Practical Global Optimization for Multiview Geometry

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

[1] Centre for Mathematical Sciences
Lund University, Sweden

[2] Dept of Computer Science and Engineering
University of California San Diego, USA

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. Even though the worst case complexity of our algorithm is exponential, the practical efficacy 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 on a digital camera user's home computer.

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, 5, 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.

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.

1.1 Related Work

Recently there has been some progress made towards finding the global solution to a *few* of the multiview 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, 12], a wider class of geometric reconstruction problems was solved globally, but with L_∞-norm.

A preliminary version of this work was presented in the conference paper [13]. Also, our framework of branch and bround has recently been extended to the problems of (i) optimal triangulation of line and conic features in [14] and (ii) autocalibration in [15].

1.2 Outline

We begin by formulating the problems we are interested in solving in the next section. Then, an exposition on fractional programming is given in Section 2. Contained therein is an introduction to branch and bound algorithms (Section 3.1) followed by details of the construction of lower bounds (Section 3.2) and our branching strategy (Section 3.3). We justify in Section 4 the

claim that a broad class of multiview geometry problems with different noise models can be cast in the unifying framework of fractional programming. Section 5 presents two innovations crucial to expeditious convergence that exploit the special properties of structure and motion problems - a novel *bounds propagation scheme* to restrict the branching process to a small, fixed number of dimensions independent of the problem size and an intuitive *initialization* strategy based on reprojection error. Finally, Section 6 presents the experimental results of the extensive evaluation of our algorithm on a variety of synthetic and real data sets over several different noise levels.

2 Problem Formulation

A perspective camera can be modelled as a linear mapping $\mathbb{P}^3 \mapsto \mathbb{P}^2$ from projective 3-space to a projective image plane. In matrix notation, a 3D scene point, represented by a homogeneous 4-vector X, and its projected image point, represented by a homogeneous 3-vector x, are related by

$$\lambda x = PX$$

where λ is a scalar accounting for depth and P is the 3×4 camera matrix encoding intrinsic and extrinsic parameters of the camera.

We consider the following two problems under three different noise models, namely the Gaussian and two variants of the bivariate Laplacian.

1. Structure Estimation: Given N images of a point and the corresponding camera matrices, estimation of the position of the point in $\mathbb{P}^3$. This is also known as *the triangulation problem*.

2. Transformation Estimation: Given the position of N points in the projective space $\mathbb{P}^n$ and their images in the space $\mathbb{P}^m$, estimation of the projective transformation P that maps these points from $\mathbb{P}^n$ to $\mathbb{P}^m$. When $n = 3$ and $m = 2$, that is, the transformation is a 3×4 camera matrix, the problem is also known as *camera resectioning*.

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 3D homogeneous coordinates, then the reprojection residual vector for one image is given by

$$r = \left(u - \frac{p_1^\top X}{p_3^\top X}, v - \frac{p_2^\top X}{p_3^\top X} \right)^\top . \tag{2.1}$$

Under a Gaussian noise model, the objective function to minimize is the sum-of-squared residuals which becomes

$$\sum_{i=1}^{N} ||r_i||_2^2, \tag{2.2}$$

where N is the number of residual terms in the problem. Other noise models will also be considered later on.

Minimizing the sum-of-squares objective function (2.2) is known to be a troublesome non-convex optimization problem for both structure and transformation estimation [3]. Already the seemingly simple two-view triangulation problem have several local minima [7]. This phenomena causes difficulties for local optimization techniques such as Newton-based methods since they may get stucked in local minima.

As an example, consider the following three-view triangulation problem (first published in [11]) in which there are three local L_2 minima, all lying in front of all three cameras. In this example, all points lie in the plane $z = 0$,

so we may simplify the problem to a 2-dimensional triangulation problem. Adding a third dimension makes no significant difference to the example.

Let P_0 be represented by the camera matrix $P_0 = \left[\begin{smallmatrix} -3 & 1 & -8 \\ -1 & -3 & -6 \end{smallmatrix}\right]$. The centre of this camera is at the point $(-3, -1, 1)^\top$. We obtain two other cameras P_1 and P_2 by rotating around the origin by $\pm 120^\circ$.

Now, for all $i = 0, \ldots, 2$, let $x_i = (3, 1)^\top$; this is simply the point with non-homogeneous coordinate 3 in the image. It is easily seen that all points of the form $(x, -1, 1)^\top$ map to the same point $(3, 1)^\top$ in the P_0 image. These points lie along the line $y = -1$, which is therefore the ray corresponding to the image point $x_0 = (3, 1)^\top$ for the P_0 camera. The rays corresponding to the points measured in the other images lie on lines rotated by $\pm 120^0$ around the origin. The three rays form a triangle. Since this configuration has three-fold symmetry, if there is to be a single minimum to the cost function, then it could only be the origin, which is the symmetry centre. It is easily seen that the origin is not the global optimum. One might suspect that the local optima are at the vertices of the triangle. However, the best L_2 solutions do not lie exactly at the vertices of the triangle. The contour plot (sublevel-set plot) of the L_2 error (of a slightly perturbed problem) is shown in Figure 1.

3 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 minimiza-

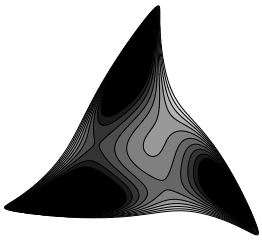


Figure 1: *A contour plot of the L_2 error for a three-view triangulation problem in which there are three local minima for the L_2 cost function.*

tion problem

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

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 through iterative algorithms that solve an appropriate convex optimization problem at each step.

For the purposes of the development of the Branch and Bound algorithm, 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:

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

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 3.1 *$(x^*, t^*, s^*) \in \mathbb{R}^{n+2p}$ 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).

3.1 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.

We will restrict our treatment to minimization problems as that is the case we will encounter in pertinent structure and motion problems.

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:

(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

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

An intuitive technique to determine the ϵ-suboptimal solution would be to divide the whole search region Q_0 into a grid with cells of sides δ and compute the minimum of a lower bounding function Φ_{lb} defined over each grid cell, with the presumption that each $\Phi_{\text{lb}}(Q)$ is easier to compute than the corresponding $\Phi_{\min}(Q)$. However, the number of such grid cells increases rapidly as $\delta \to 0$, so a clever procedure must be deployed to create as few cells as possible and "prune" away as many of these grid cells as possible (without having to compute the lower bounding function for these cells).

Branch and bound algorithms iteratively subdivide domain into rectangles and employ clever strategies to "prune" away as many rectangles as possible to restrict the search region.

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 2.

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

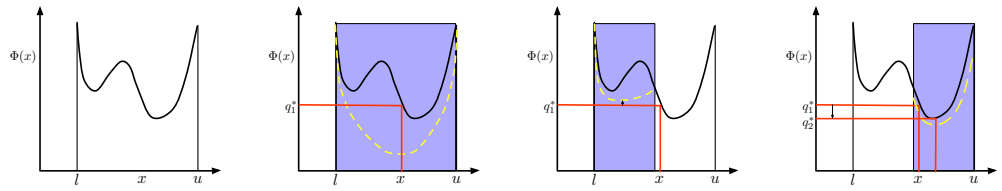


Figure 2: This figure illustrates the operation of a branch and bound algorithm on a one dimensional non-convex minimization problem. Figure (a) shows the the function $\Phi(x)$ and the interval $l \leq x \leq u$ in which it is to be minimized. Figure (b) shows the convex relaxation of $\Phi(x)$ (indicated 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 $\Phi(x)$.

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. Further detailed descriptions of the bounding and branching procedures are given in the next two subsections.

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.

3.2 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 3.1 (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

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 satisfied
  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

Second Order Cone Program (SOCP):

$$\mathbf{convenv}\begin{bmatrix}t\\ \hline s\end{bmatrix} = \min_{r,r',s'} \quad r$$

subject to $\qquad r, r', s' \in \mathbb{R}$

$$\left\| \begin{array}{c} 2\lambda\sqrt{l} \\ r' - s' \end{array} \right\| \leq r' + s' \qquad \left\| \begin{array}{c} 2(1-\lambda)\sqrt{u} \\ r - r' - s + s' \end{array} \right\| \leq r - r' + s - s'$$

$$\lambda L \leq s' \leq \lambda U \qquad (1-\lambda)L \leq s - s' \leq (1-\lambda)U$$

$$r' \geq 0 \qquad r - r' \geq 0$$

where we have substituted $\lambda = \dfrac{u-t}{u-l}$ for ease of notation, and r, r', 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 $\mathbf{convenv}\, f \geq \sum_i \mathbf{convenv}\, t_i/s_i$. 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 [16].

In summary, in order to compute a lower bound $\Phi_{\text{lb}}(Q)$ on the rectangle $Q = [\, l_1, u_1 \,] \times \cdots \times [\, l_p, u_p \,] \times [\, L_1, U_1 \,] \times \cdots \times [\, L_p, U_p \,]$, the following SOCP

is solved:

$$\min_{x,r,r',s,s',t} \quad \sum_{i=1}^{p} r_i$$

subject to $x \in \mathbb{R}^n,\ r, r', s, s', t \in \mathbb{R}^p$

$$\left\| \begin{array}{c} 2\lambda_i\sqrt{l_i} \\ r_i' - s_i' \end{array} \right\| \leq r_i' + s_i' \qquad \left\| \begin{array}{c} 2(1-\lambda_i)\sqrt{u_i} \\ r_i - r_i' - s_i + s_i' \end{array} \right\| \leq r_i - r_i' + s_i - s_i'$$

$$\lambda_i L_i \leq s_i' \leq \lambda_i U_i \qquad (1-\lambda_i)L_i \leq s_i - s_i' \leq (1-\lambda_i)U_i$$

$$r_i' \geq 0 \qquad r_i - r_i' \geq 0$$

$$l_i \leq t_i \leq u_i \qquad L_i \leq s_i \leq U_i$$

$$f_i(x) \leq t_i \qquad g_i(x) \geq s_i \qquad \text{for } i = 1, \ldots, p.$$

This construction of convex envelopes satisfies conditions (L1) and (L2), cf. [5], and therefore is well-suited for our branch and bound algorithm.

3.3 Branching

There are three issues that must be addressed within the branching phase - the rectangle to branch on, the dimension of the chosen rectangle to split along and the point at which to split the chosen dimension.

The choice of rectangle to be partitioned is 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 first.

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 [17]. However, we demonstrate in Section 5.1 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)

In [5], it is shown that the α-bisection leads to a branch-and-bound algorithm which is convergent.

4 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. The most common noise model is the Gaussian distribution which has a very thin tail, that is, the probability of large deviation decreases to zero very rapidly. In practice, however, large errors occur more often than predicted by the Gaussian distribution, for instance, due to erroneous localization of interest points or just bad correspondences. There are two ways of getting around this problem. The first is to robustify the cost function by reducing the penalty for large deviations and the second is to consider noise models with thicker tails [18]. The latter choice then translates into a modified likelihood function. We will consider the Gaussian and two variants of the Laplacian noise model.

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

$$p(x|x') = (2\pi\sigma^2)^{-1} \exp(-\|x - x'\|_2^2/(2\sigma^2)) \ . \tag{4.1}$$

Thus maximizing the likelihood, 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 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 [19]. 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:

$$p(x|x') = (2\pi\sigma)^{-1} \exp(-\|x - x'\|_2/\sigma) \ . \tag{4.2}$$

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

$$p(x|x') = \frac{1}{2\sigma} e^{-\frac{1}{\sigma}|u-u'|} \frac{1}{2\sigma} e^{-\frac{1}{\sigma}|v-v'|} \ = \ (4\sigma^2)^{-1} \exp(-\|x - x'\|_1/\sigma) \ . \tag{4.3}$$

Maximizing the likelihoods in (4.2) and (4.3) 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. In this notation the (L_2, L_∞)-case of the problems has recently been solved in polynomial time [11].

4.1 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

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. In the notation (L_p, L_q), the first norm L_p corresponds to the image norm used and the second one L_q to the norm of the residual vector.

reprojection residual vector for this image is given by

$$r = \left(u - \frac{p_1^\top X}{p_3^\top X}, v - \frac{p_2^\top X}{p_3^\top X} \right)^\top \tag{4.4}$$

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 (4.4) 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:

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

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 $\mathbf{epi}f = \left\{(X,t) \,|\, t \geq \|(a^\top X,\, b^\top X)\|\right\}$ 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.

4.2 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 (4.4) 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.

4.3 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 estimation 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 [20].

5 Multiview Fractional Programming

In this section, we present some important aspects of our implementation which extend the traditional methods to solve fractional programs by exploiting properties specific to the structure of multiview geometry problems. In fact, these developments form the basis for the excellent convergence rates our implementation achieves, as opposed to an exponential search in several dimensions that a naïve implementation of existing fractional programming techniques results in.

5.1 Bounds Propagation

Consider a fractional program with k fractions. Traditional approaches to fractional programming require branching in at least k dimensions corresponding to the denominators for the algorithm to converge correctly. For

a triangulation problem, k is the number of cameras and for a resectioning problem, it is the number of points. A branching dimension in a traditional branch and bound algorithm is the denominator of the reprojection error term corresponding to each point (for resectioning) or camera (for triangulation). It is evident that the search space of a branch and bound algorithm that branches in k dimensions can be untenably large even for medium-sized problems. Contemporary literature [17] documents reasonable results for practical problems with k at most 10 to 12.

However, we can do much better with the realization that for all problems presented in Section 4, the denominator is a linear function in the unknowns. To elucidate the concept, let us assume the problem is one of triangulating the location of a (homogeneous) point $X = (U, V, 1)^\top \in \mathbb{R}^3$ so that the branching entity (the denominator $g(X)$) is a linear function in two variables U and V. Please refer to Figure 3 for an illustration. Each bounding constraint restricts the denominator to lie in a particular half space in $\mathbb{R}^2$, thus, a pair of lower and upper bounds on two linearly independent denominators g_1 and g_2 restrict the feasible values to a convex quadrilateral on the $2D$ plane. Further, U and V are linear in g_1 and g_2 and so are the denominators of all the other fractions in the triangulation problem corresponding to views $3, \cdots, k$. So, the convex polygon that represents the bounds on the denominators g_1 and g_2 induces bounds on the denominators of all the fractions in the triangulation problem.

Extending the analogy to the case of triangulation in three dimensions, the unknown point coordinates $X = (U, V, W, 1)^\top$ are linear in $g_i(X) = p_{3i}^\top X$ for $i = 1, \ldots, k$. Suppose $k > 3$ and bounds are given on three denominators,

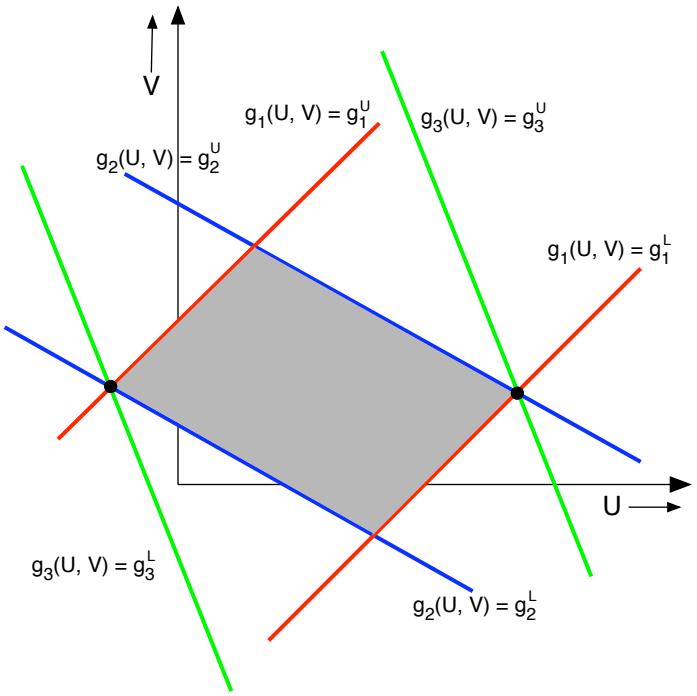


Figure 3: The red lines indicate the lower and upper bounds on the denominator g_1 while the blue lines indicate bounds on the denominator g_2. The shaded gray region represents the induced bounds on the variables U and V. Any linear function of U and V restricted to the domain represented by the gray polygon will attain its extremal values at two vertices of this simplex, as illustrated by the thick black points for some linear function $g_3(U, V)$ represented by the green lines.

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 $g_4, \ldots, g_k$. The bounds can be obtained by solving a set of linear equations each time branching is performed.

$$\begin{array}{lccc} \text{for } i = 1, \ldots, k, & \min g_i(x) & & \max g_i(x) \\ & L_j \leq g_j(x) \leq U_j & L_j \leq g_j(x) \leq U_j & j = 1, 2, 3. \end{array} \tag{R1}$$

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.

The choice of the n dimensions to be used for bounds propagation is, in our opinion, a matter of implementation preferences. A simple heuristic is to branch on the dimension along which the rectangle to be split is the widest and incorporate the same as one of the n dimensions used in the subsequent step of propagating bounds. This might, in principle, avoid issues with committing once and for all to some choice of n particular denominators as the ones branched upon, such as the case when two or more faces of the bounding constraints polytope are nearly parallel. However, in both our synthetic and real experiments, we have observed no such (numerical) instabilities.

As a practical note, we must point out that as the number of fractions increases, bounds propagation becomes the time critical step of the algorithm. However, the gains accrued in reduced dimensionality of the search

space more than outweighs any cost involved in solving the large LP which constitutes the bounds propagation step.

5.2 Initialization

Besides bounds propagation, another component of the algorithm crucial to a rapid convergence is the 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 $\mathbb{R}^{2k}$ is the starting point for the branch and bound algorithm.

It is clear that the size of this initial search region will affect the runtime of the search algorithm. However it is not clear how the user should specify the bounds that define the initial region, especially since they depend on the problem geometry and are not straightforward to guess intuitively.

What is intuitive, however, is the notion of reprojection error (in pixels) and it is easy for the user to specify a reasonable upper bound on the worst reprojection error. This upper bound can then be used to construct bounds on the numerator and denominator by solving a set of simple optimization problems.

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:

$$\text{for } i = 1, \ldots, k, \qquad \min g_i(x) \quad \frac{f_j(x)}{g_j(x)} \leq \gamma \qquad \max g_i(x) \quad \frac{f_j(x)}{g_j(x)} \leq \gamma \qquad j = 1, \ldots, k. \tag{5.1}$$

Depending on the choice of error norm, the above optimization problems will

be instances of linear programming (for $L_1 - L_1$) or quadratic programming (for $L_2 - L_1$ and $L_2 - L_2$). We will call this γ-initialization.

If the user-specified reprojection error is too small to lead to a feasible solution or so large that the SOCP solver is mired in numerical errors, the algorithm defaults to initial bounds which are wide enough for usual problem scales and known to be small enough to be numerically stable. This situation arises sometimes in our experiments, but we have found that the search space shrinks rapidly even with extremely liberal default values for the initial bounds.

As a further note on the implementation, while tight bounds on the denominators are crucial for the performance of the overall algorithm, the bounds on the numerators are not. Therefore, we set the numerator bounds to preset values.

5.3 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.

6 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 [16] and the code is publicly available[1]. 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 5.2) 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. In addition to the methods based on fractional programming, the results are also compared to that of bundle adjustment initialized with a linear method [3].

6.1 Synthetic Data

We demonstrate the various aspects of our algorithm such as scalability, runtime and termination using extensive simulations on 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 geometry for triangulation, whereby three cameras are placed along a line at distances 5, 6 and 7 units, respectively, from the origin. In Fig-

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

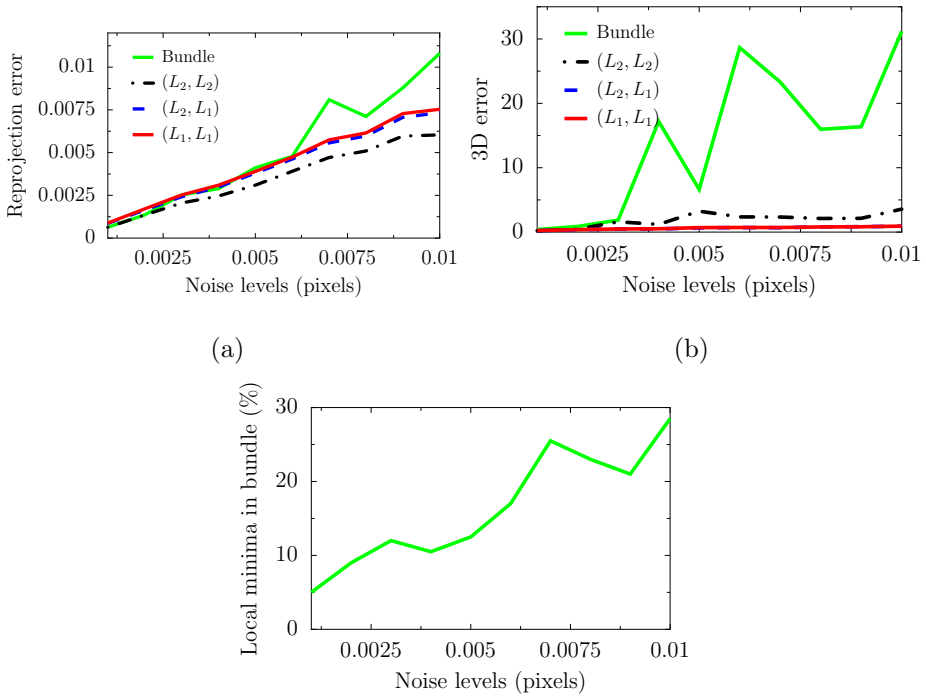
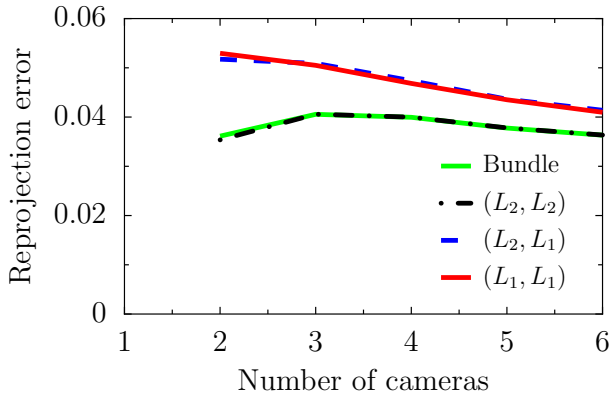


Figure 4: Triangulation with forward motion. Figure (a) compares the reprojection error of the three algorithms with bundle adjustment. Note the degradation in performance of bundle adjustment with increasing noise in the image, further demonstrated in Figure (b) which plots the mean 3D error for the four algorithms. Figure (c) shows percentage number of times the (L_2, L_2) algorithm found a better solution than bundle adjustment.

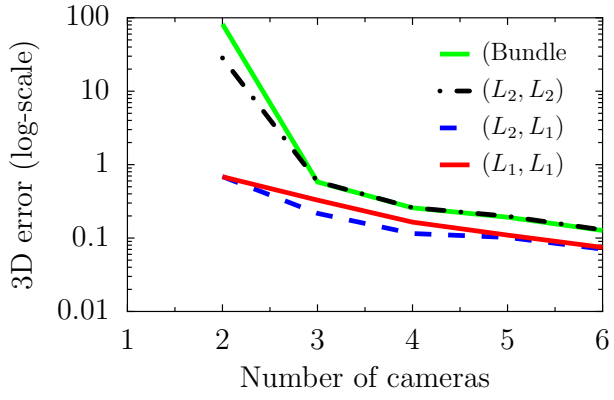
ures 4(a) and (b), the reprojection errors and the 3D errors are plotted, respectively. The (L_2, L_2) method (see Section 4 for notation), on the average, results in a much lower error than bundle adjustment, which can be attributed to bundle adjustment being enmeshed in local minima due to the non-convexity of the problem. The graph in Figure 4(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(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 6. Results for bundle adjustment and the (L_2, L_2) methods are identical and thus, likelihood of local minima is low. No errors on the estimated quantities are given since it is not meaningful to compare



(a)



(b)

Figure 5: (a) and (b) show reprojection and 3D erorrs, respectively, for triangulation with one outlier. Despite a higher reprojection error, the L_1-algorithms work better bundle adjustment in terms of 3D error.

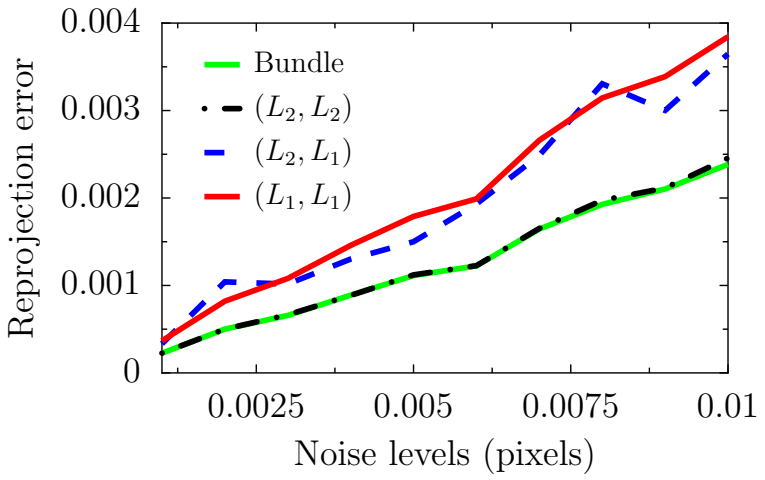
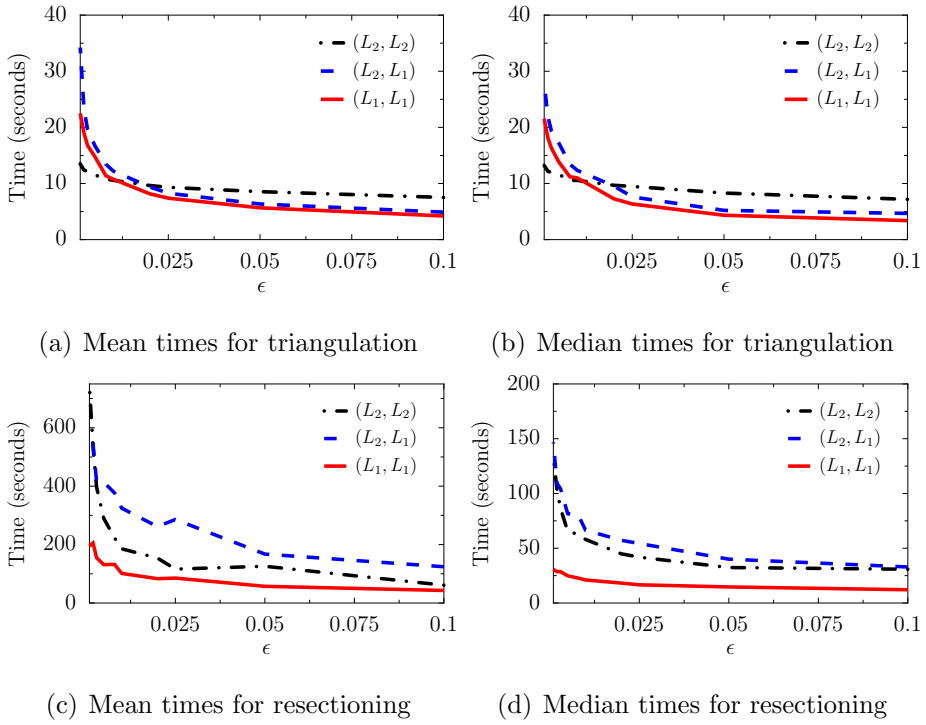


Figure 6: Reprojection errors for camera resectioning.

(homogeneous) camera matrices.

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. As can be seen, both median and mean runtime scale almost linearly with the size of the problem, making this an attractive algorithm for use in practice.

Finally, we demonstrate the effect of the optimality tolerance, ϵ, on the time it takes the branch and bound algorithm to converge. Five cameras were used for the triangulation experiment, placed in a circular arc of radius 1, looking towards the origin, with an angular separation of 10° between adjacent cameras. The points to be triangulated are generated in the cube



(a) Mean times for triangulation

(b) Median times for triangulation

(c) Mean times for resectioning

(d) Median times for resectioning

Figure 7: (a) and (b) show trends for the mean and median times, respectively, over 200 trials, for termination of the triangulation algorithm as the optimality tolerance, ϵ, is varied from 0.01 to 0.1. (c) and (d) show the same for the resectioning experiment.

$[-0.5, 0.5]^3$ and Gaussian noise of standard deviation 1% of image size is added to the image coordinates. Six points in general position are used for the resectioning experiments with similar additive noise.

The mean and median times over 200 trials for the triangulation and resectioning experiments are recorded in Figure 7 as ϵ is varied from 0.01 to 0.1.

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.

6.2 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 interestingly is perhaps the number of iterations and execution times on a standard PC (3 GHz), see Tables 4 and 5, respectively. We must point out that the implementations are (unoptimized) MATLAB functions. In the case of triangulation, a point is typically visible

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.

in a couple of frames. The differences in iterations and runtimes are most likely due to the setup: the dinosaur sequence has a circular camera motion and thereby a more well-posed camera geometry compared to the forward-moving camera in the corridor sequence.

In the camera resection problem, more parameters have to be estimated and therefore longer execution times compared to the triangulation problem. However, the main reason for the difference in performance is that the number of visible points are in the order of several hundreds for each camera.

7 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

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.

Experiment	Bundle		(L_2, L_2)		(L_2, L_1)		(L_1, L_1)	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
Dino (triangulation)	1.0	0.4	5.5	4.5	12.1	4.0	17.0	9.9
Corridor (triangulation)	1.0	0.6	18.7	17.7	51.0	47.3	46.4	51.6
Dino (resection)	4.0	3.0	273.1	192.3	640.0	554.1	312.8	304.9
Corridor (resection)	38.3	15.7	1433.5	348.0	1271.6	608.1	1122.7	565.0

Table 5: Triangulation and resectioning runtimes (in seconds) for real datasets.

algorithm is proposed that provably finds a solution arbitrarily close to the global optimum, with a fast convergence rate in practice. Note that the worst case complexity is exponential. 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. 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. Thus, from a practitioner's viewpoint, it is useful to set a lower criterion for global optimality and use gradient descent in the neighborhood of the resulting solution.

Needless to say, other segments of the computer vision community can also benefit from our approach as it is general enough to be applicable to any

problem formulated as a fractional program in a few independent dimensions. Another avenue for potential future work is the exploration of other algorithms for achieving global optimality in specialized fractional programs. As faster and more reliable algorithms are designed for achieving global optimality in fractional programs, we can anticipate corresponding improvements in our solution times.

8 Acknowledgements

S. Agarwal and S. 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. M. Chandraker and D. Kriegman are supported by NSF EIA 0303622 & NSF IIS-0308185. F. 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. Computer Vision and Image Understanding **68** (1997) 146–157

[8] Stewénius, H., Schaffalitzky, F., Nistér, D.: How hard is three-view triangulation really? In: Int. Conf. Computer Vision, Beijing, China (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: Conf. Computer Vision and Pattern Recognition, San Diego, USA (2005) 739–746

[11] Kahl, F.: Multiple view geometry and the L_∞-norm. In: Int. Conf. Computer Vision, Beijing, China (2005) 1002–1009

[12] Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. In: Int. Conf. Computer Vision, Beijing, China (2005) 986 – 993

[13] Agarwal, S., Chandraker, M., Kahl, F., Belongie, S., Kriegman, D.: Practical global optimization for multiview geometry. In: European Conf. Computer Vision, Graz, Austria (2006) 592–605

[14] Josephson, K., Kahl, F.: Triangulation of points, lines and conics. In: Scandinavian Conf. on Image Analysis, Aalborg, Denmark (2007)

[15] Chandraker, M.K., Agarwal, S., Kriegman, D.J., Belongie, S.: Globally convergent algorithms for affine and metric upgrades in stratified autocalibration. In: Int. Conf. Computer Vision, Rio de Janeiro, Brazil (2007)

[16] Sturm, J.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software **11-12** (1999) 625–653

[17] Schaible, S., Shi, J.: Fractional programming: the sum-of-ratios case. Opt. Meth. Soft. **18** (2003) 219–229

[18] Huber, P.: Robust statistics. Addison-Wesley, New York (1981)

[19] Kotz, S., Kozubowski, T.J., Podgorski, K.: The Laplace distribution and generalizations. Birkhäuser (2001)

[20] 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