Practical Global Optimization for Multiview Geometry

Sameer Agarwal[1], Manmohan Krishna Chandraker[1], Fredrik Kahl[2], David Kriegman[1], and Serge Belongie[1]

[1] University of California, San Diego, CA 92093, USA,
{sagarwal, mkchandraker, kriegman, sjb}@cs.ucsd.edu
[2] Lund University, Lund, Sweden
fredrik@maths.lth.se

Abstract. This paper presents a practical method for finding the provably globally optimal solution to numerous problems in projective geometry including multiview triangulation, camera resectioning and homography estimation. Unlike traditional methods which may get trapped in local minima due to the non-convex nature of these problems, this approach provides a theoretical guarantee of global optimality. The formulation relies on recent developments in fractional programming and the theory of convex underestimators and allows a unified framework for minimizing the standard L_2-norm of reprojection errors which is optimal under Gaussian noise as well as the more robust L_1-norm which is less sensitive to outliers. The efficacy of our algorithm is empirically demonstrated by good performance on experiments for both synthetic and real data. An open source MATLAB toolbox that implements the algorithm is also made available to facilitate further research.

1 Introduction

Projective geometry is one of the success stories of computer vision. Methods for recovering the three dimensional structure of a scene from multiple images and the projective transformations that relate the scene and its images are now the workhorse subroutines in applications ranging from specialized tasks like matchmove in filmmaking to consumer products like image mosaicing for digital camera users.

The key step in each of these methods is the solution of an appropriately formulated optimization problem. These optimization problems are typically highly non-linear and finding their global optima in general has been shown to be NP-hard [1]. Methods for solving these problems are based on a combination of heuristic initialization and local optimization to converge to a *locally* optimal solution. A common method for finding the initial solution is to use a direct linear transform (for example, the eight-point algorithm [2]) to convert the optimization problem into a linear least squares problem. The solution then serves as the initial point for a non-linear minimization method based on the Jacobian and Hessian of the objective function, for instance, bundle adjustment. As has been documented, the success of these methods critically depends on the quality of the initial estimate [3].

In this paper we present the first practical algorithm for finding the globally optimal solution to a variety of problems in multiview geometry. The problems we address include general n-view triangulation, camera resectioning (also called cameras pose or absolute orientation) and the estimation of general projections $\mathbb{P}^n \mapsto \mathbb{P}^m$, for $n \geq m$. We solve each of these problems under three different noise models, including the standard Gaussian distribution and two variants of the bi-variate Laplace distribution. Our algorithm is provably optimal, that is, given any tolerance ϵ, if the optimization problem is feasible, the algorithm returns a solution which is at most ϵ far from the global optimum. The algorithm is a branch and bound style method based on extensions to recent developments in the fractional and convex programming literature [4–6]. While the worst case complexity of our algorithm is exponential, we will show in our experiments that for a fixed ϵ the runtime of our algorithm scales almost linearly with problem size, making this a very attractive approach for use in practice.

Recently there has been some progress made towards finding the global solution to a *few* of these optimization problems. An attempt to generalize the optimal solution of two-view triangulation [7] to three views was done in [8] based on Gröbner basis. However, the resulting algorithm is numerically unstable, computationally expensive and does not generalize for more views or harder problems like resectioning. In [9], linear matrix inequalities were used to approximate the global optimum, but no guarantee of actually obtaining the global optimum is given. Also, there are unsolved problems concerning numerical stability. Robustification using the L_1-norm was presented in [10], but the approach is restricted to the affine camera model. In [11], a wider class of geometric reconstruction problems was solved globally, but with L_∞-norm.

In summary, our main contributions are:

- A scalable algorithm for solving a class of multiview problems with a guarantee of global optimality.
- In addition to using the standard L_2-norm of reprojection errors, we are able to handle the robust L_1-norm for the perspective camera model.
- Introduction of fractional programming to the computer vision community.

We begin with an exposition on fractional programming in the next section along with an introduction to branch and bound algorithms. We describe in detail the construction of the lower bounds and present our initialization methods along with a novel bounds propagation scheme. This scheme exploits the special properties of structure and motion problems to restrict the branching process to a small, fixed number of dimensions independent of the problem size. Finally, we demonstrate that various structure and motion problems can indeed be formulated as fractional programs of the type we deal with and present the results of our experiments.

2 Fractional Programming

In its most general form, fractional programming seeks to minimize/maximize the sum of $p \geq 1$ fractions subject to convex constraints. Our interest from the point of view of multiview geometry, however, is specific to the minimization problem

$$\min_{x} \quad \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \quad \text{subject to} \qquad x \in D \tag{F1}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex and concave functions, respectively, and the domain $D \subset \mathbb{R}^n$ is a convex, compact set. Further, it is assumed that both f_i and g_i are positive with lower and upper bounds over D. Even with these restrictions the above problem is NP-complete [1], but we demonstrate that practical and reliable estimation of the global optimum is still possible for the multiview problems considered.

Let us assume that we have available to us upper and lower bounds on the functions $f_i(x)$ and $g_i(x)$, denoted by the intervals $[\,l_i, u_i\,]$ and $[\,L_i, U_i\,]$, respectively. Let Q_0 denote the $2p$-dimensional rectangle $[\,l_1, u_1\,] \times \cdots \times [\,l_p, u_p\,] \times [\,L_1, U_1\,] \times \cdots \times [\,L_p, U_p\,]$. Introducing auxiliary variables $t = (t_1, \ldots, t_p)^\top$ and $s = (s_1, \ldots, s_p)^\top$, consider the following alternate optimization problem:

$$\begin{array}{llll} & \displaystyle\min_{x,t,s} \quad \sum_{i=1}^{p} \frac{t_i}{s_i} & & \\ \text{subject to} & f_i(x) \leq t_i & g_i(x) \geq s_i & \\ & \qquad x \in D & (t,s) \in Q_0. & \end{array} \tag{F2}$$

We note that the feasible set for problem (F2) is a convex, compact set and that (F2) is feasible if and only if (F1) is. Indeed the following holds true [5]:

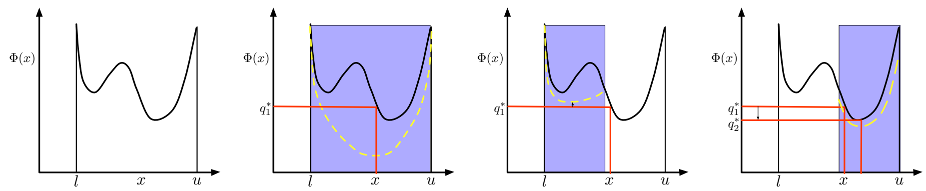
Theorem 1. *$(x^*, t^*, s^*) \in \mathbb{R}^{n+2p}$ is a global, optimal solution for* (F2) *if and only if $t_i^* = f_i(x^*)$, $s_i^* = g_i(x^*)$, $\quad i = 1, \cdots, p$ and $x^* \in \mathbb{R}^n$ is a global optimal solution for* (F1).

Thus, Problems (F1) and (F2) are equivalent, and henceforth we shall restrict our attention to Problem (F2).

2.1 Branch and Bound Theory

Branch and bound algorithms are non-heuristic methods for global optimization in non-convex problems. They maintain a provable upper and/or lower bound on the (globally) optimal objective value and terminate with a certificate proving that the solution is ϵ-suboptimal (that is, within ϵ of the global optimum), for arbitrarily small ϵ.

Consider a non-convex, scalar-valued objective function $\Phi(x)$, for which we seek a global optimum over a rectangle Q_0 as in Problem (F2). For a rectangle $Q \subseteq Q_0$, let $\Phi_{\min}(Q)$ denote the minimum value of the function Φ over Q. Also, let $\Phi_{\text{lb}}(Q)$ be a function that satisfies the following conditions:



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

Fig. 1. This figure illustrates the operation of a branch and bound algorithm on a one dimensional non-convex minimization problem. Figure (a) shows the the function $\Phi(x)$ and the interval $l \leq x \leq u$ in which it is to be minimized. Figure (b) shows the convex relaxation of $\Phi(x)$ (indicated in yellow/dashed), its domain (indicated in blue/shaded) and the point for which it attains a minimum value. q_1^* is the corresponding value of the function Φ. This value is the best estimate of the minimum of $\Phi(x)$ is used to reject the left subinterval in Figure (c) as the minimum value of the convex relaxation is higher than q_1^*. Figure (d) shows the lower bounding operation in the right sub-interval in which a new estimate q_2^* of the minimum value of $\Phi(x)$.

(L1) $\Phi_{\text{lb}}(Q)$ computes a lower bound on $\Phi_{\min}(Q)$ over the domain Q, that is, $\Phi_{\text{lb}}(Q) \leq \Phi_{\min}(Q)$.

(L2) The approximation gap $\Phi_{\min}(Q) - \Phi_{\text{lb}}(Q)$ uniformly converges to zero as the maximum half-length of sides of Q, denoted $|Q|$, tends to zero, that is

$$\forall\, \epsilon > 0,\ \exists\, \delta > 0 \text{ s.t. } \forall Q \subseteq Q_0,\ |Q| \leq \delta \Rightarrow \Phi_{\min}(Q) - \Phi_{\text{lb}}(Q) \leq \epsilon.$$

The branch and bound algorithm begins by computing $\Phi_{\text{lb}}(Q_0)$ and the point $q^* \in Q_0$ which minimizes $\Phi_{\text{lb}}(Q_0)$. If $\Phi(q^*) - \Phi_{\text{lb}}(Q_0) < \epsilon$, the algorithm terminates. Otherwise Q_0 is partitioned as a union of subrectangles $Q_0 = Q_1 \cup \cdots Q_k$ for some $k \geq 2$ and the lower bounds $\Phi_{\text{lb}}(Q_i)$ as well as points q_i (at which these lower bounds are attained) are computed for each Q_i. Let $q^* = \arg\min_{\{q_i\}_{i=1}^k} \Phi(q_i)$. We deem $\Phi(q^*)$ to be the current best estimate of $\Phi_{\min}(Q_0)$. The algorithm terminates when $\Phi(q^*) - \min_{1 \leq i \leq k} \Phi_{\text{lb}}(Q_i) < \epsilon$, else the partition of Q_0 is refined by further dividing some subrectangle and repeating the above. The rectangles Q_i for which $\Phi_{\text{lb}}(Q_i) > \Phi(q^*)$ cannot contain the global minimum and are not considered for further refinement. A graphical illustration of the algorithm is presented in Figure 1.

Computation of the lower bounding functions is referred to as *bounding*, while the procedure that chooses a rectangle and subdivides it is called *branching*. The choice of the rectangle picked for refinement in the branching step and the actual subdivision itself are essentially heuristic. We consider the rectangle with the smallest minimum of Φ_{lb} as the most promising to contain the global minimum and subdivide it into $k = 2$ rectangles. Algorithm 1 uses the abovementioned functions to present a concise pseudocode for the branch and bound method.

Although guaranteed to find the global optimum (or a point arbitrarily close to it), the worst case complexity of a branch and bound algorithm is exponential.

However, we will show in our experiments that the special properties offered by multiview problems lead to fast convergence rates in practice.

Algorithm 1 Branch and Bound

Require: Initial rectangle Q_0 and $\epsilon > 0$.

Bound : Compute $\Phi_{\text{lb}}(Q_0)$ and minimizer $q^* \in Q_0$.
$S = \{Q_0\}$ //Initialize the set of candidate rectangles
loop
  $Q' = \arg\min_{Q \in S} \Phi_{\text{lb}}(Q)$ //Choose rectangle with lowest bound
  if $\Phi(q^*) - \Phi_{\text{lb}}(Q') < \epsilon$ **then**
    return q^* //Termination condition satsified
  end if
  Branch : $Q' = Q_l \cup Q_r$
  $S = (S/\{Q'\}) \cup \{Q_l, Q_r\}$ //Update the set of candidate rectangles
  Bound : Compute $\Phi_{\text{lb}}(Q_l)$ and minimizer $q_l \in Q_l$.
  if $\Phi(q_l) < \Phi(q^*)$ **then**
    $q^* = q_l$ //Update the best feasible solution
  end if
  Bound : Compute $\Phi_{\text{lb}}(Q_r)$ and minimizer $q_r \in Q_r$.
  if $\Phi(q_r) < \Phi(q^*)$ **then**
    $q^* = q_r$ //Update the best feasible solution
  end if
  $S = \{\, Q \,|\, Q \in S, \Phi_{\text{lb}}(Q) < \Phi(q^*) \,\}$ //Discard rectangles with high lower bounds
end loop

2.2 Bounding

The goal of the **Bound** procedure is to provide the branch and bound algorithm with a bound on the smallest value the objective function takes in a domain. The computation of the function Φ_{lb} must possess three properties - crucial to the efficiency and convergence of the algorithm: (i) it must be easily computable, (ii) must provide as tight a bound as possible and (iii) must be easily minimizable. Precisely these features are inherent in the *convex envelope* of our objective function, which we define below.

Definition 1 (Convex Envelope). *Let $f : S \to \mathbb{R}$, where $S \subset \mathbb{R}^n$ is a non-empty convex set. The convex envelope of f over S (denoted* **convenv** *f) is a convex function such that (i)* **convenv** *$f(x) \leq f(x)$ for all $x \in S$ and (ii) for any other convex function u, satisfying $u(x) \leq f(x)$ for all $x \in S$, we have* **convenv** *$f(x) \geq u(x)$ for all $x \in S$.*

Finding the convex envelope of an arbitrary function may be as hard as finding the global minimum. To be of any advantage, the envelope construction must be cheaper than the optimal estimation.

In [4], it was shown that the convex envelope for a single fraction t/s, where $t \in [\,l, u\,]$ and $s \in [\,L, U\,]$, is given as the solution to the following *Second Order Cone Program* (SOCP):

$$\begin{array}{rll}
\text{minimize} & \rho & \\
\text{subject to} & \left\| \begin{matrix} 2\lambda\sqrt{l} \\ \rho' - s' \end{matrix} \right\| \leq \rho' + s' & \left\| \begin{matrix} 2(1-\lambda)\sqrt{u} \\ \rho - \rho' - s + s' \end{matrix} \right\| \leq \rho - \rho' + s - s' \\
& \lambda L \leq s \leq \lambda U & (1-\lambda)L \leq s - s' \leq (1-\lambda)U \\
& \rho' \geq 0 & \rho - \rho' \geq 0 \\
& l \leq t \leq u & L \leq s \leq U
\end{array}$$

where we have substituted $\lambda = \dfrac{u-t}{u-l}$ for ease of notation, and ρ, ρ', s' are auxiliary scalar variables.

It is easy to show that the convex envelop of a sum is always greater (or equal) than the sum of convex envelopes. That is, if $f = \sum_i t_i/s_i$ then $\textbf{convenv}\, f \geq \sum_i \textbf{convenv}\, t_i/s_i$. It follows that in order to compute a lower bound on Problem (F2), one can compute the sum of convex envelopes for t_i/s_i subject to the convex constraints. Hence, this way of computing a lower bound $\Phi_{\text{lb}}(Q)$ amounts to solving a convex SOCP problem which can be done efficiently [12]. It can be shown [5] that the convex envelope satisfies conditions (L1) and (L2), and therefore, is well-suited for our branch and bound algorithm.

2.3 Branching

Branch and bound algorithms can be slow, in fact, the worst case complexity grows exponentially with problem size. Thus, one must devise a sufficiently sophisticated branching strategy to expedite the convergence.

A general branching strategy applicable to fractional programs [5] is to branch along p dimensions corresponding to the denominators s_i of each fractional term t_i/s_i in Problem (F2). This limits the practical applicability to problems containing 10-12 fractions [13]. However, we demonstrate in Section 4.1 that for our class of problems, it is possible to restrict the branching to a *small* and *fixed* number of dimensions regardless of the number of fractions, which substantially enhances the number of fractions our algorithm can handle.

After a choice has been made of the rectangle to be further partitioned, there are two issues that must be addressed within the branching phase - namely, deciding the dimensions along *which* to split the rectangle and *where* along a chosen dimension to split the rectangle. We pick the dimension with the largest interval and employ a simple spatial division procedure, called α-bisection (see Algorithm 2) for a given scalar α, $0 < \alpha \leq 0.5$. It can be shown [5] that the α-bisection leads to a branch-and-bound algorithm which is convergent.

Algorithm 2 α-bisection

Require: A rectangle $Q \subset \mathbb{R}^{2p}$
1: $j = \arg\max_{i=1,\ldots,p}(U_i - L_i)$
2: $V_j = \alpha(U_j - L_j)$
3: $Q_l = [\,l_1, u_1\,] \times \cdots \times [\,l_p, u_p\,] \times [\,L_1, U_1\,] \times \cdots \times [\,L_j, V_j\,] \times \cdots \times [\,L_p, U_p\,]$
4: $Q_r = [\,l_1, u_1\,] \times \cdots \times [\,l_p, u_p\,] \times [\,L_1, U_1\,] \times \cdots \times [\,V_j, U_j\,] \times \cdots \times [\,L_p, U_p\,]$
5: **return** (Q_l, Q_r)

3 Applications to Multiview Geometry

In this section, we elaborate on adapting the theory developed in the previous section to common problems of multiview geometry. In the standard formulation of these problems based on the Maximum Likelihood Principle, the exact form of the objective function to be optimized depends on the choice of noise model. The noise model describes how the errors in the observations are statistically distributed given the ground truth.

In the Gaussian noise model, assuming an isotropic distribution of error with a known standard deviation σ, the likelihood for two image points - one measured point x and one true x' - is

$$p(x|x') = (2\pi\sigma^2)^{-1} \exp(-\|x - x'\|_2^2/(2\sigma^2)) \ . \tag{1}$$

Thus maximizing the likelihood of the observed point correspondences and assuming iid noise, is equivalent to minimizing $\sum_i \|x_i - x_i'\|_2^2$, which we interpret as a combination of two vector norms - the first for the point-wise error in the image and the second that cumulates these point-wise errors. We call this the (L_2, L_2)-formulation.

The exact definition of the Laplace noise model depends on the particular definition of the multivariate Laplace distribution [14]. In the current work we choose two of the simpler definitions. The first one is a special case of the multivariate exponential power distribution giving us the likelihood function:

$$p(x|x') = (2\pi\sigma)^{-1} \exp(-\|x - x'\|_2/\sigma) \ . \tag{2}$$

An alternative view of the bivariate Laplace distribution is to consider it as the joint distribution of two iid univariate Laplace random variables, where $x = (u, v)^\top$ and $x' = (u', v')^\top$ which gives us the following likelihood function

$$p(x|x') = \frac{1}{2\sigma} e^{-\frac{1}{\sigma}|u-u'|} \, \frac{1}{2\sigma} e^{-\frac{1}{\sigma}|v-v'|} \ = \ (4\sigma^2)^{-1} \exp(-\|x - x'\|_1/\sigma) \ . \tag{3}$$

Maximizing the likelihoods in (2) and (3) is equivalent to minimizing $\sum_i \|x_i - x_i'\|_2$ and $\sum_i \|x_i - x_i'\|_1$, respectively. Again, in our interpretation of these expressions as a combination of two vector norms, we denote these minimizations as (L_2, L_1) and (L_1, L_1), respectively.

We summarize the classification of overall error under various noise models in Table 1.

Gaussian	Laplacian I	Laplacian II
$\sum_i \|x_i - x'_i\|_2^2$	$\sum_i \|x_i - x'_i\|_2$	$\sum_i \|x_i - x'_i\|_1$
(L_2, L_2)	(L_2, L_1)	(L_1, L_1)

Table 1. Different cost-functions of reprojection errors.

3.1 Triangulation

The primary concern in triangulation is to recover the 3D scene point given measured image points and known camera matrices in $N \geq 2$ views. Let $P = [p_1\ p_2\ p_3]^\top$ denote the 3×4 camera where p_i is a 4-vector, $(u, v)^\top$ image coordinates, $X = (U, V, W, 1)^\top$ the extended 3D point coordinates, then the reprojection residual vector for this image is given by

$$r = \left(u - \frac{p_1^\top X}{p_3^\top X}, v - \frac{p_2^\top X}{p_3^\top X} \right)^\top \tag{4}$$

and hence the objective function to minimize becomes $\sum_{i=1}^N ||r_i||_p^q$ for the (L_p, L_q)-case. In addition, one can require that $p_3^\top X > 0$ which corresponds to the 3D point being in front of the camera. We now show that by defining $||r||_p^q$ as an appropriate ratio f/g of a convex function f and a concave function g, the problem in (4) can be identified with the one in (F2).

(L_2, L_2). The norm-squared residual of r can be written $||r||_2^2 = ((a^\top X)^2 + (b^\top X)^2)/(p_3^\top X)^2$ where a, b are 4-vectors dependent on the known image coordinates *and* the known camera matrix. By setting $f = ((a^\top X)^2 + (b^\top X)^2))/(p_3^\top X)$ and $g = p_3^\top X$, a convex-concave ratio is obtained. It is straightforward to verify the convexity of f via the convexity of its epigraph:

$$\begin{aligned} \mathbf{epi} f &= \{(X, t) \mid t \geq f(X)\} \\ &= \left\{ (X, t) \mid \frac{1}{2}(t + p_3^\top X) \geq \left\| \left(a^\top X,\, b^\top X,\, \frac{1}{2}(t - p_3^\top X) \right) \right\| \right\}, \end{aligned}$$

which is a second-order convex cone [6].

(L_2, L_1). Similar to the (L_2, L_2)-case, the norm of r can be written $||r||_2 = f/g$ where $f = \sqrt{(a^\top X)^2 + (b^\top X)^2}$ and $g = p_3^\top X$. Again, the convexity of f can be established by noting that the epigraph $\mathbf{epi} f = \{(X, t) \mid t \geq \|(a^\top X,\, b^\top X)\|\}$ is a second-order cone.

(L_1, L_1). Using the same notation as above, the L_1-norm of r is given by $||r||_1 = f/g$ where $f = |a^\top X| + |b^\top X|$ and $g = p_3^\top X$.

In all the cases above, g is trivially concave since it is linear in X.

3.2 Camera Resectioning

The problem of camera resectioning is the analogous counterpart of triangulation whereby the aim is to recover the camera matrix given $N \geq 6$ scene points and their corresponding images. The main difference compared to the triangulation problem is that the number of degrees of freedom has increased from 3 to 11.

Let $p = \left(p_1^\top, p_2^\top, p_3^\top\right)^\top$ be a homogeneous 12-vector of the unknown elements in the camera matrix P. Now, the squared norm of the residual vector r in (4) can be rewritten in the form $||r||_2^2 = ((a^\top p)^2 + (b^\top p)^2)/(X^\top p_3)^2$, where a, b are 12-vectors determined by the coordinates of the image point x and the scene point X. Recalling the derivations for the (L_2, L_2)-case of triangulation, it follows that $||r||_2^2$ can be written as a fraction f/g with $f = ((a^\top p)^2 + (b^\top p)^2)/(X^\top p_3)$ which is convex and $g = X^\top p_3$ concave in accordance with Problem (F2). Similar derivations show that the same is true for camera resectioning with (L_2, L_1)-norm as well as (L_1, L_1)-norm.

3.3 Projections from $\mathbb{P}^n$ to $\mathbb{P}^m$

Our formulation for the camera resectioning problem is very general and not restricted by the dimensionality of the world or image points. Thus, it can be viewed as a special case of a $\mathbb{P}^n \mapsto \mathbb{P}^m$ projection with $n = 3$ and $m = 2$.

When $m = n$, the mapping is called a homography. Typical applications include homography estamation of planar scene points to the image plane, or inter-image homographies ($m = n = 2$) as well as the estimation of 3D homographies due to different coordinate systems ($m = n = 3$). For projections ($n > m$), camera resection is the most common application, but numerous other instances appear in the computer vision field [15].

4 Multiview Fractional Programming

4.1 Bounds Propagation

Consider a fractional program with p fractions. For all problems presented in Section 3, the denominator is a linear function in the unknowns. For example, in the case of triangulation, the unknown point coordinates $X = (U, V, W, 1)^\top$ are linear in $g_i(X) = p_{3i}^\top X$ for $i = 1, \dots, p$. Suppose $p > 3$ and bounds are given on three denominators, say g_1, g_2, g_3 which are not linearly dependent. These bounds then define a convex polytope in $\mathbb{R}^3$. This polytope constrains the possible values of U, V and W which in turn induce bounds on the other denominators $g_4, \dots, g_p$. The bounds can be obtained by solving a set of linear equations each time branching is performed.

Thus, it is sufficient to branch on three dimensions in the case of triangulation. Similarly, in the case of camera resectioning, the denominator has only three degrees of freedom and more generally, for projections $\mathbb{P}^n \mapsto \mathbb{P}^m$, the denominator has n degrees of freedom.

4.2 Coordinate System Independence

All three error norms (see Table 1) are independent of the coordinate system chosen for the scene (or source) points. In the image, one can translate and scale the points without effecting the norms. For all problem instances and all three error norms considered, the coordinate system can be chosen such that the first denominator g_1 is a constant equal to one. Thus, there is no need to approximate the first term in the cost-function with a convex envelope, since it is a convex function already.

4.3 Initialization

In the construction of the algorithm we assumed that initial bounds are available on the numerator and the denominator of each of the fractions. This initial rectangle Q_0 in $\mathbb{R}^{2p}$ is the starting point for the branch and bound algorithm.

Let γ be an upper bound on the reprojection error in pixels (specified by the user), then we can bound the denominators $g_i(x)$ by solving the following set of optimization problems:

$$\text{for } i = 1, \ldots, p, \qquad \begin{array}{l} \min g_i(x) \\ \dfrac{f_j(x)}{g_j(x)} \leq \gamma \end{array} \qquad \begin{array}{l} \max g_i(x) \\ \dfrac{f_j(x)}{g_j(x)} \leq \gamma \end{array} \qquad j = 1, \ldots, p.$$

Depending on the choice of error norm, the above optimization problems will be instances of linear or quadratic programming. We will call this γ-initialization. While tight bounds on the denominators are crucial for the performance of the overall algorithm, we have found that the bounds on the numerators are not. Therefore, we set the numerator bounds to preset values.

5 Experiments

Both triangulation and estimation of projections $\mathbb{P}^n \mapsto \mathbb{P}^m$ have been implemented for all three error norms in Table 1 in the Matlab environment using the convex solver SeDuMi [12] and the code is publicly available[3]. The optimization is based on the branch and bound procedure as described in Algorithm 1 and α-bisection (see Algorithm 2) with $\alpha = 0.5$. To compute the initial bounds, γ-initialization is used (see Section 4.3) with $\gamma = 15$ pixels for both real and synthetic data. The branch and bound terminates when the difference between the global optimum and the underestimator is less than $\epsilon = 0.05$. In all experiments, the Root Mean Squares (RMS) errors of the reprojection residuals are reported regardless of the computation method.

5.1 Synthetic Data

Our data is generated by creating random 3D points within the cube $[-1, 1]^3$ and then projecting to the images. The image coordinates are corrupted with iid Gaussian noise with different levels of variance. In all graphs, the average of 200 trials are plotted. In the first experiment, we employ a weak camera

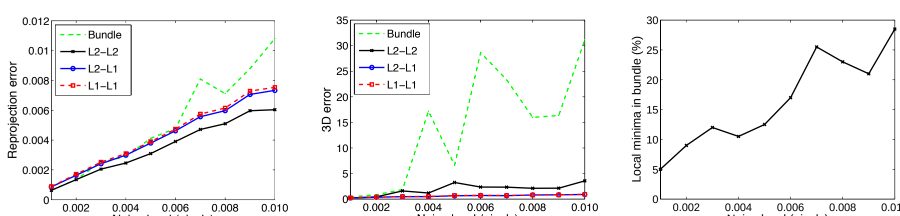
[3] See http://www.maths.lth.se/matematiklth/personal/fredrik/download.html.

geometry for triangulation, whereby three cameras are placed along a line at distances 5, 6 and 7 units, respectively, from the origin. In Figures 2(a) and (b), the reprojection errors and the 3D errors are plotted, respectively. The (L_2, L_2) method, on the average, results in a much lower error than bundle adjustment, which can be attributed to bundle adjustment being enmeshed in local minima due to the non-convexity of the problem. The graph in Figure 2(c) depicts the percentage number of times (L_2, L_2) outperforms bundle adjustment in accuracy. It is evident that higher the noise level, the more likely it is that the bundle adjustment method does not attain the global optimum.

In the next experiment, we simulate outliers in the data in the following manner. Varying numbers of cameras, placed 10^o apart and viewing toward the origin, are generated in a circular motion of radius 2 units. In addition to Gaussian noise with standard deviation 0.01 pixels for all image points, the coordinates for *one* of the image points have been perturbed by adding or subtracting 0.1 pixels. This point may be regarded as an outlier. As can seen from Figure 5.1(a) and (b), the reprojection errors are lowest for the (L_2, L_2) and bundle methods, as expected. However, in terms of 3D-error, the L_1 methods perform best and already from two cameras one gets a reasonable estimate of the scene point.

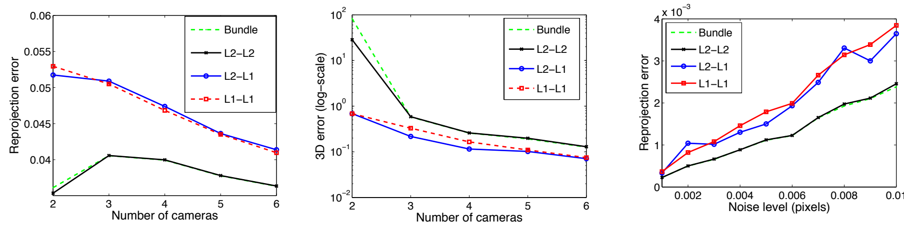
In the third experiment, six 3D points in general position are used to compute the camera matrix. Note that this is a minimal case, as it is not possible to compute the camera matrix from five points. The true camera location is at a distance of two units from the origin. The reprojection errors are graphed in Figure 5.1(c). Results for bundle adjustment and the (L_2, L_2) methods are identical and thus, likelihood of local minima is low.

To demonstrate scalability, Table 2 reports the runtime of our algorithm over a variety of problem sizes for resectioning. The tolerance, ϵ, here is set to within 1 percent of the global optimum, the maximum number of iterations to 500 and mean and median runtimes are reported over 200 trials. The algorithm's excellent runtime performance is demonstrated by almost linear scaling in runtimes.



(a) (b) (c)

Fig. 2. Triangulation with forward motion. The performance of bundle adjustment degrades rapidly with increasing noise, while our algorithm continues to perform well, both in terms of (a) reprojection error and (b) 3D error. The plot in (c) shows percentage number of times our algorithm outperforms bundle adjustment.



(a) (b) (c)

Fig. 3. (a) and (b) show reprojection and 3D erorrs, respectively, for triangulation with one outlier. Despite a higher reprojection error, the L_1-algorithms better bundle adjustment in terms of 3D error. (c) Reprojection errors for camera resectioning.

5.2 Real Data

We have evaluated the performance on two publicly available data sets as well - the dinosaur and the corridor sequences. In Table 3, the reprojection errors are given for (1) triangulation of all 3D points given pre-computed camera motion and (2) resection of cameras given pre-computed 3D points. Both the mean error and the estimated standard deviation are given. There is no difference between the bundle adjustment and the (L_2, L_2) method. Thus, for these particular sequences, the bundle adjustment did not get trapped in any local optimum. The L_1 methods also result in low reprojection errors as measured by the RMS criterion. More interesting is, perhaps, the number of iterations on a standard PC (3 GHz), see Table 4. In the case of triangulation, a point is typically visible in a couple of frames. The differences in iterations are most likely due to the setup: the dinosaur sequence has circular camera motion which is a better-posed geometry compared to forward motion in the corridor sequence.

Points	(L_2, L_2)			(L_2, L_1)			(L_1, L_1)		
	Mean	**Median**	**MI**	**Mean**	**Median**	**MI**	**Mean**	**Median**	**MI**
6	42.8	35.5	0.5	41.6	31.5	1.5	7.9	4.7	0.0
10	51.8	41.9	0.5	105.8	66.6	3.5	20.3	13.5	0.5
20	72.7	50.5	2.5	210.2	121.2	9.0	46.8	28.2	1.0
50	145.5	86.5	4.5	457.9	278.3	8.5	143.0	75.9	2.5
70	172.5	107.8	3.5	616.5	368.7	7.5	173.0	102.8	1.5
100	246.2	148.5	4.5	728.7	472.4	4.0	242.3	133.6	2.0

Table 2. Mean and median runtimes (in seconds) for the three algorithms as the number of points for a resectioning problem is increased. MI is the percentage number of times the algorithm reached 500 iterations.

Experiment	Bundle		(L_2, L_2)		(L_2, L_1)		(L_1, L_1)	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
Dino (triangulation)	0.30	0.14	0.30	0.14	0.18	0.09	0.22	0.11
Corridor (triangulation)	0.21	0.16	0.21	0.16	0.13	0.13	0.15	0.12
Dino (resection)	0.33	0.04	0.33	0.04	0.34	0.04	0.34	0.04
Corridor (resection)	0.28	0.05	0.28	0.05	0.28	0.05	0.28	0.05

Table 3. Reprojection errors (in pixels) for triangulation and resectioning in the Dinosaur and Corridor data sets. “Dinosaur” has 36 turntable images with 324 tracked points, while “Corridor” has 11 images in forward motion with a total of 737 points.

6 Discussions

In this paper, we have demonstrated that several problems in multiview geometry can be formulated within the unified framework of fractional programming, in a form amenable to global optimization. A branch and bound algorithm is proposed that provably finds a solution arbitrarily close to the global optimum, with a fast convergence rate in practice. Besides minimizing reprojection error under Gaussian noise, our framework allows incorporation of robust L_1 norms, reducing sensitivity to outliers. Two improvements that exploit the underlying problem structure and are critical for expiditious convergence are: branching in a small, constant number of dimensions and bounds propagation.

It is inevitable that our solution times be compared with those of bundle adjustment, but we must point out that it is producing a certificate of optimality that forms the most significant portion of our algorithm's runtime. In fact, it is our empirical observation that the optimal point ultimately reported by the branch and bound is usually obtained within the first few iterations.

A distinction must also be made between the accuracy of a solution and the optimality guarantee associated with it. An optimality criterion of, say $\epsilon = 0.95$, is only a worst case bound and does not necessarily mean a solution 5% away from optimal. Indeed, as evidenced by our experiments, our solutions consistently equal or better those of bundle adjustment in accuracy.

Experiment	(L_2, L_2)		(L_2, L_1)		(L_1, L_1)	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
Dino (triangulation)	1.2	1.5	1.0	0.2	6.7	3.4
Corridor (triangulation)	8.9	9.4	27.4	26.3	25.9	27.4
Dino (resection)	49.8	40.1	84.4	53.4	54.9	42.9
Corridor (resection)	39.9	2.9	49.2	20.6	47.9	7.9

Table 4. Number of branch and bound iterations for triangulation and resectioning on the Dinosaur and Corridor datasets. More parameters are estimated for resectioning, but the main reason for the difference in performance between triangulation and resectioning is that several hundred points are visible to each camera for the latter.

7 Acknowledgements

Sameer Agarwal and Serge Belongie are supported by NSF-CAREER #0448615, DOE/LLNL contract no. W-7405-ENG-48 (subcontracts B542001 and B547328), and the Alfred P. Sloan Fellowship. Manmohan Chandraker and David Kriegman are supported by NSF EIA 0303622 & NSF IIS-0308185. Fredrik Kahl is supported by Swedish Research Council (VR 2004-4579) & European Commission (Grant 011838, SMERobot).

References

1. Freund, R.W., Jarre, F.: Solving the sum-of-ratios problem by an interior-point method. J. Glob. Opt. **19** (2001) 83–102
2. Longuet-Higgins, H.: A computer algorithm for reconstructing a scene from two projections. Nature **vol.293** (1981) 133–135
3. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004) Second Edition.
4. Tawarmalani, M., Sahinidis, N.V.: Semidefinite relaxations of fractional programs via novel convexification techniques. J. Glob. Opt. **20** (2001) 137–158
5. Benson, H.P.: Using concave envelopes to globally solve the nonlinear sum of ratios problem. J. Glob. Opt. **22** (2002) 343–364
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
7. Hartley, R., Sturm, P.: Triangulation. CVIU **68** (1997) 146–157
8. Stewénius, H., Schaffalitzky, F., Nistér, D.: How hard is three-view triangulation really? In: Int. Conf. Computer Vision. (2005) 686–693
9. Kahl, F., Henrion, D.: Globally optimal estimates for geometric reconstruction problems. In: Int. Conf. Computer Vision, Beijing, China (2005) 978–985
10. Ke, Q., Kanade, T.: Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In: CVPR. (2005) 739–746
11. Kahl, F.: Multiple view geometry and the L_∞-norm. In: Int. Conf. Computer Vision, Beijing, China (2005) 1002–1009
12. Sturm, J.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software **11-12** (1999) 625–653
13. Schaible, S., Shi, J.: Fractional programming: the sum-of-ratios case. Opt. Meth. Soft. **18** (2003) 219–229
14. Kotz, S., Kozubowski, T.J., Podgorski, K.: The Laplace distribution and generalizations. Birkhäuser (2001)
15. Wolf, L., Shashua, A.: On projection matrices $P^k \mapsto P^2$, $k = 3, \ldots, 6$, and their applications in computer vision. Int. Journal Computer Vision **48** (2002) 53–67