization

Minimizing with respect to U and V simultaneously using the successive linear programming algorithm in Section 3.3 is a promising alternative to Wiberg. Successive linear programming can converge quadratically[1], and in our experiments its convergence and speed compete strongly with Wiberg.

To minimize with respect U and V simultaneously, we use the same objective function (15) and error function as the Wiberg algorithm. So, all we need are the derivatives of the predictions with respect to U and V for the update step (13). The derivatives of the prediction $u_{r_i} v_{c_i}$ with respect to u_{r_i} and v_{c_i} are:

$$\frac{du_{r_i} v_{c_i}}{du_{r_i}} = v_{c_i}^{\mathrm{T}} \tag{21}$$

$$\frac{du_{r_i} v_{c_i}}{dv_{c_i}} = u_{r_i} \tag{22}$$

The other derivatives are zero.

We'll also look at simultaneous minimization as an alternative our general and nested Wiberg algorithms in Sections 4 and 5 below. When we refer to a "simultaneous" algorithm below, we mean minimizing with respect to all of the unknowns as in this subsection.

3.6. Analysis

Wiberg effectively removes V from the minimization, which at first blush promises to be faster than minimizing with respect to U and V simultaneously. But, L_1 Wiberg has two speed issues.

First, when we solve for the step in each iteration in (13), we solve a linear programming problem (9), (10) that includes the error t_i for each observation as an unknown. The t_i's remain whether we use Wiberg or simultaneous minimization, and their number can dwarf the size of U and V. So, depending on the original problem dimensions, removing V may not reduce the linear programming problem size significantly. Second and more important, the derivative matrix in solving for the step in (13) is denser for Wiberg (18) than for simultaneous minimization (21), (22), which greatly increases the linear programming time.

So, we'll see in the results below that either Wiberg or simultaneous minimization can be faster depending on the problem size and sparsity. This consideration also carries over to the general and nested Wiberg methods.

Section 3.5 mentioned that successive linear programming can converge quadratically, and as a straightforward instance of successive linear programming, the simultaneous factorization in that section can converge quadratically. The Wiberg factorization, general Wiberg, and nested Wiberg algorithms in Sections 3.4, 4, and 5 use successive linear programming as the outer loop and, surprising, can also converge quadratically despite their increasing complexity.

4. General Wiberg Minimization

In this section we generalize the Wiberg approach to arbitrary nonlinear functions of two sets of variables. We call the resulting algorithm general Wiberg minimization. As an example of this idea, we implement L_1 bundle adjustment as a general Wiberg minimization.

4.1. General Algorithm

Wiberg L_1 factorization solves for V and its derivative with respect to U using the closed-form linear minimization in 3.2, but solves for U using the iterative nonlinear minimization in 3.3. So, adapting the algorithm to minimize a nonlinear function of U is straightforward – possibly just by changing a few lines of code – as long as the function is linear in V. But many functions are nonlinear in two sets of variables. In bundle adjustment, for instance, the objective function is a sum of reprojection errors, which are nonlinear in both the three-dimensional point positions and the six-degree-of-freedom camera positions.

To handle objective functions like these, we use iterative minimization for V as well as U. With this approach, we have an outer loop that minimizes with respect to U, and within each U iteration, we have an inner loop that minimizes with respect to V. This method is best suited for problems like bundle adjustment (and factorization) where given U, V breaks down into independent subproblems v_c. In this case the time for iteratively solving for the v_c is small because each v_c is much smaller than U.

But in the Wiberg approach, the v_c's vary implicitly with U via dv_c/dU. How do we find dv_c/dU if we found v_c iteratively?

Consider solving for v_c using the algorithm in 3.3, by substituting v_c for x there. Then, our final estimate for v_c is

$v_c^{\text{previous}} + \delta_{v_c}$ for some constant v_c^{previous}. So, the derivative of v_c with respect to U is the derivative of δ_{v_c}.

$$\frac{dv_c}{dU} = \frac{d\delta_{v_c}}{dU} \quad (23)$$

$$= \frac{d\delta_{v_c}}{d(dp(v_c)/dv_c)} \frac{d(dp(v_c)/dv_c)}{dU} \quad (24)$$

$$+ \frac{d\delta_{v_c}}{d\text{error}(v_c)} \frac{d\text{error}(v_c)}{dU} \quad (25)$$

where $p(v_c)$ is our prediction for the observations that are a function of v_c. The derivatives of $\text{error}(v_c)$ and $dp(v_c)/dv_c$ with respect to U depend on the specific function we're minimizing.

Once we have the v_c's and dv_c/dU's, we compute the derivatives of the predictions with respect to U by generalizing equation (18) to:

$$\frac{dp(U)}{dU} = \frac{\partial p(U)}{\partial U} + \frac{\partial p(V)}{\partial V}\frac{dV}{dU} \quad (26)$$

We can then minimize with respect to U by substituting this derivative for $df(x)/dx$ in Section 3.3, just as we did in the Wiberg factorization.

If the inner iterations for v_c converge, the final steps δ_{v_c} will be zero. This means that in the linear programming solution for δ_{v_c}, the simplex method can exclude elements of δ_{v_c} from the basis, and its derivatives with respect to U will be zero according to the method we described in Sections 3.1 and 3.2. In this case, the method degenerates to an EM-like method in which the v_c's are effectively fixed during the U update.

To prevent this, we ensure that v_c will be included in the basis as follows. Instead of substituting $\text{error}(v_c)$ and $dp(v_c)/dv_c$ for $\text{error}(x)$ and $df(x)/dx$ directly in equation (13):

$$\min_{\delta_{v_c}} ||\text{error}(v_c) - \frac{dp(v_c)}{dv_c}\delta_{v_c}||_1 \quad (27)$$

we instead solve for $\delta'_{v_c} = \delta_{v_c} + \epsilon$, $\epsilon = [10^{-6} \ldots 10^{-6}]$ in:

$$\min_{\delta'_{v_c}} ||(\text{error}(v_c) + \frac{dp(v_c)}{dv_c}\epsilon) - \frac{dp(v_c)}{dv_c}\delta'_{v_c}||_1 \quad (28)$$

δ'_{v_c} will be ϵ at convergence, and included in the basis since it is nonzero. We then take $\delta_{v_c} = \delta'_{v_c} - \epsilon$, and take the derivatives of δ_{v_c} to be those of δ'_{v_c}. This method works well for including v_c in the basis, although other approaches are possible.

Section 3.3 explained that a trust region is necessary to prevent divergence of the successive linear programming iteration. In our general Wiberg method, both our outer and inner iterations are successive linear programming, and we've found that the trust region is necessary to prevent divergence in both cases.

4.2. Wiberg L_1 Bundle Adjustment

Bundle adjustment is the go-to algorithm for structure-from-motion. Given two-dimensional observations $x_{i,j}$ of three-dimensional points in an image collection, bundle adjustment estimates the three-dimensional position of the each point X_j and the six-degree-of-freedom position (rotation ρ_i and translation t_i) of the camera for each image, by minimizing:

$$\sum_{i,j} ||x_{i,j} - \pi(R(\rho_i)X_j + t_i)||_2^2 \quad (29)$$

where π is the perspective projection and $R(\rho_i)$ is the rotation matrix for Euler angles ρ_i. But, least squares can be sensitive to outliers in the observations, and bundle adjustment in particular requires aggressive outlier detection. Minimizing the L_1 norm instead would be naturally more robust to outliers:

$$\sum_{i,j} ||x_{i,j} - \pi(R(\rho_i)X_j + t_i)||_1 \quad (30)$$

The Huber norm has long been bundle-adjusted and provides a smooth, arbitrarily close approximation to the L_1 norm[5]. However, the Wiberg and simultaneous algorithms can minimize (30) without approximation, and we present results for both general Wiberg and simultaneous L_1 bundle adjustment below, along with a comparison of L_1 and L_2 bundle adjustment. For Wiberg, we've arbitrarily chosen to minimize with respect to the individual points in inner iterations, and to minimize with respect to the camera rotations and translations in the outer iteration.

4.3. Solving Large Problems with the Primal

Linear programs can be solved using either the primal or the dual formulation. The dual is usually faster, and we report dual times for the synthetic experiments in Section 6. But as shown there, solve times grow quickly with problem size even with the dual, making large problems impractical.

In contrast, the primal solve is slower, but can return some solution even when interrupted after a fixed time. This stopped solution might not be optimal or even feasible, but when used to find δ_x in (13), even a suboptimal solution sometimes reduces the overall objective. Further, solving problems with small trust regions seems to be faster. So, if a stopped solution does not reduce the objective on one iteration, a (possibly stopped) solution on a subsequent iteration with a smaller trust region will produce a useful step. We used this strategy successfully to estimate structure and motion from the long "rover" sequence in Section 6.2, which includes about 700 images and 10,000 points.

5. Nested Wiberg Minimization

Our general Wiberg minimization in Section 4.1 works by solving for V iteratively rather than in closed form. Since it

is found iteratively, V can itself be split into two sets of variables found using the Wiberg approach. This results in a nested Wiberg minimization that can effectively minimize with respect to three sets of variables. In this section, we demonstrate this idea on L_1 projective structure-from-motion, where the three sets of unknowns are camera matrices, three-dimensional point positions, and projective depths.

So suppose we have three sets of variables U, V, and D; that given U and V we minimize with respect to D in closed form; that given U we minimize with respect to V and D in an inner iteration; and that we minimize with respect to U in an outer iteration. Then to minimize with respect to U using the nonlinear L_1 minimization, we'll need the total derivative of our predictions p with respect to U:

$$\frac{dp}{dU} = \frac{\partial p}{\partial U} + \left(\frac{\partial p}{\partial V} + \frac{\partial p}{\partial D}\frac{dD}{dV} \right) \frac{dV}{dU} + \frac{\partial p}{\partial D}\frac{dD}{dU} \quad (31)$$

Equation (26) is a total derivative of p with respect to U, with V a function of U. Equation (31) generalizes this to the total derivative of p with respect to U, with V a function of U and D a function of both U and V. The factor in parentheses is the total derivative of p with respect to V, with D a function of V, and reflects the nesting.

Deriving (31) requires us to expand a complex tree of derivatives. Because of limited space, we can't completely explore that tree here. But to suggest the full procedure, we summarize one path from the root of this tree as follows:

1. (31) includes the factor dV/dU.
2. dV/dU is similar to equation (23-25), and includes $d(dp(v_c)/dv_c)/dU$.
3. $dp(v_c)/dv_c$ is a total derivative and includes the factor dD/d_{v_c}.
4. D is found using the procedure in 3.2, which also gives the derivatives of D with respect to the coefficient matrix and right-hand side there.
5. That coefficient matrix and right-hand side are functions of v_c, so we can use the chain rule to get the derivative of D with respect to v_c.

The derivatives at the other leaves are found similarly and combined using the rules for matrix derivatives[3].

5.1. Wiberg L_1 Projective Bundle Adjustment

The bundle adjustment in Section 4.2 can be used when the camera calibration (e.g., focal length) is known. In contrast, projective bundle adjustment recovers structure and motion from uncalibrated image sequences. A projective reconstruction includes 3×4 camera projection matrices C_i and 4-dimensional projective points X_j that are consistent with the image observations and are known up to a common 4×4 transformation. This transformation can be identified, and the projective reconstruction upgraded to a metric reconstruction, given some knowledge of the scene geometry or camera intrinsics.

Our objective function is:

$$\sum_{i,j} ||[u_{i,j}\ v_{i,j}\ 1]^{\mathrm{T}} - d_{i,j} C_i X_j||_1 \quad (32)$$

where $(u_{i,j}, v_{i,j})$ and $d_{i,j}$ are the two-dimensional image location and inverse projective depth of point j in image i. We can minimize (32) with respect to C_i, X_j, and $d_{i,j}$, using either nested Wiberg minimization or simultaneous minimization.

We present results for both Wiberg and simultaneous minimization in Section 6.3 below. For Wiberg, we've chosen to find each inverse depth independently given point and projection matrix estimates, in closed form; to find each projection matrix given point estimates using an inner Wiberg minimization, letting the inverse depths vary implicitly; and to solve for the points in an outer Wiberg minimization, letting the projection matrices and inverse depths vary implicitly. In short, U represents the points, V the camera matrices, and D the projective depths.

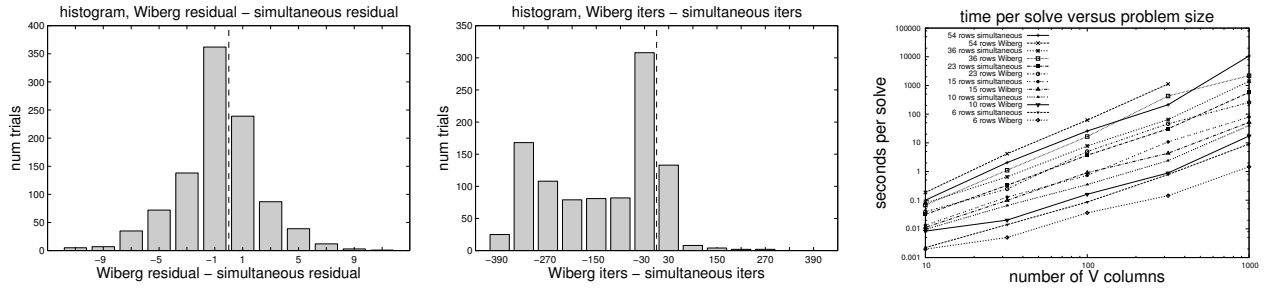
Given point and projection matrix estimates, it's also possible to "read off"[4] the projective depths as the last element of CX rather than estimating them. This results in the projective bundle adjustment algorithm given by Hartley and Zisserman[5]. Here, we explicitly estimate the inverse depths as an example of a nested Wiberg minimization. The objective function using this approach is similar to that in factorization methods for projective structure-from-motion, which do require that the depths be explicitly estimated. Oliensis and Hartley[9] also perform an L_2 projective bundle adjustment by minimizing with respect to the projection matrices, points, and depths.

6. Results

6.1. Factorization

In this section we briefly compare Eriksson and van den Hengel's Wiberg L_1 factorization algorithm (Section 3.4) against the simultaneous algorithm described in Section 3.5. The simultaneous algorithm is a stronger baseline than the baseline Eriksson and van den Hengel consider, Ke and Kanade's alternative convex programming approach[6].

In an experiment similar to Eriksson and van den Hengel's[2], we generated 1000 random (not low-rank by construction) measurement matrices Y with missing entries and outliers. We factored each into rank 3 matrices using both Wiberg and simultaneous minimization. The results are summarized in Figure 1. Figure 1(a) shows that Wiberg



(a) (b) (c)

Figure 1. Wiberg versus simultaneous matrix factorization on 1000 synthetic examples. Wiberg produces slightly better final residuals (a) and usually requires fewer iterations to converge (b). But, the simultaneous method has much faster individual iterations than Wiberg for large problem sizes, with the crossover at 15 or 23 rows in U in this example (c). Note that both axes are logarithmic in (c).

produced slightly better residuals. But as shown in Figure 1(b), Wiberg often converged in many fewer iterations.

In these small problems (factoring a 7×12 matrix into rank 3 factors), the linear programming solve time for the main update step was 1.1 ms for the Wiberg step and 3.1 ms for the simultaneous algorithm step. However, for larger problems, the linear programming time increases quickly, as shown in Figure 1(c). This figure plots the time for the linear programming solve for one U update, for different problem sizes, for both the Wiberg and simultaneous approaches. The horizontal axis gives the number of columns in V; the vertical axis gives the time in seconds; and the separate plots give the times for different numbers of rows in U. Note that both axes are logarithmic. The graph reflects our analysis in Section 3.6. Wiberg is faster than simultaneous if U has few rows, and simultaneous is faster otherwise.

6.2. Bundle Adjustment

We compared the convergence and speed of Wiberg and simultaneous L_1 bundle adjustment, using experiments similar to the factorization experiments in Section 6.1. For a wide range of initial errors in the initial camera and point estimates, Wiberg and simultaneous both converged to the ground truth residual, with Wiberg sometimes converging in fewer iterations. For gross errors in the initial estimates, Wiberg usually converged to a lower residual than simultaneous, but not always to the ground truth residual. The Wiberg method linear programming solve was faster than the simultaneous solve only for problems with 2 or 3 images, while simultaneous was faster for 5 or more images. The supplementary material contains more detailed descriptions and plots for these experiments.

We also compared the robustness of L_1 and L_2 bundle adjustment, and demonstrated that the Wiberg algorithm correctly estimates structure and motion for "rover"[13], a real sequence with about 700 images, 10,000 points, and extreme perspective effects (Figure 2). These experiments are also described in the supplementary material.

6.3. Projective Bundle Adjustment

We compared the convergence of the nested Wiberg and simultaneous projective bundle adjustment algorithms using synthetic experiments similar to those in Section 6.2. In brief, for final residuals, both methods converged reliably to the ground truth residual for a large range of initial estimates; Wiberg converged more reliably for wider ranges of projective depths; and simultaneous produced smaller residuals for the largest gross errors in the initial estimates, although the final residuals were not usually the ground truth residual in these extreme examples. For iterations until convergence, Wiberg reliably converged to its final residual in fewer iterations than simultaneous. In one timing experiment, we found that the Wiberg algorithm had faster linear programming solves than simultaneous for problems including up to 51 points, and simultaneous had faster solves for problems with more than 51 points.

The supplementary material includes more detailed descriptions and plots for these experiments.

7. Conclusion

We've introduced general and nested Wiberg minimization, which extend Wiberg's approach to matrix factorization to general functions that are nonlinear in two or three sets of variables. We've focused on L_1 minimization, extending Eriksson and van den Hengel's Wiberg L_1 factorization, and shown that L_1 bundle adjustment and projective bundle adjustment can be implemented as general and nested Wiberg minimizations, respectively. We also introduced successive linear programming algorithms for these problems, which estimate all of the unknowns simultaneously.

Both the Wiberg and simultaneous algorithms can converge quadratically, despite the complexity of the Wiberg approach, and both converge from a wide range of initial estimates. Wiberg reliably converges to its final residual in fewer iterations than simultaneous, but as Section 3.6 explains, the simultaneous approach produces a sparser linear

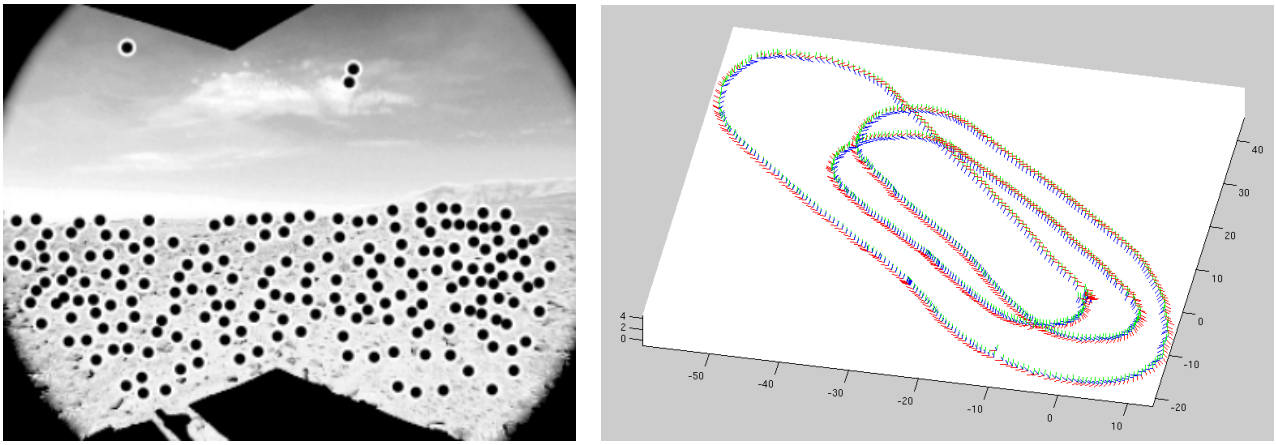


Figure 2. L_1-Wiberg bundle adjustment reconstructs the correct structure and motion from the "rover" sequence, which includes about 700 images and 10,000 points. Left: an example image from the sequence, with tracked points shown as black dots. Right: an oblique view of the recovered camera positions at the time of each image.

program than Wiberg, so simultaneous iterations are faster for large problems. But as described n Section 4.3, even very large Wiberg problems can be solved effectively by combining primal linear program solves with the adaptive trust region in successive linear programming.

The general Wiberg approach can be used for L_1 minimization, L_2 minimization, and maximum likelihood estimation, and there are many potential applications besides those we've described. In an upcoming paper, we'll describe Wiberg L_2 bundle adjustment and Poisson matrix factorization using Wiberg maximum likelihood estimation.

Acknowledgments. Thanks to Emilie Danna for her linear programming advice, which made the large-scale rover result possible, as described in Section 4.3; to Jay Yagnik, Luca Bertelli, Mei Han, Vivek Kwatra, Mohamed Eldawy, and Rich Gossweiler for feedback on early versions of this paper; to Jim Teza, Chris Urmson, Michael Wagner, and David Wettergreen for capturing the "rover" sequence; and to the anonymous reviewers for suggesting the experimental comparison between L_1 and L_2 bundle adjustment and pointing out the Huber norm.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. Shetty. *Nonlinear programming: theory and algorithms*. Wiley, Hoboken, New Jersey, third edition, 2006.

[2] A. Eriksson and A. van den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L_1 norm. In *Computer Vision and Pattern Recognition*, San Francisco, CA, June 2010.

[3] P. L. Fackler. Notes on matrix calculus. http://www4.ncsu.edu/∼pfackler/MatCalc.pdf, September 2005. Accessed: 08/29/2011.

[4] D. A. Forsyth and J. Ponce. *Computer vision: A modern approach*. Prentice Hall, Upper Saddle River, New Jersey, 2003.

[5] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, Cambridge, UK, second edition, 2003.

[6] Q. Ke and T. Kanade. Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[7] T. Okatani and K. Deguchi. On the Wiberg algorithm for matrix factorization in the presence of missing components. *International Journal of Computer Vision*, 72(3), May 2007.

[8] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *International Conference on Computer Vision*, Barcelona, Spain, November 2011.

[9] J. Oliensis and R. Hartley. Iterative extensions of the Sturm/Triggs algorithm: convergence and nonconvergence. *IEEE PAMI*, 29(12):2217–2233, December 2007.

[10] C. Poelman. *The paraperspective and projective factorization methods for recovering shape and motion*. PhD thesis, Carnegie Mellon University, 1995.

[11] F. Richards. A method of maximum-likelihood estimation. *Journal of the Royal Statistical Society, Series B (Methodological)*, 23(2):469–475, 1961.

[12] A. Ruhe and P. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):318–337, 1980.

[13] D. Strelow and S. Singh. Motion estimation from image and inertial measurements. *International Journal of Robotics Research*, 23(12):1157–1195, December 2004.

[14] T. Wiberg. Computation of principal components when data are missing. In *Second Symposium of Computation Statistics*, pages 229–326, Berlin, 1976.