General, Nested, and Constrained Wiberg Minimization

Dennis Strelow, Qifan Wang, Luo Si, and Anders Eriksson

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 $||Y - f(U, V)||_1$ for more general functions $f(U, V)$ that are nonlinear in each of two sets of variables. We demonstrate the idea with a practical Wiberg algorithm for L_1 bundle adjustment. 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. Wiberg minimization also generalizes to handle nonlinear constraints, and we demonstrate this idea with Constrained Wiberg Minimization for Multiple Instance Learning (CWM-MIL), which removes one set of variables from the constrained optimization. Our experiments emphasize isolating the effect of Wiberg by comparing against the algorithm it modifies, successive linear programming.

Index Terms—Low-rank matrix factorization, L_1 minimization, successive linear programming, structure-from-motion, multiple instance learning.

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[1] approached the least squares 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[2][3]. Recently Eriksson and van den Hengel[4][5] 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 $||Y - f(U, V)||_1$ for more general nonlinear functions of two sets of variables, $f(U, V)$. Our general Wiberg minimization can be used for L_1 minimization, least squares 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. Splitting V 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 approach can also handle nonlinear constraints, and we demonstrate this idea with Constrained Wiberg Minimization for Multiple Instance Learning (CWM-MIL), which effectively removes one set of variables from the constrained minimization and optimizes with respect to just the remaining variables. An extensive set of experiments on several benchmark datasets demonstrates that the proposed CWM-MIL approach converges much faster than several state-of-the-art MIL methods, and usually converges to a local minimum with a lower objective than alternating methods.

Our main contributions are:

- General, nested, and constrained Wiberg minimization, and their development for L_1 minimization.
- 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.
- CWM-MIL, Constrained Wiberg Minimization for Multiple Instance Learning, and multiple instance

- *Dennis Strelow and Qifan Wang are with Google, 1600 Amphitheatre Parkway, Mountain View, CA 95118.*
  E-mail: strelow@google.com, wqfcr@google.com.
- *Luo Si is with the Department of Computer Science, Purdue University, 300 N. University Street, West Lafayette, IN 47907-2066.*
  E-mail: lsi@purdue.edu.
- *Anders Eriksson is with the School of Electrical Engineering and Computer Science, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001.*
  E-mail: anders.eriksson@qut.edu.au.

learning using successive linear programming.

We also revisit Eriksson and van den Hengel's original problem, L_1 matrix factorization. We present a space-optimized implementation of the algorithm that scales to much larger problems than the original reference implementation; 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 faster than than L_1 Wiberg for some problem sizes.

Each of our L_1 Wiberg algorithms, as well as Eriksson and van den Hengel's factorization algorithm, modifies an existing algorithm: minimizing with respect to all of the unknowns simultaneously using successive linear programming. So, our experiments focus on isolating Wiberg's effect by comparing against successive linear programming.

2 Related work

Wiberg[1] presented a least squares 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 al.*[2][3] 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[4][5] 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[2] least squares experiment with Wiberg and Levenberg-Marquardt.

Wiberg's method was an application of Ruhe and Wedin's[7] 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[8] 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, and we present general and nested Wiberg for least squares and maximum likelihood problems in a separate paper[9]. For L_1, which we consider in this paper, 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[2] 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[11], which often converges quadratically.

Since Eriksson and van den Hengel[4], new factorization algorithms have convincingly improved on L_1 Wiberg's scalability, speed, and convergence. Liu, *et al.*[12] target the same L_1-norm objective as L_1 Wiberg, and produce the same final residuals in most trials in their experiments. But, they adopt a dynamic system approach that is faster than and handles larger problems than the successive linear programming approach in L_1 Wiberg. Zheng, *et al.*[13] tweaked the non-convex, L_1-norm objective by adding a convex trace-norm regularization term and optimized it with a first-order method, and their method's convergence dominated Wiberg in experiments. Wang, *et al.*[14] minimize a very different (non-L_1) objective function that explicitly models corrupted elements in the observed matrix. They find an initial solution using a convex relaxation of the objective, then refine that solution by alternating between factoring the matrix and identifying the corrupt entries. Although they don't optimize the same norm as [4], they demonstrate their algorithm reliably factoring matrices that [4] cannot.

Our work differs from these in that matrix factorization isn't the problem we're solving. Instead, we extend the L_1 Wiberg idea to more general problems outside of factorization.

However, a common theme in [12][13][14] is the poor scalability of [4], and in this paper we revisit and greatly improve L_1 Wiberg's scalability. In Section 3.6, we describe an implementation that eliminates most of the space used by [4], removing space as the limiting factor. In Section 4.3, we describe a heuristic for solving large problems quickly by terminating linear program solves early. With these improvements we can solve much larger problems than [4], and we use them with general Wiberg to solve a structure-from-motion problem with about 700 images and 10,000 points in Section 9.

Strelow[15] is an earlier presentation of the general and nested Wiberg methods given in this paper.

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); a more scalable implementation of L_1 Wiberg factorization than Eriksson and van den Hengel's original (3.6); and a qualitative analysis of L_1 Wiberg (3.7).

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[16] 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:

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

(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.

Suppose A is in $R^{m\times n}$ with linearly independent rows. 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 $Ax \le b$ to equalities by introducing nonnegative slack variables s:

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

The optimal solution $[x; s]$ will include basic components x_B, which can be nonzero; and nonbasic components x_N, which will be zero. Then by Theorem 2.4 of [17] there exists an $x_B \in R^m$ such that the corresponding columns of $[A\, I]$, denoted B, are linearly independent and $Bx_B = b$. Then since $x_B = B^{-1}b$, it can be shown[18] 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.

Our implementation uses Clp (COIN-OR linear programming) for linear programming.

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 $x \ge 0$, 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:

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

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

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

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

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

Note that in addition to the explicit inequality constraints in (10), the linear program also inherits the $x \ge 0$ constraint from (1), so $y^+ \ge 0$, $y^- \ge 0$, and $t \ge 0$.

The objective function in (9) just says to minimize the sum of the individual L_1 errors, the t_i's. 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.

Note that just in themselves, the inequalities (7), (8) aren't equivalent to the equality (6). But, since we minimize with respect to t_i, one of (7) or (8) will be tight, and so equivalent to (6).

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[11] 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.

The Wiberg approach alternates between "inner" and "outer" solve steps. First, in the inner solve step, 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.

Second, in the outer solve step, 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

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

Equation (18) is the total derivative of the prediction $u_{r_i}v_{c_i}$ with respect to U from freshman calculus. The total derivative accounts for the fact that the prediction depends on U directly, but also indirectly through $v_{c_i}(U)$. It contrasts with the partial derivative $\partial u_{r_i}v_{c_i}/\partial U$ (also present in Equation (18)), which only models the prediction's direct dependency on U. We'll revisit the total derivative in the general Wiberg section (4.1) and nested Wiberg section (5) below.

Solving for the v_c independently produces the same results as solving for all of V in one linear program (as in [4], [5]) while simplifying the method. Note that although the v_c's are solved independently, each v_c is still a function of the entire matrix U. So, the outer minimization still solves with respect to all of the elements of U simultaneously.

3.5 Simultaneous L_1 Factorization

Eriksson and van den Hengel's L_1 Wiberg factorization in Section 3.4 is a close modification of the successive linear programming algorithm in Section 3.3. So, using successive linear programming to just minimize with respect to U and V simultaneously is the most relevant baseline for isolating and evaluating the change that Wiberg introduces. Successive linear programming can converge quadratically[11], 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 Scalable implementation

As [12], [13], and [14] describe, L_1 Wiberg factorization as originally described in Eriksson and van den Hengel's paper[4] and reference implementation doesn't scale. Space is the most limiting factor: equations (37)-(40) in [4] explicitly compute a series of derivatives using Kronecker products, and these matrices can explode in size. Liu et al.[12] identify dA/dU (equation 38 in [4]) as the worst offender, giving its full size as $(4mn^2r + 6m^2n^2) \times (mn^2r)$, and we found that even just the nonzero elements of dA/dU quickly become too numerous to store as the problem size increases.

However, the derivative we actually need, dy/dC, can be computed relatively easily without Kronecker products or explicit building dA/dU. In outline:

- $x = [y^+ \, y^- \, t]$ and $y = y^+ - y^-$, so dy/dC incorporates some components from dx/dC.
- $A = [C \; -C \; I; -C \; C \; I]$, so dx/dC incorporates some components from dx/dA.
- dx/dA is a zero padded version of dx_B/dB, given by equation (3).

Using this approach and sparse matrix representation for the matrices that we do compute explicitly, space is no longer a limiting factor, even for very large problems. For instance, in our general Wiberg results below, we build and solve a structure-from-motion problem ("rover") with about 700 images and about 10,000 points.

Instead, the limiting factor becomes the linear program solve times for Equation (13) in the outer iteration refining U. We investigate this solve time for different problem sizes throughout our results in Section 9. To help sidestep the linear program time for large problems, we introduce an early termination heuristic in Section 4.3 that's applicable to problems like "rover."

3.7 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. Section A in the supplementary material explores the relative sparsity of the two approaches more.

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 more general 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 alternate between an inner solve and an outer solve just like we did for factorization in Section 3.4, but now the inner solve is a loop that iteratively minimizes with respect to V. This method is best suited for problems like bundle adjustment 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}.

$$\frac{dv_c}{dU} = \frac{d\delta_{v_c}}{dU} \tag{23}$$

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

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

where $p(v_c)$ is our prediction for the observations that are a function of v_c, i.e., the components of $f(U, V)$ that depend on v_c. Here, the derivatives are computed in a manner similar to Section 3.2, with δ_{v_c}, $dp(v_c)/dv_c$, and $\mathrm{error}(v_c)$ playing the role of y, C, and d there. The derivatives of $\mathrm{error}(v_c)$ and $dp(v_c)/dv_c$ with respect to U depend on the specific function we're minimizing.

Note that general Wiberg solves more general problems than factorization, but f cannot be completely arbitrary: f must have first derivatives with respect to U and V, and the second derivative with respect to V then U, which appears as $d(dp(v_c)/dv_c)/dU$ on the right of equation (24). (Here as above, p is the application of f corresponding to one of the input observations.)

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:

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

Like Equation (18), Equation (26) is a total derivative that accounts for the direct dependency of p on U, but also p's indirect dependency on U through $V(U)$. 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 $\mathrm{error}(v_c)$ and $dp(v_c)/dv_c$ for $\mathrm{error}(x)$ and $df(x)/dx$ directly in equation (13):

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

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

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

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:

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

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:

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

The Huber norm has long been bundle-adjusted and provides a smooth, arbitrarily close approximation to the L_1 norm[19]. 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 least squares 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 Early Termination

As shown in Section 9, our linear program solve times grow quickly with problem size for both the Wiberg and simultaneous algorithm, which at first seems to make large problems impractical. However, we can interrupt solves after a fixed time. The stopped solution might not be optimal or even feasible, but when used to find δ_x in (13), even a suboptimal solution often 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 C, which includes about 700 images and 10,000 points.

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 9. For the rover sequence in Section C, we used the primal, since our software allowed early termination only with the primal. But in general, using complementary slackness, it is also possible to recover an approximate primal solution from an approximate (prematurely terminated) dual solution.

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; 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. Motivated by our projective bundle adjustment application below, we'll assume that D can be solved in closed form given U, V. Then the algorithm is composed of three increasingly complex L_1 minimizations:

Innermost solve for D: In the innermost solve, we're given current estimates of U and V, and we solve for an individual component $d_{p,c}$ of D, corresponding to one block U_p of U and one block V_c of V. This is a linear L_1 minimization as described in Section 3.2.

Middle solve for V: In the middle solve, we're given current estimates of U and V, and the main task is to refine each block V_c of V. On each iteration, we also solve for each component of D that depends on V_c in closed form using the innermost solve above.

This solve is a Wiberg minimization where the objective is linear in the inner variables (components of D), so similar to the Wiberg matrix factorization in Section 3.4.

Outer solve for U: In the outermost solve, we're given initial estimates of U and V, and the main task is to refine U. On each iteration, we also iteratively refine each block V_c of V using the middle solve above, which in turn solves for each component of D in closed form.

This solve is nearly the same as the general Wiberg minimization in Section 4. The difference is that there, the coefficient matrix in the system for the step ΔU is composed of total derivatives of the predictions with respect to U (equation (26)), and the predictions there are functions of U and V(U). Here, the predictions are functions of three sets of variables U, V(U), and D(U, V(U)), so the total derivative becomes:

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

The factor in parentheses is the total derivative of p with respect to V, with D a function of V, and reflects the nesting.

In (31), $\partial p/\partial U$, $\partial p/\partial V$, and $\partial p/\partial D$ are straightforward partial derivatives that depend on the form of p (i.e., f). The middle term, $(\partial p/\partial V) + (\partial p/\partial D)(dD/dV)$, is the same total derivative used in computing the V step in the middle iteration; it's the same total derivative as Equation (18) (where it appears in L_1 Wiberg matrix factorization) and Equation (26) (where it appears in general Wiberg).

The formula for dV/dU is similar to equation (23-25) in the non-nested general Wiberg algorithm. The major difference is that in the nested case, $dp(v_c)/dv_c$ is a total derivative reflecting the fact that D changes with v_c, and the derivation of dV/dU becomes complex in accounting for this.

dD/dU is also a potentially complicated term, since D is a function of V, which is in turn a function of U. Although we computed dV/dU analytically in our implementation of projective bundle adjustment below, we took the reasonable shortcut of using finite differences for dD/dU.

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.

We've implemented three algorithms for projective structure-from-motion: (1) a successive linear programming algorithm that minimizes with respect to all of the unknowns simultaneously, (2) a nested Wiberg algorithm, and (3) a non-nested Wiberg algorithm. (1) and (2) minimize the following objective function with respect to the camera matrices, three-dimensional points, and inverse projective depths $d_{i,j}$:

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

where $(u_{i,j}, v_{i,j})$ is the two-dimensional image location of point j in image i.

For nested Wiberg (algorithm 2), 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. Note that this differs from our general Wiberg bundle adjustment in Section 4.2, where the outer iteration variable U was the cameras, and the inner iteration variable V was the points; the choice of which variables are the inside or outside variables is arbitrary.

In algorithms 1 and 2, 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[20] also perform a least squares projective bundle adjustment by minimizing with respect to the projection matrices, points, and depths. We compare algorithms 1 and 2 in Section D in the supplementary material.

But given point and projection matrix estimates, it's also possible to "read off"[21] 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[19]. Algorithm 3 is an non-nested Wiberg algorithm using this approach, which estimates the camera matrices and three-dimensional points, but doesn't estimate the projective depths explicitly. We compare this approach with the nested Wiberg algorithm in Section D.3 in the supplementary material.

6 Multiple Instance Learning

Our general Wiberg approach can also handle nonlinear constraints, and we explore constrained Wiberg and its application to multiple instance learning in the next three sections.

Traditional learning methods treat each data example as a single instance and represent the example by one feature vector. However, this single instance setting is not desirable in many scenarios. In image classification, the semantic meaning of an image is usually represented by a certain region or regions inside it, rather than the whole image with irrelevant backgrounds. As one variation of traditional learning methods, Multiple Instance Learning (MIL) [22], [23], [24], [25] has been proposed. In MIL, each data example is treated as a bag consisting of multiple instances. The labels are assigned to the bags rather than individual instances under the assumption that *a bag is labeled positive if at least one of its instances is positive, whereas a negative bag only contains negative instances*. In the case of image classification, each image is treated as a bag and different regions inside the image are viewed as individual instances.

In this section, we first provide a unified framework to describe previous MIL formulations, and give our formulation of MIL as well as the alternating optimization method. In Section 7, we describe a constrained Wiberg algorithm for MIL. In Section 8 we describe constrained Wiberg for general problems.

Suppose there are n labeled bags: $D = \{(B_i, y_i), i = 1, \ldots, n\}$ in the training set, where B_i represents the i-th bag and $y_i \in \{+1, -1\}$ is its label. The bag B_i consists of a set of instances denoted as: $B_i = \{B_{i1}, \ldots, B_{in_i}\}$, where $B_{ij} \in R^d$ is the j-th instance in B_i and n_i is the number of instances in the i-th bag.

6.1 Previous MIL Formulations

The main challenge of MIL is the label ambiguity problem, since labels are assigned to bags rather than individual instances. Therefore, most MIL methods explicitly or implicitly introduce a set of variables, $v_i = \{v_{i1}, \ldots, v_{in_i}\}$ for each bag, to indicate whether its instances are positive or negative. Existing MIL formulations can be summarized into two groups: instance level and bag level.

The instance level framework formulates the MIL as:

$$\begin{aligned} \min_{w,v,\xi} \quad & \|w\|_{\{1,2\}} + \lambda \sum_{i,j} \xi_{ij} \\ s.t. \quad & v_{ij} w^T B_{ij} \geq 1 - \xi_{ij} \\ & v \in C, \quad \xi_{ij} \geq 0 \end{aligned} \tag{33}$$

where w is the classifier and ξ_{ij} is the classification loss on each instance. $\|\|_{\{1,2\}}$ stands for an L_1 or least squares regularizer. C is the set of multiple instance constraints and different MIL methods may form different constraints. The mi-SVM [26] method explicitly transforms the MIL assumption into constraints C: $\{v_{ij} \in \{+1, -1\},\ v_{ij} = -1\ \forall y_i = -1,\ and\ \sum_j \frac{v_{ij}+1}{2} \geq 1\ \forall y_i = 1\}$. [24] modifies the classification loss and margin constraints by treating positive and negative bag loss asymmetrically. Alternating methods are utilized to solve the optimization problem in Equation (33): (1) update v given the current classifier w by setting $v_{ij} = sgn(w^T B_{ij})$ for instances in positive bags[1]; (2) solve w using a standard SVM by fixing v.

The bag level framework formulates the MIL as:

$$\begin{aligned} \min_{w,v,\xi} \quad & \|w\|_{\{1,2\}} + \lambda \sum_{i} \xi_{i} \\ s.t. \quad & y_i w^T \phi(B_i, v_i) \geq 1 - \xi_{i} \\ & v \in C, \quad \xi_{i} \geq 0 \end{aligned} \tag{34}$$

here ξ_i is the classification loss on each bag and ϕ is the function to construct bag level features based on the instances and indicating variables. For example, in [22], [27], [28], each bag is represented by the most positive instance inside it. In other words, constraints C enforce one element in v_i to be 1 with others to be 0 ($v_{ij} \in \{0, 1\}$, $\sum_j v_{ij} = 1$). $\phi(B_i, v_i)$ is simply the instance B_{ij} with $v_{ij} = 1$. In [29], a convex function combines all the instances $\phi(B_i, v_i) = B_i^T v_i$. The work in [30] and [31] proposes more complex functions to construct the bag level features. These MIL methods then solve the optimization problem in Equation (34) using an EM-like alternating algorithm: (1) update v given the current classifier w by setting $v_{ij^*} = 1$ with $j^* = \arg\max_j w^T \phi(B_i, v_i)$ and $v_{ij} = 0$ for other j; and (2) solve w by fixing v and using a standard SVM.

Note that many other sophisticated MIL methods such as [28] and [32] also formulate their problems into these

1. If all v_{ij} are -1, these methods will pick the largest one and set it to be 1 since at least one instance should be positive for a positive bag.

frameworks by reformulating the constraints. An alternating scheme is adopted in these methods to solve the optimization problems.

6.2 Our MIL Formulation

We formulate the MIL problem based on the observation that a positive bag contains one or more positive instances while a negative bag only contains negative instances. Therefore, we only introduce indicating variable $v_i = \{v_{ij}\}$ on positive bags and formulate the MIL problem as follows:

$$\begin{aligned} \min_{w,v,\xi} \quad & \|w\|_{\{1,2\}} + \lambda_1 \sum_i \xi_i^+ + \lambda_2 \sum_{ij} \xi_{ij}^- \\ s.t. \quad & w^T B_i v_i \geq 1 - \xi_i^+ \quad \forall y_i = 1 \\ & -w^T B_{ij} \geq 1 - \xi_{ij}^- \quad \forall y_i = -1 \\ & \sum_j v_{ij} = 1, \quad v_i \geq 0, \quad \xi \geq 0 \end{aligned} \tag{35}$$

here ξ_i^+ and ξ_{ij}^- are the classification loss on positive bags and negative instances. λ_1 and λ_2 are the trade-off parameters. The first set of constraints ensures the classification loss of positive bags to be minimized. Each positive bag is represented by its average positive instance, $B_i v_i$, weighted by indicating variables. $\sum_j v_{ij} = 1$ and $v_i \geq 0$ are the normalization constraints, which can also be viewed as a continuous relaxation of the hard constraints that one element in v_i is 1 with others to be 0. The intuitive idea behind these constraints is that there could be more than one positive instance in each positive bag, and thus we should allow more than one indicating variables to take positive values. The second set of constraints is on each negative instance from negative bags.[2] In the rest of this paper, we will focus on L_1 minimization since it is considered to be more difficult than least squares in the literature.

The alternating optimization method for our MIL formulation is similar to the EM scheme discussed in Section 6.1. More precisely, we alternate the following two steps until convergence: (1) update v given the current classifier w by solving a linear program with respect to v; and (2) update the classifier w given the indicating variables v by solving a linear L_1 minimization problem with respect to w, which can be efficiently solved as shown in Section 3.2.

7 Constrained Wiberg Minimization

7.1 Simultaneous Minimization

Minimizing the objective in Equation (35) with respect to w and v simultaneously is more effective than alternating methods, and competes more effectively with Wiberg methods than alternating methods do. In this section, we derive a constrained simultaneous optimization method for our MIL problem.

2. While we use a linear classifier in our experiments, the formulation can be easily extended to kernels.

Successive linear programming [33] is utilized to solve the optimization problem simultaneously. The idea of SLP is to iteratively solve a sequence of linear L_1 minimization problems by linearizing the nonlinear constraints in Equation (35) during each iteration. Specifically, if at iteration t, the estimates are $[w^t, v^t, \xi^t]$, then the goal is to find new estimates, $[w^{t+1}, v^{t+1}, \xi^{t+1}] = [w^t + \delta w, v^t + \delta v, \xi^t + \delta\xi]$, that have a lower objective value:

$$\begin{aligned} \min_{\delta w, \delta v, \delta\xi} \quad & \|w^t + \delta w\|_1 + \lambda_1 \sum_i \delta\xi_i^+ + \lambda_2 \sum_{ij} \delta\xi_{ij}^- \\ s.t. \quad & (w^t + \delta w)^T B_i v_i^t + (w^t)^T B_i \delta v_i \\ & \qquad \geq 1 - \xi_i^{+,t} - \delta\xi_i^+ \qquad \forall y_i = 1 \\ & -(w^t + \delta w)^T B_{ij} \geq 1 - \xi_{ij}^{-,t} - \delta\xi_{ij}^- \qquad \forall y_i = -1 \\ & \sum_j \delta v_{ij} = 0, \quad v_i^t + \delta v_i \geq 0, \quad \xi^t + \delta\xi \geq 0 \end{aligned} \tag{36}$$

Note that the new solution may be infeasible, because the optimal δ that minimizes Equation (36) may be outside the region where the linearization is accurate. In other words, the new solution may not satisfy all the nonlinear constraints. Therefore, as described in Section 3.3, an adaptive trust region constraint is added to help ensure the feasibility, i.e., $\|\delta w\|_\infty \leq \mu$, $\|\delta v\|_\infty \leq \mu$, $\|\delta\xi\|_\infty \leq \mu$.Then the linear L_1 minimization problem in Equation (36) can be simply solved using linear programming and we iteratively solve this minimization problem until convergence to find an optimal solution of problem (35).

7.2 Wiberg Minimization

Eriksson and van den Hengel use the Wiberg method to solve L_1 matrix factorization [4][5] and Section 4 generalizes the Wiberg method to minimize any nonlinear functions of two sets of variables. However, these methods do not consider the constrained optimization problem and thus can not be applied to solve MIL problem in Equation (35) due to the nonlinear constraints.

In this section, we propose a constrained Wiberg minimization approach to efficiently solve the MIL problem. Our observation is that given the classifier w, the optimal indicating variables v can be obtained by solving the following problem:

$$\begin{aligned} \min_{v,\xi} \quad & \lambda_1 \sum_i \xi_i^+ \\ s.t. \quad & w^T B_i v_i \geq 1 - \xi_i^+ \quad \forall y_i = 1 \\ & \sum_j v_{ij} = 1, \quad v_i \geq 0, \quad \xi \geq 0 \end{aligned} \tag{37}$$

The above problem can be further decomposed into several independent smaller problems, corresponding to different positive bags. Each problem can be solved using a linear program and we can then obtain the derivatives of v_i w.r.t. w, i.e. $\frac{dv_i}{dw}$, from the linear program, as shown in Section 3.2. Thus we can represent v as an implicit function of w, i.e., $v = v(w)$, and rewrite

Equation (35) as a minimization problem with respect to w only:

$$\begin{aligned} \min_{w,\xi} \quad & \|w\|_1 + \lambda_1 \sum_i \xi_i^+ + \lambda_2 \sum_{ij} \xi_{ij}^- \\ s.t. \quad & w^T B_i v_i(w) \geq 1 - \xi_i^+ \quad \forall y_i = 1 \\ & -w^T B_{ij} \geq 1 - \xi_{ij}^- \quad \forall y_i = -1 \\ & \sum_j v_{ij}(w) = 1, \quad v_i(w) \geq 0, \quad \xi \geq 0 \end{aligned} \tag{38}$$

We also apply SLP to solve the above constrained minimization problem by linearizing the nonlinear constraints. More precisely, if at iteration t, the estimates are $[w^t, \xi^t]$, then the goal is to find new estimates, $[w^{t+1}, \xi^{t+1}] = [w^t + \delta w, \xi^t + \delta \xi]$, that have a lower objective value:

$$\begin{aligned} \min_{\delta w, \delta \xi} \quad & \|w^t + \delta w\|_1 + \lambda_1 \sum_i \delta\xi_i^+ + \lambda_2 \sum_{ij} \delta\xi_{ij}^- \\ s.t. \quad & (w^t + \delta w)^T B_i v_i^t + (w^t)^T B_i \frac{dv_i}{dw} \delta w \\ & \qquad \geq 1 - \xi_i^{+,t} - \delta\xi_i^+ \quad \forall y_i = 1 \\ & -(w^t + \delta w)^T B_{ij} \geq 1 - \xi_{ij}^{-,t} - \delta\xi_{ij}^- \quad \forall y_i = -1 \\ & \sum_j \frac{dv_{ij}}{dw} \delta w = 0, \quad v_i^t + \frac{dv_i}{dw} \delta w \geq 0, \quad \xi^t + \delta\xi \geq 0 \\ & \|\delta w\|_\infty \leq \mu, \quad \|\delta \xi\|_\infty \leq \mu \end{aligned} \tag{39}$$

Here $v^t = v(w^t)$ and $\frac{dv}{dw}$ is the derivative of v w.r.t. w at the current iteration v^t, which is obtained by solving Equation (37). The above objective and constraints are derived from the following linearization using first order Taylor expansion:

$$v(w^t + \delta w) \approx v(w^t) + \frac{dv}{dw}\delta w = v^t + \frac{dv}{dw}\delta w \tag{40}$$

$$\begin{aligned} & (w^t + \delta w)^T B_i v_i(w^t + \delta w) \approx \\ & \qquad (w^t + \delta w)^T B_i v_i^t + (w^t)^T B_i \frac{dv_i}{dw} \delta w \end{aligned} \tag{41}$$

Then the Wiberg minimization problem can be solved by iteratively applying linear programming to solve Equation (39) until convergence. Note that the new v^{t+1} can be estimated using Equation (40). It is worth pointing out that the new estimates can be an infeasible solution to the original problem in Equation (35), since the constraints are linearized using first order approximation. In other words, the new estimates might violate some of the constraints. A simple way to fix this problem is to reduce the trust region size and recalculate the solution so that the nonlinear constraints are better approximated. However, in our experiments, we found that this will dramatically hurt the convergence speed in most cases. The reason is that if some of the constraints are tight under the current estimates, the linear program will make very little progress during the iteration. In order to handle this problem, we propose to use the following strategy: if the new estimate is a feasible solution to the original problem, we will accept this solution and increase μ by a factor of 10. Otherwise, we will project the infeasible solution back to a feasible point by solving an exact linear program in Equation (37) to obtain a feasible v^{t+1} and ξ^{t+1} using the new estimate w^{t+1}. At the same time, μ will be decreased by a factor of 10 in the next Wiberg iteration. We adopt the same strategy for simultaneous minimization.

7.3 SLP with Stochastic Optimization

The size of linear program for Equation (39) grows linearly with the number of bags and instances. For example, in a problem of 1,000 bags with 10,000 instances, the linear program in each iteration contains more than 10,000 variables with more than 10,000 constraints, which is non-trivial to solve. Valuable research work has been proposed to handle similar large scale optimization problems. Inspired by Stochastic Gradient Descent (SGD) [34] for traditional learning frameworks, we propose to combine SLP with stochastic optimization to deal with large problems in MIL. The idea is that during each iteration, we only update the variables based on a subset of bags instead of using all bags, resulting in solving a much smaller linear program. In particular, assume at iteration t with current estimate w^t, we first randomly pick a set of k positive and negative bags $S_t \in B$. The corresponding indicating variables and derivatives of the selected positive bags, v_i^t and $\frac{dv_i^t}{dw}$, $i \in S_t$, are then obtained by solving Equation (37) only for the selected bags. Finally, we solve a modified version of Equation (39) by only dealing with the objective and constraints over the selected bags, i.e., $i \in S_t$. By doing this, the linear program size can be reduced from $O(n)$ to $O(k)$ during each iteration. The SLP with stochastic optimization method can be similarly applied to simultaneous minimization.

7.4 Analysis

Wiberg minimization effectively removes v from the optimization, which at first blush promises to be faster than minimizing with respect to w and v simultaneously. There are two remarks we want to point out. First, during each iteration of Wiberg and simultaneous minimization, we both solve a linear program (formed from Equations (39) and (36) respectively) to obtain new estimates. However, an additional calculation of the derivative matrix, $\frac{dv}{dw}$, is involved in Wiberg method, which increases the running time for each Wiberg iteration. We will discuss more in the experiments. Second, from Equation (35) we can see that an alternative Wiberg minimization scheme is to eliminate w from the optimization instead of v. In this case, we have to calculate the derivative matrix, $\frac{dw}{dv}$, from the following linear program:

$$\begin{aligned} \min_{w,\xi} \quad & \|w\|_1 + \lambda_1 \sum_i \xi_i^+ + \lambda_2 \sum_{ij} \xi_{ij}^- \\ s.t. \quad & w^T B_i v_i \geq 1 - \xi_i^+ \qquad \forall y_i = 1 \\ \xi \geq 0, \ & -w^T B_{ij} \geq 1 - \xi_{ij}^- \qquad \forall y_i = -1 \end{aligned} \tag{42}$$

However, the above linear program cannot be divided into independent small problems as we did for Equation (37), which will take much more time for computing the derivative matrix under the assumption that w and v have about the same dimensions. Actually, choosing which variable to eliminate depends on the dimensionality of the variables and also data sparsity. In our MIL problem, the dimensionality of v is the total number of instances in positive bags, which is usually much larger than the dimension of w (feature dimension), especially for large scale data. Therefore we choose to remove v instead of w in Wiberg minimization.

8 Generalized Constrained Wiberg Minimization

In this section, we generalize our approach to solve a group of nonlinear constrained optimization problems of two sets of variables under certain assumptions. We shall see that multiple instance learning is one specific problem that belongs to this group.

We write the generalized constrained optimization problem as follow:

$$\begin{aligned} \min_{w,v} \quad & f(w,v) \\ s.t. \quad & g_i(w,v) \leq 0, \ 1 \leq i \leq n \end{aligned} \tag{43}$$

here f is the objective function to be minimized and g_i is the i-th constraint that satisfy the following differentiable partial convex assumptions:

Assumption 1: The functions f and g_i are continuous, twice differentiable and strongly convex in v.

Assumption 2: Given any w, both the primal and dual solutions to Equation (43) are unique, denoted as v^* and λ^* and Slater's condition [35] holds in an open neighborhood of w.

Under the above assumptions, we first state the existence of the derivatives $\frac{dv^*}{dw}$ in the following theorem:

Theorem 8.1: If assumptions 1 and 2 hold, then v is differentiable with respect to w at v^*, denoted as $\frac{dv^*}{dw}$. The proof of the above theorem as well as the specific form of $\frac{dv^*}{dw}$ are given in Appendix B in the supplementary material.

After obtaining this unique derivatives $\frac{dv^*}{dw}$, we can derive the generalized constrained Wiberg approach to problem (43) by removing v from the optimization, and representing v as a implicit function of w. Then we iteratively solve the constrained optimization problem by linearizing both the objective and constraints during each iteration t:

$$\begin{aligned} \min_{\delta w} \quad & \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial v}\frac{dv}{dw}\delta w \\ s.t. \quad & \frac{\partial g_i}{\partial w}\delta w + \frac{\partial g_i}{\partial v}\frac{dv}{dw}\delta w \leq -g_i(w^t, v(w^t)), \ 1 \leq i \leq n \\ & \|\delta w\|_\infty \leq \mu \end{aligned} \tag{44}$$

Here $v(w^t) = v^{t*}$ is the unique optimal solution given w^t which is provided by Assumption 2. $\frac{dv}{dw}$ is the derivative of v with respect to w at v^{t*}. SLP with an adaptive trust region can be applied to iteratively solve this optimization problem. From the above discussion, it is clear that MIL in Equation (35) is a special problem that lies in this group. Note that we can alternatively remove w from the optimization if the above assumptions also hold for w. As discussed in section 4.4, removing which variable in the Wiberg minimization really depends on the problem, i.e., structure of the problem and dimensionality of its variables.

9 Results

Our experiments examine general Wiberg for bundle adjustment, nested Wiberg for projective bundle adjustment, and constrained Wiberg for multiple instance learning.

As Section 3.5 describes, each of our L_1 Wiberg algorithms modifies an existing algorithm: minimizing with respect to all of the unknowns simultaneously using successive linear programming. So, our experiments focus on isolating Wiberg's effect by comparing our algorithms against successive linear programming. In addition, Eriksson and van den Hengel's original factorization paper[4] compared their algorithm against Ke and Kanade's alternating algorithm[6]. But, successive linear programming is a better baseline for comparison, so we also include a new comparison of [4] to successive linear programming.

The focus of our experiments is on isolating the difference between Wiberg and the successive linear programming approach it modifies. But, L_1-norm minimization is generally attractive because it is more robust to outliers than least squares. So, we also include experiments comparing L_1 Wiberg to an existing robust algorithm, Huber norm minimization, for factorization and bundle adjustment; and to least squares for bundle adjustment.

Our implementation uses Clp (COIN-OR linear programming) for linear programming.

In this section, we briefly summarize our experiments and results. The supplementary material describes the experiments in detail.

9.1 L_1 Wiberg versus Successive Linear Programming

In our experiments, Wiberg and successive linear programming each won some rounds. Often Wiberg converges faster or more reliably than successive linear programming. Wiberg iterations are faster below some problem size while successive linear programming iterations are faster above that size.

Bundle adjustment. We compared general Wiberg against successive linear programming using synthetic cameras and points. For gross errors in the initial estimates, Wiberg usually converged to a lower residual than simultaneous, but not always to the ground truth residual. Wiberg iterations were faster for problems with 4 or less images, while successive linear programming iterations were faster for more than 4 images. Iteration times across our problem sizes were 5.7×10^{-4} (2 images, 10 points) to 11.1 seconds (7 images, 316 points) per iteration for Wiberg, and 1.9×10^{-3} (2 images, 10 points) to 1.6×10^{2} seconds (11 images, 1000 points) for successive linear programming. However, these Wiberg times exclude large problem sizes that didn't complete; the graphs in Section C.2 in the supplementary material tell the full story.

We also include an L_1 Wiberg bundling result for a real image sequence with about 700 images and about 10,000 points (Figure 1). This result relies on the early termination heuristic in Section 4.3, and like all of our other results, the scalable Wiberg implementation described in Section 3.6. Using 10 minutes per iteration, the method reduced an initial average residual per observation from 31.4 pixels to 1.89 pxiels in 15 iterations.

Section C in the supplementary material details these experiments.

Projective bundle adjustment. We compared successive linear programming (algorithm 1 in Section 5.1) and nested Wiberg (Section 5.1's algorithm 2) using synthetic cameras and points, with each method winning some rounds. For instance, Wiberg converged more reliably for wider ranges of projective depths, but simultaneous produced smaller residuals for the largest gross errors in the initial estimates. Wiberg iterations were faster for problems with 51 points or less, while successive linear programming iterations were faster for larger problems. Iteration times varied from 8.8×10^{-3} (10 images, 7 points) to 1.6×10^{4} (50 images, 100 points) seconds per iteration for Wiberg, and 2.0×10^{-2} (10 images, 7 points) to 1.0×10^{4} seconds (50 images, 100 points) per iteration for successive linear programming.

We also compared our nested Wiberg projective bundle adjustment (algorithm 2), which minimizes with respect to cameras, points, and projective depths, against a non-nested Wiberg (Section 5.1's algorithm 3) that minimizes with respect to just cameras and points. The accuracy of the two algorithms is quite close, with non-nested Wiberg having a slight edge in final residuals, but nested Wiberg having a slight edge in number of iterations required to reach the final residual.

Section D in the supplementary material describes these experiments in detail.

Multiple instance learning. We compared constrained Wiberg and simultaneous minimization on several MIL datasets. We found that Wiberg converges faster than the simultaneous method. Wiberg finds the same final objective in fewer iterations than the simultaneous method. For example, in the Tiger dataset, the simultaneous method takes around 30 iterations (2.0 seconds) to converge in average, while Wiberg converges in 10 iterations (1.1 seconds). Moreover, Wiberg usually converges to a local minimal with smaller objective than the simultaneous method.

Section E in the supplementary material describes these experiments.

Matrix factorization. Eriksson and van den Hengel's original paper compared L_1 Wiberg matrix factorization against Ke and Kanade's alternating algorithm[6], but successive linear programming is a more relevant baseline. Wiberg has a slight advantage in final residuals and a larger advantage in iterations until convergence. Wiberg had faster iterations for matrices with 15 rows or less, and successive linear programming iterations were faster for matrices with 23 rows or more. Across our problem sizes, successive linear programming times varied from 2.2×10^{-3} (6 rows, 10 columns) to 1.1×10^{4} seconds (54 rows, 1000 columns) per iteration, and Wiberg times varied from 2.5×10^{-3} (6 rows, 10 columns) to 1.1×10^{3} (54 rows, 316 columns). But like our bundle adjustment experiments above, these Wiberg times exclude larger problem sizes that didn't complete; the graphs in Section F tell the timing story in full.

Section F in the supplementary material describes these experiments in detail.

9.2 L_1 Wiberg versus Huber Norm Minimization

L_1 minimization is appealing because of its robustness to outliers relative to least squares. So, in additional experiments, we also compared L_1 to least squares minimization and to Huber norm minimization, which both have faster iteration times than our linear-programming-based approaches, Wiberg and successive linear programming.

Bundle adjustment. In experiments with gross outliers, L_1 tolerated outliers much better than least squares, but Huber tolerated a larger fraction of outliers than L_1 before producing gross errors in the estimates. Huber norm minimization was also faster in our experiments, requiring just 1.7 seconds per linear system solve for our largest problem, versus 1.6×10^{2} seconds per linear program solve for successive linear programming. Sections C.3 and C.4 in the supplementary material describe these experiments.

Matrix factorization. Wiberg produced slightly better residuals than Huber norm minimization in our experiments, which were similar to Eriksson and van den Hengel's original experiments comparing Wiberg to EM. But Huber norm minimization was hugely faster: our slowest L_1 Wiberg problem required 1.1×10^{3} seconds per linear program solve, whereas Huber norm minimization required just 3.4×10^{-2} seconds per linear

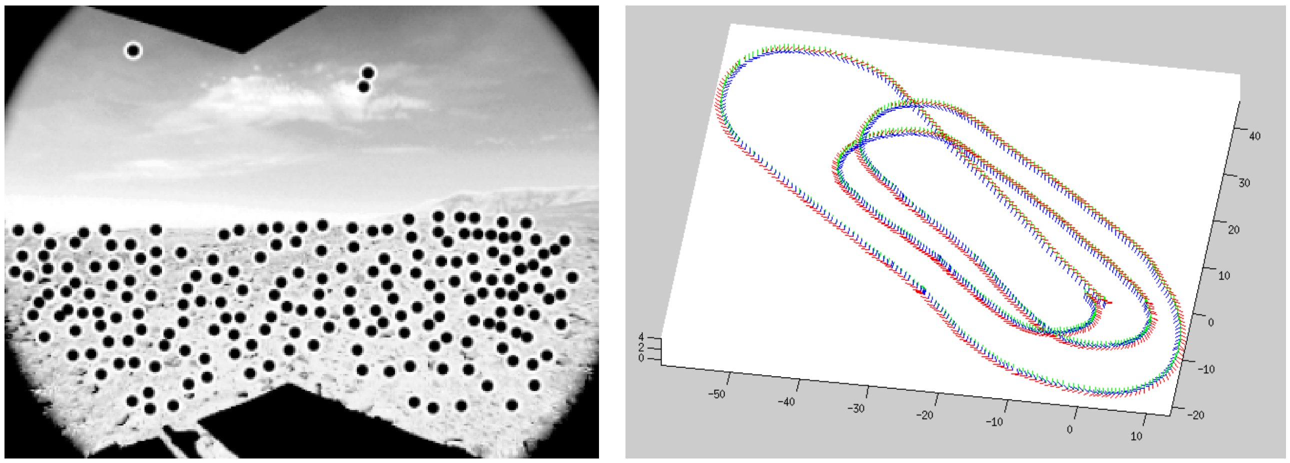

Fig. 1. L_1-Wiberg bundle adjustment reconstructs the correct structure and motion from the "rover" sequence, which includes about 700 images and 10,000 points. Left: an example image from the sequence, with tracked points shown as black dots. Right: an oblique view of the recovered camera positions at the time of each image.

system solve for the same problem. Section F.2 in the supplementary material describes these experiments in detail.

10 Conclusion

We've introduced general, nested, and constrained 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, showing that L_1 bundle adjustment and projective bundle adjustment can be implemented as general and nested Wiberg minimizations, respectively; and showing that multiple instance learning can be attacked as a constrained Wiberg minimization. We also introduced successive linear programming algorithms for these problems, which estimate all of the unknowns simultaneously.

For L_1 matrix factorization and bundle adjustment, 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 3.7 explains, the simultaneous approach produces a sparser linear 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.

For multiple instance learning, this paper presents an efficient and effective optimization method named Constrained Wiberg Minimization for Multiple Instance Learning (CWM-MIL). The proposed method eliminates the indicating variables from the optimization and iteratively optimizes with respect to the classifier variables only. The resulting constrained Wiberg MIL problem is solved by successive linear programming with stochastic optimization. A simultaneous minimization method is also proposed as a strong baseline. Furthermore, we generalize our Wiberg approach to general constrained optimization problem with two sets of variables. Experimental results demonstrate the superior convergence speed of the proposed CWM-MIL approach. In the future, we plan to (1) design a more efficient method to calculate the derivative matrix to further decrease the Wiberg iteration time; (2) extend this method to the multi-label case; (3) derive the theoretical error bound for the general Wiberg method.

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.

We also want to thank the anonymous reviewers, who went far beyond the call of duty, for their time and expertise; and the reviewers of an earlier version for suggesting the experimental comparison between L_1 and least squares bundle adjustment and pointing out the Huber norm.

Dr. Eriksson's research was supported by the Australian Research Council through the project DE130101775.

References

[1] T. Wiberg, "Computation of principal components when data are missing," in *Second Symposium of Computation Statistics*, Berlin, 1976, pp. 229–326.

[2] T. Okatani and K. Deguchi, "On the Wiberg algorithm for matrix factorization in the presence of missing components," *International Journal of Computer Vision*, vol. 72, no. 3, May 2007.

[3] 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.

[4] 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.

[5] ——, "Efficient computation of robust weighted low-rank matrix approximations using the L_1 norm," *TPAMI*, vol. 34, no. 9, pp. 1681–1690, 2012.

[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] A. Ruhe and P. Wedin, "Algorithms for separable nonlinear least squares problems," *SIAM Review*, vol. 22, no. 3, pp. 318–337, 1980.

[8] F. Richards, "A method of maximum-likelihood estimation," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 23, no. 2, pp. 469–475, 1961.

[9] D. Strelow, "General and nested Wiberg minimization: L_2 and maximum likelihood," in *ECCV*, Florence, Italy, 2012.

[10] C. Poelman, "The paraperspective and projective factorization methods for recovering shape and motion," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania, July 1995.

[11] M. S. Bazaraa, H. D. Sherali, and C. Shetty, *Nonlinear programming: theory and algorithms*, 3rd ed. Hoboken, New Jersey: Wiley, 2006.

[12] Y. Liu, B. Liu, Y. Pu, X. Chen, and H. Cheng, "Low-rank matrix decomposition in l_1-norm by dynamic systems," *Image and Vision Computing*, vol. 30, pp. 915–921, November 2012.

[13] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi, "Practical low-rank matrix approximation under robust l_1-norm," in *Computer Vision and Pattern Recognition*, Portland, Oregon, June 2013.

[14] Y.-X. Wang, C. M. Lee, L.-F. Cheong, and K.-C. Toh, "Practical matrix completion and corruption recovery using proximal alternating robust subspace minimization," *International Journal of Computer Vision*, vol. 111, pp. 315–344, February 2015.

[15] D. Strelow, "General and nested Wiberg minimization," in *CVPR*, Providence, Rhode Island, 2012.

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

[17] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[18] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Tech. Rep., 2006.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003.

[20] J. Oliensis and R. Hartley, "Iterative extensions of the Sturm/Triggs algorithm: convergence and nonconvergence," *IEEE PAMI*, vol. 29, no. 12, pp. 2217–2233, December 2007.

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

[22] C. Bergeron, G. M. Moore, J. Zaretzki, C. M. Breneman, and K. P. Bennett, "Fast bundle algorithm for multiple-instance learning," *TPAMI*, vol. 34, no. 6, pp. 1068–1079, 2012.

[23] Y. Hu, M. Li, and N. Yu, "Multiple-instance ranking: Learning to rank images for image retrieval," in *CVPR*, 2008.

[24] C. Yang, M. Dong, and J. Hua, "Region-based image annotation using asymmetrical support vector machine-based multiple-instance learning," in *CVPR (2)*, 2006, pp. 2057–2063.

[25] Q. Wang, L. Ruan, and L. Si, "Adaptive knowledge transfer for multiple instance learning in image classification," in *AAAI*, 2014, pp. 1334–1340.

[26] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *NIPS*, 2002, pp. 561–568.

[27] Q. Wang, L. Si, and D. Zhang, "A discriminative data-dependent mixture-model approach for multiple instance learning in image classification," in *ECCV (4)*, 2012, pp. 660–673.

[28] D. Zhang, J. He, L. Si, and R. Lawrence, "Mileage: Multiple instance learning with global embedding," in *ICML*, 2013.

[29] O. L. Mangasarian and E. W. Wild, "Multiple instance classification via successive linear programming," *Journal of Optimization Theory and Applications*, vol. 137, no. 3, pp. 555–568, 2008.

[30] Y. Chen, J. Bi, and J. Z. Wang, "Miles: Multiple-instance learning via embedded instance selection," *TPAMI*, vol. 28, no. 12, pp. 1931–1947, 2006.

[31] Z. Fu and A. Robles-Kelly, "An instance selection approach to multiple instance learning," in *CVPR*, 2009, pp. 911–918.

[32] D. Zhang, J. He, and R. D. Lawrence, "M2ls: Multi-instance learning from multiple information sources," in *KDD*, 2013.

[33] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms (3. ed.)*. Wiley, 2006.

[34] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.

[35] J. M. Borwein and A. S. Lewis, "Convex analysis and nonlinear optimization, theory and examples," 2000.

Dennis Strelow received a B.S. from the University of Wisconsin in 1994, an M.S. from the University of Illinois in 1996, and a Ph.D. from Carnegie Mellon in 2004, all in computer science. He is an engineer in Google's machine perception group, and his research interests are in computer vision and optimization.

Qifan Wang received both his B.S. and M.S. from Tsinghua University, and Ph.D. from Purdue University, all in Computer Science. He is an engineer in Google's machine intelligence group, and his research interests include machine learning, information retrieval, data mining and computer vision.

Luo Si is a faculty member in the Department of Computer Science, Department of Statistics (by courtesy), Purdue University. Luo Si leads a research group working on topics of information retrieval, applied machine learning, and intelligent tutoring. He has published more than 90 journal and conference papers. His research has been supported by National Science Foundation, State of Indiana, Purdue University and industry companies such as Yahoo! and Google. He obtained NSF career award in 2008. He is an associate editor of ACM Transactions on Information System (TOIS) and ACM Transactions on Interactive Information Systems.

Anders Eriksson is an Australian Research Council Fellow and a Senior Research Associate at the School of Electrical Engineering and Computer Science, Queensland University of Technology. He received his Masters of Science degree in Electrical Engineering in 2000 and his PhD in Mathematics in 2008 from Lund University, Sweden. His research areas include optimisation theory and numerical methods applied to the fields of computer vision and machine learning. In 2010 his work on robust low-rank matrix approximation won the best paper award at the 23rd IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA.