Fast Algorithms for L_∞ Problems in Multiview Geometry

Sameer Agarwal
sagarwal@cs.washington.edu

Noah Snavely
snavely@cs.washington.edu

Steven M. Seitz
seitz@cs.washington.edu

University of Washington, Seattle

Abstract

Many problems in multi-view geometry, when posed as minimization of the maximum reprojection error across observations, can be solved optimally in polynomial time. We show that these problems are instances of a convex-concave generalized fractional program. We survey the major solution methods for solving problems of this form and present them in a unified framework centered around a single parametric optimization problem. We propose two new algorithms and show that the algorithm proposed by Olsson et al. [21] is a special case of a classical algorithm for generalized fractional programming. The performance of all the algorithms is compared on a variety of datasets, and the algorithm proposed by Gugat [12] stands out as a clear winner. An open source `MATLAB` *toolbox thats implements all the algorithms presented here is made available.*

1. Introduction

As the theory of multi-view geometry has matured, the focus of research has recently shifted from the study of the geometric and algebraic structure of the problem to the numerical solution of the resulting optimization problems.

A particularly fruitful line of work has been the development of methods that minimize the maximum reprojection error across observations (the L_∞ norm of the vector of reprojection errors) instead of the more commonly used sum of squared reprojection errors. The advantage of this approach is that in many cases the resulting optimization problem has new structure that is amenable to global optimization. In particular these optimization problems turn out to be quasi-convex [13, 15, 16] enabling efficient, globally optimal solutions using the methods of convex optimization [5]. A wide range of multi-view geometry problems have been solved in the L_∞ framework, including triangulation, camera resectioning, homography estimation, structure and translation with known rotations, reconstruction by using a reference plane, camera motion estimation and outlier removal [13, 15–17, 24, 25].

In all of these works, the method used for solving the L_∞ optimization problem is a bisection search for the minimax reprojection error. While this approach may be reasonable for small problems like triangulation and camera resectioning, the bisection algorithm is very slow for large scale problems like structure and translation estimation with known rotations, where the number of variables can be in the hundreds of thousands for large problems [18].

The objective of this paper is to present fast algorithms for the solution of large scale L_∞ problems. We first show that L_∞ problems in multi-view geometry are convex-concave generalized fractional programs (Section 2). Like the L_∞ problem, generalized fractional programs are also quasi-convex and can be solved using the bisection algorithm. However, unlike a generic quasi-convex program they have specific structure that can be exploited to build algorithms which are significantly faster than the bisection algorithm. We then introduce the parametric optimization problem that lies at the heart of a number of methods for solving generalized fractional programs (Section 3). We survey the major methods for solving generalized fractional programs and present them in a unified framework centered around this parametric optimization problem (Sections 4-7). Along the way, we propose two new algorithms for solving L_∞ problems (Section 4) and show that a recently proposed algorithm by Olsson et al. [21] for L_∞ optimization is a special case of a classical algorithm for generalized fractional programming (Section 5). We then compare the performance of the various algorithms on a variety of large scale data sets and show that an algorithm proposed by Gugat [12] stands out as a clear winner (Section 8). Last but not least, we make available an open source `MATLAB` toolbox for doing large scale L_∞ optimization. The toolbox includes all the code used to perform the experiments reported in this paper.

We now summarize the notational conventions used in the rest of the paper. Upper case letters, e.g., P_i, denote matrices, lower case Roman and Greek letters, e.g., a, γ, denote scalars, and bold-faced letters e.g., $\boldsymbol{x}, \boldsymbol{\lambda}$, denote column vectors. $\mathbf{0}$ and $\mathbf{1}$ denote vectors of all zeros and ones respectively. Superscripted symbols, e.g., $\boldsymbol{x}^k$ indicate iterates of an algorithm and the superscript $*$, e.g., γ^*, denotes an optimal solution. For two vectors $\boldsymbol{x} = [x_1, \ldots, x_n]$ and $\boldsymbol{y} = [y_1, \ldots, y_n]$, $\boldsymbol{x} \preceq \boldsymbol{y}$ is used to indicate $x_i \leq y_i$, $\forall i = 1, \ldots, n$. Finally, given scalar functions $f_i(\boldsymbol{x})\ i = 1, \ldots, m$, $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]$.

2. The L_∞ problem

We begin with a brief review of the L_∞ problem in multi-view geometry and its relation to generalized fractional programming. We use the triangulation problem as an example.

Given camera matrices $P_i = [R_i|\boldsymbol{t}_i], i = 1, \ldots, m$, where $R_i = [\boldsymbol{r}_{i1}, \boldsymbol{r}_{i2}, \boldsymbol{r}_{i3}]^\top$ and $\boldsymbol{t}_i = [t_{i1}, t_{i2}, t_{13}]$ and the corresponding images $[u_i, v_i]$ of a point $\boldsymbol{x} \in \mathbb{R}^3$, we wish to find that value of $\boldsymbol{x}$ which minimizes the maximum reprojection error across all images:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \max_{i=1}^{m} \left\| \left[u_i - \frac{\boldsymbol{r}_{i1}^\top \boldsymbol{x} + t_{i1}}{\boldsymbol{r}_{i3}^\top \boldsymbol{x} + t_{i3}}, v_i - \frac{\boldsymbol{r}_{i2}^\top \boldsymbol{x} + t_{i2}}{\boldsymbol{r}_{i3}^\top \boldsymbol{x} + t_{i3}} \right] \right\| \\ \text{subject to} \quad & \boldsymbol{r}_{i3}^\top \boldsymbol{x} + t_{i3} > 0, \quad \forall\, i = 1, \ldots, m. \end{aligned}$$

The constraint $\boldsymbol{r}_{i3}^\top \boldsymbol{x} + t_{i3} > 0$ ensures that the point x lies in front of each camera, and making use of it, the above problem can be re-written as a general problem of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \max_{i=1}^{m} \frac{\left\| \left[\boldsymbol{a}_{i1}^\top \boldsymbol{x} + b_{i1}, \boldsymbol{a}_{i2}^\top \boldsymbol{x} + b_{i2} \right] \right\|}{\boldsymbol{a}_{i3}^\top \boldsymbol{x} + b_{i3}} \\ \text{subject to} \quad & C\boldsymbol{x} \preceq \boldsymbol{d}, \end{aligned}$$

where the constants $\boldsymbol{a}_{ij}, \boldsymbol{b}_{ij}, C$ and $\boldsymbol{d}$ are appropriately defined.

There is flexibility in the choice of the norm $\|\cdot\|$. The L_2-norm leads to the formulation considered by Kahl [15] and Ke & Kanade [16], and the L_1-norm leads to the formulation considered by Seo & Hartley [23]. In both of these cases, each fraction in the objective function is of the form $f_i(\boldsymbol{x})/g_i(\boldsymbol{x})$, where $f_i(\boldsymbol{x}) = \|\boldsymbol{a}_{i1}^\top \boldsymbol{x} + b_{i1}, \boldsymbol{a}_{i2}^\top \boldsymbol{x} + b_{i2}\|$ is a convex function and $g_i(\boldsymbol{x}) = \boldsymbol{a}_{i3}^\top \boldsymbol{x} + b_{i3}$ is concave, in particular $g_i(\boldsymbol{x})$ is affine. For the remainder of this paper we will not differentiate between the two norms, and consider the generic optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad \max_{i=1}^{m} \frac{f_i(\boldsymbol{x})}{g_i(\boldsymbol{x})} \tag{P}$$

where $\mathcal{X} = \{\boldsymbol{x} | C\boldsymbol{x} \preceq \boldsymbol{d}\}$ is the convex polyhedral feasible set. Compactness of the feasible set is a common requirement for the convergence analysis of optimization algorithms. For L_∞ problems, the set $\mathcal{X}$ is usually not compact. This is however not a significant hurdle. A closed bounded set is a compact set. The feasible set $\mathcal{X}$ is closed by definition, and it is always possible to enforce compactness by adding a constraint $\|x\|_\infty \leq M$ to $\mathcal{X}$ for some large constant M without affecting the solution to the original problem. Therefore, without loss of generality, we assume that $\mathcal{X}$ is a compact convex polyhedral set with a non-empty interior.

2.1. Generalized Fractional Programming

A non-linear optimization problem is a *generalized fractional program* if it can be written as

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad \max_{i=1}^{m} \frac{f_i(\boldsymbol{x})}{g_i(\boldsymbol{x})} \tag{GFP}$$

where $\mathcal{X}$ is a nonempty subset of $\mathbb{R}^n$, $f_i(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are continuous on $\mathcal{X}$ and $g_i(\boldsymbol{x})$ are positive on $\mathcal{X}$ [10]. Further, if we assume

1. $\mathcal{X}$ is a convex set,
2. $\forall i$, $f_i(\boldsymbol{x})$ is convex and $g_i(\boldsymbol{x})$ is concave, and
3. $\forall i$, either $f_i(\boldsymbol{x})$ are non-negative or the functions $g_i(\boldsymbol{x})$ are affine,

then GFP is a convex-concave generalized fractional program. P is therefore a convex-concave generalized fractional program. From here on, the phrase generalized fractional program will always refer to convex-concave generalized fractional programs.

As was shown in [15] and [16], P is quasiconvex. In fact, all generalized fractional programs are quasiconvex. The proof is as follows:

By definition, a function $h(\boldsymbol{x})$ is quasi-convex if its domain is convex and for all γ, the sublevel sets $S_\gamma = \{\boldsymbol{x} | h(\boldsymbol{x}) \leq \gamma\}$ are convex [5]. The domain of the objective function $h(\boldsymbol{x}) = \max_i f_i(\boldsymbol{x})/g_i(\boldsymbol{x})$ is the convex set $\mathcal{X}$ and its γ-sublevel set is given by

$$\begin{aligned} S_\gamma &= \{\boldsymbol{x} \in \mathcal{X} | h(\boldsymbol{x}) \leq \gamma\} \\ &= \{\boldsymbol{x} \in \mathcal{X} | \max_i f_i(\boldsymbol{x})/g_i(\boldsymbol{x}) \leq \gamma\} \\ &= \{\boldsymbol{x} \in \mathcal{X} | f_i(\boldsymbol{x})/g_i(\boldsymbol{x}) \leq \gamma,\ \forall i = 1, \ldots, m\} \\ &= \{\boldsymbol{x} \in \mathcal{X} | \boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq \boldsymbol{0}\} \end{aligned}$$

If $\boldsymbol{f}(\boldsymbol{x})$ is non-negative, then S_γ is empty for $\gamma < 0$, otherwise $\boldsymbol{g}(\boldsymbol{x})$ is affine and $\boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x})$ is a convex function for all $\gamma \in \mathbb{R}$. Thus, depending on the value of γ, the set S_γ is either an intersection of a set of convex sets, or else S_γ is empty; in either case it is a convex set.

2.2. The Bisection Algorithm

Given initial bounds on the optimal value γ^*, we can perform a bisection search to find the minimum value of γ for which S_γ is non-empty, at each iteration solving an instance of the following feasibility problem(Algorithm 1).

$$\begin{aligned} \text{Find} \quad & \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathcal{X} \end{aligned} \tag{P_γ}$$

If S_γ is non-empty, the optimization algorithm will return some $x \in S_\gamma$ as output, otherwise it will report that the problem is infeasible. For the L_2 norm this reduces to a Second Order Cone Program (SOCP), and for the L_1 norm it reduces to a Linear Program (LP). In both cases, efficient polynomial time methods exist for solving the resulting optimization problem [5]. This is the standard method for solving generic quasi-convex optimization problems [5] and the algorithm suggested by Kahl [15] and Ke & Kanade [16].

Algorithm 1 Bisection Algorithm

Require: Initial interval $[l^1, u^1]$ s.t. $l^1 \leq \gamma^* \leq u^1$.

loop
  $\gamma^k = (l^k + u^k)/2$, Solve P_{γ^k} to get $\boldsymbol{x}^k$
  if feasible **then**
    $\boldsymbol{x}^* = \boldsymbol{x}^k, u^{k+1} = \max_i f_i(\boldsymbol{x}^k)/g_i(\boldsymbol{x}^k), l^{k+1} = l^k$
  else
    $l^{k+1} = \gamma^k, u^{k+1} = u^k$
  end if
  if $u^{k+1} - l^{k+1} \leq \epsilon_1$ **then**
    return $(\boldsymbol{x}^*, u^{k+1})$
  end if
end loop

Consider the sequence γ_k, which converges to γ^* in the limit. If, for some α and $c < 1$, the following limit exists

$$\lim_{n\to\infty} \frac{|\gamma^{k+1} - \gamma^*|}{|\gamma^k - \gamma^*|^\alpha} = c$$

then the sequence γ^k is said to have an order of convergence α. Sequences for which $\alpha = 1$ are said to converge linearly. In general, sequences with higher orders of convergence converge faster than sequences with lower orders. In many cases of interest, the exact order of convergence is hard to prove and we have to satisfy ourselves with lower bounds on the order of convergence. For example, if

$$\lim_{n\to\infty} \frac{|\gamma^{k+1} - \gamma^*|}{|\gamma^k - \gamma^*|} = 0,$$

it implies that $\alpha > 1$ and the sequence is said to be *super linearly* convergent.

While simple to implement and analyze, the bisection algorithm suffers from two major shortcomings. First, in each iteration, the bisection algorithm reduces the search space by half, hence $\alpha = 1$ and $c = 1/2$. Thus the bisection algorithm converges linearly. Second, effort spent on searching for a feasible point when the set S_γ is empty is wasted, i.e., it tells us nothing about the solution beyond the fact that the optimal mini-max reprojection error is greater than γ.

2.3. Related Work in Computer Vision

The complexity of the L_∞ optimization problem is a function of the number of observations. One interesting feature of the L_∞ problem is that only a small subset of the observations actually constrain the solution, i.e., if we remove all observations not in the support set of the optimum, this reduced problem would have the same solution [25]. Using this observation, Seo & Hartley propose an iterative algorithm that solves a series of L_∞ problems to construct a subset of the observations which is guaranteed to contain the support set of the optimal solution [23]. The hope is that each intermediate L_∞ problem is small enough to be solved quickly and that the total effort is less than what is needed to solve the full problem. In our experience, the performance of this algorithm depends crucially on the distribution of reprojection errors and for distributions with thick tails the performance can be quite poor.

Olsson et al. exploited the pseudo-convexity of the reprojection error to construct an interior point method that numerically solves the Karush-Kuhn-Tucker equations [21]. This method was found to be numerically unstable with slow or pre-mature convergence. They also proposed a second method based on solving a series of SOCPs, which had good empirical performance. However, no convergence theory for this method was given. In this paper we show that this second method is in fact a classical method for solving generalized fractional programs and has super-linear convergence.

We note that in this paper we do not cover the work on interior point methods developed specially for fractional programs [11, 20]. These methods require the development of specialized codes. Our interest is in methods which can exploit the development of advanced solvers for linear and conic programming to build scalable algorithms.

3. A Parametric View of L_∞ Optimization

Let us now consider the following parametric optimization problem:

$$w(\gamma) = \left\{ \begin{array}{ll} \min_{w,\boldsymbol{x}} & w \\ \text{subject to} & \boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq w\mathbf{1} \\ & \boldsymbol{x} \in \mathcal{X} \end{array} \right. \qquad (Q_\gamma)$$

Parametric here implies that we will be considering the solution of this optimization problem for various values of the parameter γ. We denote by $w(\gamma)$ the optimal value function; for each γ, $w(\gamma)$ is equal to the minimum value attained by the variable w. Earlier, we saw that for fixed values of γ, $\boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x})$ is convex, thus Q_γ is also a convex program. Note that if we fix $w = 0$ then Q_γ reduces to P_γ. Further, if the set $\mathcal{X}$ is non-empty, then Q_γ is feasible for all values of γ, and $w(\gamma) > 0$ if and only if P_γ is infeasible.

The optimal value function $w(\gamma)$ has a number of interesting properties:

Theorem 1 ([7]). *For all γ, $w(\gamma)$ is finite, decreasing and continuous. P and Q_γ always have optimal solutions. The optimal value of P, γ^*, is finite and $w(\gamma^*) = 0$. $w(\gamma) = 0$ implies $\gamma = \gamma^*$.*

Theorem 1 establishes a link between the solutions of Q_γ and P. The problem of solving P can now be rephrased the problem of finding the zero of the function $w(\gamma)$. In the next four sections we describe four different approaches to this problem. All of the approaches are based on solving a series of problems Q_{γ^k}; what differentiates them from each other

is how they exploit the structure of P and Q_γ to determine the sequence γ^k, and how quickly this sequence converges to the optimal γ^*.

4. Bisection and related methods

The problem of finding the roots of an function of one variable is one of the oldest problems in mathematics, and there are a wide variety of solution methods available.

The simplest algorithm is the bisection algorithm, which starts with an interval known to contain a root and performs a binary search to find it. The bisection method for solving quasi-convex problems, described earlier, does exactly this; since this method only considers the sign of $w(\gamma)$, an indication of the feasibility or infeasibility of P_γ is enough.

We now consider two new methods for finding the root of $w(\gamma)$.

4.1. A New Bisection Algorithm

Because the feasible set S_γ gets smaller as γ gets closer to γ^*, the feasibility problem P_γ gets harder as we get closer to the solution. Since Q_γ is always feasible, our first proposal is to use to use Q_γ in place of P_γ in the bisection problem, replacing the feasibility test with a test for the sign of $w(\gamma_k)$.

4.2. Brent's Method

The bisection algorithm is an extremely robust algorithm, but this robustness comes at the price of linear convergence. Interpolation-based algorithms, such as the secant method and the method of false position, use a model (linear or quadratic) to predict the position of the root based on the current knowledge of the function. A number of interpolation-based methods have superlinear convergence.

A modern method which combines the speed of interpolation-based methods with the robustness of the bisection algorithm is the Brent method [6]. This method uses a inverse quadratic interpolation scheme with safeguards that include bracketing and switching to bisection when the interpolation update moves too slowly. Our second proposal is to use Brent's method to find the roots of $w(\gamma)$.

5. Dinkelbach's Algorithm

In the last section we considered root finding methods that solve the equation $w(\gamma) = 0$ by querying the value of $w(\gamma)$ at various values of γ. These methods do not make use of the structure of P or Q_γ, and treat the function $w(\gamma)$ as a black box. There is hope that methods which consider the form of P and Q_γ can achieve better performance than such black box methods. Starting in this section we consider methods that take the specific form of the objective function of P into account.

The first class of methods we consider are based on estimating and using the gradient of $w(\gamma)$. We begin by considering the special case of P when $m = 1$, i.e., the single ratio problem:

$$\min_{x \in \mathcal{X}} \quad \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \tag{P1}$$

and its associated parametric problem

$$w(\gamma) = \min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) - \gamma g(\boldsymbol{x}). \tag{$Q1_\gamma$}$$

Since $w(\gamma)$ is the minimum of an affine function over a convex region, it is concave in γ. Further, let $\gamma^k \in \mathbb{R}$ and $\boldsymbol{x}^k$ be the solution to $Q1_\gamma$. Then for all γ we have

$$\begin{aligned} w(\gamma) &\leq f(\boldsymbol{x}^k) - \gamma g(\boldsymbol{x}^k) \\ &\leq (f(\boldsymbol{x}^k) - \gamma^k g(\boldsymbol{x}^k)) - g(\boldsymbol{x}^k)(\gamma - \gamma^k) \\ &\leq w(\gamma^k) - g(\boldsymbol{x}^k)(\gamma - \gamma^k) \end{aligned}$$

Thus, $-g(\boldsymbol{x}^k)$ is a *supergradient* of $w(\gamma)$ at γ^k [5]. Supergradients generalize the notion of derivatives for concave functions. When a function is differentiable, it has a unique supergradient equal to its derivative, but a general concave function can have more than one supergradient at a point.

Newton's method for finding the roots of the equation $w(\gamma) = 0$ can be stated as the following update rule:

$$\gamma^{k+1} = \gamma^k - \frac{w(\gamma^k)}{\partial_\gamma w(\gamma^k)}.$$

The utility of supergradients is that they can be used to construct a Newton method for concave functions. Replacing the gradient with the supergradient gives us

$$\gamma^{k+1} = \gamma^k + \frac{w(\gamma^k)}{g(\boldsymbol{x}^k)} = \frac{f(\boldsymbol{x}^k)}{g(\boldsymbol{x}^k)} \tag{1}$$

Depending upon the smoothness properties of $w(\gamma)$, Newton methods can have convergence rates quadratic or better. Unfortunately, since $w(\gamma)$ is not differentiable in general, and we only have access to its supergradients, the convergence rate of Eq. 1 is only superlinear [14]. This method is known in the literature as Dinkelbach's Procedure [9]. Motivated by Eq. 1, Crouzeix et al. suggested [8] using

$$\gamma^{k+1} = \max_i \frac{f(\boldsymbol{x}^k)}{g(\boldsymbol{x}^k)}$$

to solve the case when $m > 1$. Algorithm 2 describes the resulting algorithm. One would hope that an analog of the supergradient inequality will hold true for this algorithm too. Unfortunately that is not true and only a weaker inequality holds: [8]

$$w(\gamma) \leq \begin{cases} w(\gamma^k) - \min_i\{g_i(x^k)\}(\gamma - \gamma^k) & \gamma > \gamma_k \\ w(\gamma^k) - \max_i\{g_i(x^k)\}(\gamma - \gamma^k) & \gamma < \gamma_k \end{cases} \tag{2}$$

Consequently, the resulting algorithm converges only linearly.

Algorithm 2 Dinkelbach's Algorithm

Require: Initial $\gamma^1 \geq \gamma^*$
loop
  Solve Q_{γ^k} to get $(\boldsymbol{x}^k, w^k)$
  $\gamma^{k+1} = \max_i f_i(\boldsymbol{x}^k)/g_i(\boldsymbol{x}^k)$
  if $|w^k| \leq \epsilon_2$ **then**
    return $(\boldsymbol{x}^k, \gamma^{k+1})$
  end if
end loop

5.1. Scaled Dinkelbach's Algorithm

But all hope is not lost. Observe that

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad \max_{i=1}^{m} \frac{f_i(\boldsymbol{x})/v_i}{g_i(\boldsymbol{x})/v_i}$$

for any $v_i > 0$ has exactly the same solution as P. In particular, this is true for $v_i = g_i(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is an optimal solution to P. Let us consider the corresponding parametric problem

$$w'(\gamma) = \left\{ \begin{array}{rl} \min & w \\ \text{subject to} & \dfrac{f_i(\boldsymbol{x}) - \gamma g_i(\boldsymbol{x})}{g_i(\boldsymbol{x}^*)} \leq w,\ i = 1, \ldots, m \\ & \boldsymbol{x} \in \mathcal{X} \end{array} \right.$$

At γ^*, the above problem has the solution $\boldsymbol{x}^*$. Now let us see what happens to Eq. 2 at $(\gamma^*, \boldsymbol{x}^*)$. Since $\max_i\{g_i(\boldsymbol{x}^*)/g_i(\boldsymbol{x}^*)\} = 1 = \min_i\{g_i(\boldsymbol{x}^*)/g_i(\boldsymbol{x}^*)\}$, the two cases of the inequality 2 collapse into one. Thus, in the neighborhood of $\boldsymbol{x}^*$, $-\max_i\{g_i(\boldsymbol{x})/g_i(\boldsymbol{x}^*)\}$ is approximately the supergradient and we can recover superlinear convergence. Of course we do not know $\boldsymbol{x}^*$ *a priori*. But it suggests a modification to Algorithm 2, where Q_γ is replaced by the scaled problem

$$\begin{array}{rl} \min & w \\ \text{subject to} & \boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq w \boldsymbol{g}(\boldsymbol{x}^{k-1}) \\ & \boldsymbol{x} \in \mathcal{X} \end{array} \qquad (Q'_\gamma)$$

The resulting algorithm is known as Dinkelbach's Procedure of Type II, or the differential correction algorithm [2].

5.1.1 Equivalence to Olsson et al.

There is another way in which we can arrive at this algorithm. Let us re-write P as

$$\begin{array}{rl} \min & \gamma \\ \text{subject to} & \boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathcal{X} \end{array} \qquad (P)$$

Now, consider the first-order Taylor expansion of the term $\gamma g_i(\boldsymbol{x})$ around $(\gamma^k, g_i(\boldsymbol{x}^k))$

$$\begin{aligned} \gamma g_i(\boldsymbol{x}) \approx & \gamma^{k-1} g_i(\boldsymbol{x}^k) + (\gamma - \gamma^{k-1}) g_i(\boldsymbol{x}^{k-1}) \\ & + \gamma^{k-1}(g_i(\boldsymbol{x}) - g_i(\boldsymbol{x}^{k-1})) \\ = & \gamma^{k-1} g_i(\boldsymbol{x}) + (\gamma - \gamma^{k-1}) g_i(\boldsymbol{x}^{k-1}) \end{aligned} \qquad (3)$$

Let $w = \gamma - \gamma^{k-1}$. Then P can be approximated as

$$\begin{array}{rl} \min & w + \gamma^{k-1} \\ \text{subject to} & \boldsymbol{f}(\boldsymbol{x}) - \gamma^{k-1}\boldsymbol{g}(\boldsymbol{x}) \preceq w \boldsymbol{g}(\boldsymbol{x}^{k-1}) \\ & \boldsymbol{x} \in \mathcal{X} \end{array}$$

which is exactly the optimization problem suggested by [21] for the case of the L_2 norm. [21] reported good empirical performance of the resulting algorithm, but did not provide any convergence analysis. Since γ^{k-1} is a constant, this optimization problem is equivalent to Q'_{γ^k}. Thus we have shown that the algorithm suggested in [21] is the classical Dinkelbach Procedure of Type II, and therefore has superlinear convergence. Further, the algorithm is applicable to both the L_1 and L_2 norm cases.

6. Dual Dinkelbach's Algorithm

The parametric problem Q_γ is a convex program. We now consider an algorithm that uses the Lagrangian dual of Q_γ to construct a superlinearly convergent algorithm [3]. Let Σ denote the set of vectors $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that $\boldsymbol{\lambda} \succeq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{\lambda} = 1$, and let

$$\gamma(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}\in\mathcal{X}} \frac{\boldsymbol{\lambda}^\top \boldsymbol{f}(\boldsymbol{x})}{\boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x})}. \qquad (4)$$

Then the following theorem characterizes $\gamma(\boldsymbol{\lambda})$.

Theorem 2 ([3]). *If $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}\in\Sigma} \gamma(\lambda)$, then $\gamma(\boldsymbol{\lambda}^*) = \gamma^*$ and if $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{\lambda}^{*\top} \boldsymbol{f}(\boldsymbol{x})/\boldsymbol{\lambda}^{*\top} \boldsymbol{g}(\boldsymbol{x})$ then $\boldsymbol{x}^*$ is also an optimal solution of P.*

Theorem 2, motivates solving P by solving $\max_{\boldsymbol{\lambda}\in\Sigma} \gamma(\lambda)$. Notice that like $w(\gamma)$, $\gamma(\boldsymbol{\lambda})$ is the optimal value function of an optimization problem, in this case the minimum value of a generalized fractional program with a single fraction. Making an analogy with the duality theory of constrained optimization [5], it is possible to consider Eq. 4 to be a dual of P with zero duality gap.

So how does one go about maximizing $\gamma(\boldsymbol{\lambda})$? Since Eq. 4 is a fractional program, consider the parametric problem associated with it:

$$w(\gamma, \boldsymbol{\lambda}) = \min_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{\lambda}^\top (\boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}))$$

From Theorem 1 we know that $w(\gamma, \boldsymbol{\lambda}) > 0$ implies that $\gamma(\boldsymbol{\lambda}) > \gamma$. This suggests the following iterative update

$$\boldsymbol{\lambda}^{k+1} = \arg\max_{\boldsymbol{\lambda}\in\Sigma} w(\boldsymbol{\lambda}, \gamma(\boldsymbol{\lambda}^k)) \qquad (5)$$

and the following result holds true.

Algorithm 3 Dual Dinkelbach's Algorithm

Choose $\boldsymbol{\lambda}^0 \in \Sigma$
loop
  $\gamma^k = \min_{x \in \mathcal{X}} \boldsymbol{\lambda}^{k-1\top} \boldsymbol{f}(\boldsymbol{x}) / \boldsymbol{\lambda}^{k-1^\top} \boldsymbol{g}(\boldsymbol{x})$
  Solve Q_{γ^k} to get $(\boldsymbol{x}^k, w^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$
  if $|w^k| \leq \epsilon_1$ **then**
    return
  end if
end loop

Theorem 3 ([3]). *If $\boldsymbol{f}(\boldsymbol{x})$ is positive, strictly convex and $\boldsymbol{g}(\boldsymbol{x})$ is positive concave, then Eq. 5 converges superlinearly.*

Consider now the Lagrangian of Q_γ:

$$L(\boldsymbol{x}, w, \boldsymbol{\lambda}, \boldsymbol{\mu}; \gamma) = w + \boldsymbol{\lambda}^\top (\boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) - w\mathbf{1}) + \boldsymbol{\mu}^\top (C\boldsymbol{x} - \boldsymbol{d}) \quad (6)$$

Our use of the symbol $\boldsymbol{\lambda}$ as the dual variable associated with the constraint $\boldsymbol{f}(\boldsymbol{x}) - \gamma \boldsymbol{g}(\boldsymbol{x}) \preceq w\mathbf{1}$ is deliberate. Indeed the following holds true

Theorem 4 ([3]). *If $(\boldsymbol{x}^*, w^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a primal-dual solution to Q_γ, then $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda} \in \Sigma} w(\boldsymbol{\lambda}, \gamma)$*

Algorithm 3 describes the resulting algorithm. The algorithm successively approximates γ^* from below and can be considered a dual to the Dinkelbach algorithms which approximate γ^* from above.

7. Gugat's Algorithm

Recall that the reason why Dinkelbach's Procedure of Type I has linear convergence in the case of multiple ratios is because the supergradient inequality does not hold true anymore. The Scaled Dinkelbach's Algorithm works well in the neighborhood of the optimal solution, but its linearization breaks down away from the solution.

The classical presentation of Newton's method is based on assuming that the function being considered is differentiable. If, however, we are satisfied with superlinear convergence, Newton's method can be constructed using the notion of the one-sided derivative:

$$\partial_\gamma^+ w(\gamma) = \lim_{\delta \to 0+} \frac{w(\gamma + \delta) - w(\gamma)}{\delta}$$

The general problem of estimating the derivatives of the optimal value function with respect to the parameters of the optimization problem is addressed in the perturbation theory of optimization problems [4]. Under suitable regularity conditions, a classical result relates the one-sided derivative of the optimal value function with the derivatives of the Lagrangian as follows:

$$\partial_\gamma^+ w(\gamma) = \inf_{\boldsymbol{x}, w \in X(\gamma)} \sup_{\boldsymbol{\lambda}, \boldsymbol{\mu} \in \Lambda(\gamma)} \partial_\gamma L(\boldsymbol{x}, w, \boldsymbol{\lambda}, \boldsymbol{\mu}; \gamma)$$

Algorithm 4 Gugat's Algorithm

Require: $l^1 \leq \gamma^1 \leq u^1$, such that $l^1 \leq \gamma^* \leq u^1$.
loop
  Solve Q_{γ^k} to get $(\boldsymbol{x}^k, w^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$
  $z^k = \max_i f_i(x^k)/g_i(x^k)$
  if $z^k < \gamma^*$ **then**
    $\boldsymbol{x}^* = \boldsymbol{x}^k, \gamma^* = z^k$
  end if
  $u^{k+1} = \min(u^k, z^k)$
  if $w^k \geq 0$ **then**
    $l^{k+1} = \max(l^k, \gamma^k + w^k/\sigma)$
  else
    $l^{k+1} = l^k$
  end if
  if $|w^k| \leq \epsilon_1$ or $(u^{k+1} - l^{k+1}) \leq \epsilon_2$ **then**
    return $(\boldsymbol{x}^*, \gamma^*)$
  end if
  $\gamma^{k+1} = \max\left(l^{k+1}, \min\left(\gamma^k + w^k / \boldsymbol{\lambda}^{k\top} \boldsymbol{g}(\boldsymbol{x}^k), u^{k+1}\right)\right)$
end loop

Here, $L(\boldsymbol{x}, w, \boldsymbol{\lambda}, \boldsymbol{\mu}; \gamma)$ is the Lagrangian given by Eq. 6, and, $X(\gamma)$ and $\Lambda(\gamma)$ are the sets of the primal and dual solutions of Q_γ. Results of this type, are however, not particularly useful from a computational point of view, since they involve finding a saddle point over the Cartesian product of all primal and dual solutions.

Gugat showed that, given a particular primal-dual solution pair $(x^*, w^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, the derivative of the Lagrangian at that point approximates the one-sided derivative well enough that the Newton update converges superlinearly [12]. i.e.,

$$\partial_\gamma^+ L(\boldsymbol{x}^*, w^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*; \gamma) \approx -\boldsymbol{\lambda}^{*\top} \boldsymbol{g}(\boldsymbol{x}^*)$$

Thus, the update rule for γ can now be stated as

$$\gamma^{k+1} = \gamma^k + \frac{w^k}{\boldsymbol{\lambda}^{k\top} \boldsymbol{g}(\boldsymbol{x}^k)} \quad (7)$$

It is interesting to note here that, for the case when $m = 1$, $\boldsymbol{\lambda}^* = 1$ and $w^k = f_1(\boldsymbol{x}^k) - \gamma^k g_1(\boldsymbol{x})$, and the update rule reduces to the familiar Dinkelbach's update for single fractions: $\gamma^{k+1} = f_1(\boldsymbol{x}^k)/g_1(\boldsymbol{x}^k)$

Gugat's algorithm combines update rule 7 with bracketing which ensures that the iterates are always bounded and the algorithm does not diverge. Finally, we need a number σ that determines how much we can increase the lower bound l^k, if $w(\gamma^k)$ is non-negative, without missing the root. This modification ensures that the algorithm does not oscillate. σ should obey, $\sigma \geq \max_i \max_{\boldsymbol{x} \in \mathcal{X}} g_i(x)$. For our purposes, a large upper bound (1e6) is sufficient.

8. Experiments

In this section we compare the performance of the various algorithms we have described. Since our primary interest

is in large scale L_∞ optimization, we restrict our attention to the problem of estimating structure and translation with known rotations. As demonstrated by Martinec & Pajdla, solving this problem offers an alternative approach to the problem of reconstruction from multiple views [18].

Six algorithms were compared. Bisect I is from [15]. Bisect II and Brent were proposed in Section 4. Dinkel I and Dinkel II refer to Dinkelbach procedures of type I and II respectively. Gugat refers to Gugat's algorithm. The Dual Dinkelbach algorithm is omitted because it displayed extreme numerical instability when solving for γ^k using the single ratio problem. All algorithms were implemented in MATLAB. The MATLAB function `fzero` implements Brent's method and and we use this implementation in our experiments. For the L_2 norm, Q_γ is a SOCP and we use SeDuMi [27] as our solver.[1] For the L_1 norm, Q_γ is an LP, and we use MOSEK as the solver, as it had better runtime performance than SeDuMi and was able to handle problems that required memory greater than 2GB.

The algorithms were compared on 8 datasets. Tables 1 and 2 list the details of each dataset along with the performance of each algorithm on it. For each algorithm we list the runtime in seconds. Only the time used by the solver is noted here. The number in parentheses is the number of times the subproblem Q_γ was solved (P for Bisect I).

The Dino and the Oxford datasets are available from the Oxford Visual Geometry Group. The four Temple datasets are from [22]. The Pisa data set is a proprietary dataset, and the Trevi dataset is based on images of the Trevi fountain found on Flickr [1]. Except for the first two datasets, which come with camera information, the camera rotations and focal lengths were obtained from an independent bundle adjustment process [26]. Since outliers are a big issue in L_∞ problems, we used two kinds of datasets. The Dino, Oxford, and Temple 1-4 datasets are *clean* datasets with no significant outliers. Trevi and Pisa datasets contain a significant number of outliers. No results for the L_2 norm are reported for the Temple 3 & 4 and the Pisa datasets. For Temple 3, SeDuMi returned with a numerical failure. Temple 4 and Pisa were too large for the 32-bit version of SeDuMi to fit in memory. All experiments were run with an initial guess of $\gamma = 50$ and a lower bound of 0 and an upper bound of 100 pixels error. The termination parameters were $\epsilon_1 = 0.01$, $\epsilon_2 = 0.001$.

There are number of interesting features in both tables. Bisect I, Bisect II and Dinkel I are linearly convergent algorithms and it shows in their poor runtime performance as compared to the other three superlinear methods. There is no clear winner between Bisect I and Bisect II, and while Dinkel I is usually better on datasets with low noise, on datasets with a lot of outliers it consistently performs the worst.

Of the three blackbox methods, Bisect I, Bisect II and Brent, for L_1 problems Brent's method usually performs the best, but for L_2 problems Bisect I beat both Bisect II and Brent's method. Even for L_1 problems, Bisect I becomes more competitive as the problem size increases. This difference in performance can be explained by taking a closer look at the problems P_γ and Q_γ. Bisect I is based on solving P_γ which is a feasibility problem where as Bisect II and Brent's method use Q_γ, which is an optimization problem. For similar values of γ, P_γ is easier to solve since the optimizer terminates as soon as it finds a point inside the feasible set, whereas it has to find the analytic center of the constraints in case of Q_γ. Unfortunately Bisect II and Brent's method are unable to exploit the value of $w(\gamma)$ effectively enough to offset the cost of solving more expensive sub-problems. This becomes obvious if we look at the number of iterations for these methods. Bisect I consistently takes more iterations and still performs better on runtime as compared to Bisect II and Brent's method.

The clear winner out of the six algorithms is Gugat's algorithm, which had the best performance on every dataset. It particularly shines for large-scale sets, where it is between 1.5 to 4 times better than the Bisection algorithm. Its clever construction that exploits the dual solution to estimate the gradient of $w(\gamma)$ makes this algorithm both numerically robust and computationally efficient. Based on our experiences, we recommend that Gugat's algorithm be used as a standard algorithm for L_∞ optimization.

9. Discussion

In summary we have shown that L_∞ problems are a particular case of generalized fractional programming, and methods for solving them can be used with great success in multi-view geometry. While our experimental results have only considered the structure and translation estimation problem, the method presented in this paper are general and applicable to all the different L_∞ problems. We hope that Gugat's algorithm will become a standard tool for solving L_∞ problems in multi-view geometry.

It is also our observation that the L_2 problems are poorly conditioned as compared to the corresponding L_1 problems. Further, since LP solvers are much more mature than SOCP solvers, the $L1$ norm formulation is a better one to solve in our opinion. The exact cause of the conditioning problems of L_2 problems is a problem that deserves more attention.

In future work we hope to use the dual structure of Q_γ to analyze the problem of outlier removal.

[1] We also experimented with MOSEK [19], which is a leading commercial LP and SOCP solver, and found that Q_γ for moderate to large sized problems triggered a bug in the solver leading to poor numerical performance.

Dataset	Images	Points	Observations	Bisect-I	Bisect-II	Brent	Dinkel-I	Dinkel-II	Gugat
Dino	36	328	2663	12(13)	12(9)	7(5)	7(5)	6(4)	4(3)
Oxford	11	737	4035	19(13)	25(12)	12(6)	41(21)	f(f)	10(5)
Temple 1	43	4233	29163	226(13)	196(9)	109(5)	132(6)	104(5)	81(4)
Temple 2	103	8063	63373	676(13)	576(10)	275(5)	339(6)	277(5)	220(4)
Temple 3	203	15898	128530	985(13)	1646(10)	778(5)	1079(7)	794(5)	472(3)
Temple 4	312	22033	178897	1353(13)	1875(9)	1042(5)	1426(7)	1237(6)	619(3)
Trevi	58	4054	15085	191(14)	101(10)	70(7)	247(24)	59(6)	50(5)
Pisa	100	64023	436060	14435(14)	17311(13)	13665(10)	28396(28)	11352(7)	4617(4)

Table 1. Runtimes for L_1 norm reprojection error. All times are in seconds. The number in the parentheses indictes the number of times Q_γ or P_γ was solved. f denotes numerical failure. Parameter settings, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.001$, $\sigma = 1e6$.

Dataset	Images	Points	Observations	Bisect-I	Bisect-II	Brent	Dinkel-I	Dinkel-II	Gugat
Dino	36	328	2663	6(9)	21(9)	11(5)	9(4)	9(4)	8(4)
Oxford	11	737	4035	12(12)	20(9)	62(28)	84(30)	30(10)	10(4)
Temple 1	43	4233	29163	180(11)	356(9)	226(5)	298(7)	199(5)	121(3)
Temple 2	103	8063	63373	439(11)	512(5)	558(5)	842(8)	566(5)	315(3)
Trevi	58	4054	15085	123(13)	156(8)	229(13)	743(30)	130(6)	33(2)

Table 2. Runtimes for L_2 norm reprojection error. All times are in seconds. The number in the parentheses indictes the number of times Q_γ or P_γ was solved. Parameter settings, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.001$, $\sigma = 1e6$.

Acknowledgements

The authors are grateful to Prof. Paul Tseng for several useful discussions, Erling Andersen for his help with implementing the algorithms in MOSEK and Kristin Branson for reading several drafts of the paper.

This work was supported in part by National Science Foundation grant IIS-0743635, the Office of Naval Research, Microsoft, and the UW Animation Research Labs.

References

[1] Photo Tourism. http://phototour.cs.washington.edu.

[2] I. Barrodale, M. Powell, and F. Roberts. The differential correction algorithm for rational l_∞-approximation. *SIAM J. on Num. Anal.*, 9(3):493–504, 1972.

[3] A. Barros, J. Frenk, S. Schaible, and S. Zhang. A new algorithm for generalized fractional programs. *Math. Prog.*, 72(2):147–175, 1996.

[4] J. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Problems*. Springer, 2000.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] R. Brent. *Algorithms for Minimization Without Derivatives*. Courier Dover Publications, 2002.

[7] J. Crouzeix and J. Ferland. Algorithms for generalized fractional programming. *Math. Prog.*, 52(1):191–207, 1991.

[8] J. P. Crouzeix, J. A. Ferland, and S. Schaible. An algorithm for generalized fractional programs. *J. of Opt. Theory and Appl.*, 47:35–49, 1985.

[9] W. Dinkelbach. On nonlinear fractional programming. *Man. Sci.*, 13(7):492–498, 1967.

[10] J. Frenk and S. Schaible. *Fractional Programming*. Springer, 2004.

[11] R. Freund and F. Jarre. An interior-point method for fractional programs with convex constraints. *Math. Prog.*, 67(1):407–440, 1994.

[12] M. Gugat. A fast algorithm for a class of generalized fractional programs. *Man. Sci.*, 42(10):1493–1499, 1996.

[13] R. Hartley and F. Schaffalitzky. l_∞ Minimization in geometric reconstruction problems. In *CVPR*, pages 504–509, 2004.

[14] T. Ibaraki. Parametric approaches to fractional programs. *Math. Prog.*, 26(3):345–362, 1983.

[15] F. Kahl. Multiple view geometry and the L_∞-norm. In *ICCV*, pages 1002–1009, 2005.

[16] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. In *ICCV*, pages 986–993, 2005.

[17] H. Li. A practical algorithm for l_∞ triangulation with outliers. In *CVPR*, 2007.

[18] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *CVPR*, 2007.

[19] MOSEK ApS, Denmark. *The MOSEK optimization tools manual Version 5.0 (Revision 60)*.

[20] Y. Nesterov and A. Nemirovskii. An interior-point method for generalized linear-fractional programming. *Math. Prog.*, 69(1):177–204, 1995.

[21] C. Olsson, A. Eriksson, and F. Kahl. Efficient optimization for l_∞ problems using pseudoconvexity. In *ICCV*, 2007.

[22] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. pages 519–526, 2006.

[23] Y. Seo and R. Hartley. A fast method to minimize L_∞ error norm for geometric vision problems. In *ICCV*, 2007.

[24] K. Sim and R. Hartley. Recovering camera motion using l_∞ minimization. In *CVPR*, pages 1230–1237, 2006.

[25] K. Sim and R. Hartley. Removing outliers using the l_∞-norm. In *CVPR*, pages 485–494, 2006.

[26] N. Snavely, S. Seitz, and R. Szeliski. Photo Tourism: Exploring photo collections in 3D. *TOG*, 25(3):835–846, 2006.

[27] J. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Opt. Meth. and Soft.*, 11-12:625–653, 1999.