A Novel Fast Method for L_∞ Problems in Multiview Geometry

Zhijun Dai[1], Yihong Wu[3], Fengjun Zhang[1], and Hongan Wang[1,2]

[1] Intelligence Engineering Lab
[2] State Key Lab of Computer Science
Institute of Software, Chinese Academy of Sciences
[3] National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences
zhijun@iscas.ac.cn,yhwu@nlpr.ia.ac.cn,{zfj,wha}@iel.iscas.ac.cn

Abstract. Optimization using the L_∞ norm is an increasingly important area in multiview geometry. Previous work has shown that globally optimal solutions can be computed reliably using the formulation of generalized fractional programming, in which algorithms solve a sequence of convex problems independently to approximate the optimal L_∞ norm error. We found the sequence of convex problems are highly related and we propose a method to derive a Newton-like step from any given point. In our method, the feasible region of the current involved convex problem is contracted gradually along with the Newton-like steps, and the updated point locates on the boundary of the new feasible region. We propose an effective strategy to make the boundary point become an interior point through one dimension augmentation and relaxation. Results are presented and compared to the state of the art algorithms on simulated and real data for some multiview geometry problems with improved performance on both runtime and Newton-like iterations.

1 Introduction

In recent years, the L_∞ norm has been increasingly used in many kinds of computer vision problems. In multiview geometry, the L_∞ norm is geometrically meaningful. The L_∞ norm error means how well a geometric configuration explains measurements by its maximal reprojection residual. As pointed out in [1], the L_∞ norm is perhaps equally justified compared with the L_2 norm for a noise model on measurements. The L_∞ norm error assumes the noise follows a uniform distribution between 0 and an error bound. This is true in some cases of image measurements. For example in a discrete image, the pixel precision may be hard to arrive at the sub-pixel level but it is accurate in pixel unit. Thus the measurement error has equal chance to be 0 to 1 pixel, but has no chance to be more than one pixel unit.

Using the L_∞ norm for the noise model of measurements has many advantages. First, some classical problems of multiview geometry have unique single minima with the L_∞ norm, such as triangulation and camera resectioning. Second, efficient optimization solvers are available for lots of problems in the form of L_∞ norm. Thus it is commonly acknowledged that the running time of L_∞ norm is between algebraic solutions and

L_1 or L_2 norm based solutions. The efficiency of L_∞ solutions is a key factor for its popularity and usability, especially in some real time cases. It is also the basis to make more problems globally optimal solvable in a large scale, such as the relative pose problem.

In previous research, many L_∞ problems are formulated in the form of *quasi-convex* (or strongly *pseudo-convex*) programming. With this formulation, efficient algorithms are immediately available from the field of mathematical optimization. Quasi-convex is not new and has been extensively researched for decades. Apparently there is no hope to speed up the matured solvers any more. On the contrary of belief, we will show there is still some room left to make it run faster for some vision problems.

The existing algorithms for mini-max problems are mostly based on self-dual linear or second-order-cone programming (SOCP). They heavily depends on general solvers from self-dual programming. In this paper, we will propose a fast specific solver for the L_∞ norm minimization.

2 Related Works

Hartley and Schaffalitzky [2] proposed to use the L_∞ cost function in geometric vision problems. Since then there has been a booming interest in the L_∞ norm. The solved vision problems under the L_∞ formulation have been continuously growing, such as camera localization [3], camera motion recovery [4], and calibration [5]. Almost at the same time, Kahl, and Ke & Kanade have found many L_∞ minimization problems are of quasi-convex [6,7]. After that, Olsson *et al.* [8] found a stronger property for these problems, which is pseudo-convex. Based on this property, they proposed two methods to speed up the standard bisection algorithm. One is a solution based on KKT conditions. They use LOQO software to search global optima based on the conditions, but it suffers convergence problem while optimizing over variables with a large dimension. Another one is a special case of Dinkelbach type algorithms [9] which is discovered by Agarwal *et al.* [10]. This discovery brought us to the field of *generalized fractional programming* [11]. Among algorithms in this research field, they recommended to use Gugat's algorithm [12] as a standard algorithm for L_∞ optimization.

The development of generalized fractional programming could be traced back to the landmark work of [13] which gave rise to the field of interior-point methods. Modern optimization solvers are focused mainly on elegant primal-dual methods which is a powerful class of interior-point methods [14]. The mainstream of research on generalized fractional programming is using generic solvers such as SeDuMi [15]. Nevertheless, there have been interior point methods developed specially for fractional programs [16,17] for more than a decade. Unfortunately, these specialized methods have only comparable performance with the Dinkelbach type algorithms and no significant speedup has been found.

Besides solving full optimization problems, another strategy of solving L_∞ problems is based on the fact that only a small subset of measurements constrains the solution. Seo & Hartley has proposed an iterative algorithm to construct the supporting set of optimal solution [18]. The efficiency of the algorithm depends on the distribution of L_∞ residuals which is observed in [10]. In [19][20], it has been shown that the number of measurements in the supporting set is no more than $n+1$, where n is the dimension of $\boldsymbol{x}$ for problem $\min_{\boldsymbol{x}} \max_i f_i(\boldsymbol{x})$, and this has been further explained by the number of extremal points in the simplex solution of linear programming (LP) [21]. Also, the randomized algorithms of [19] work well on triangulation problem, but the complexity of subset selecting grows (sub)exponentially with n.

Finally, works on the approximation for L_∞ norm are interesting as well. In [22], a smooth function has been proposed to approximate the L_∞ norm gradually for addressing its non-differentiability. They use gradient-descent method to optimize the smooth function and the algorithm is fast to get approximate solutions for triangulation. In our practice, we observe that gradient-descent performs well for small dimensional problem, but the performance on large dimensional problem suffers slow convergence very often. With the popularity of the L_∞ norm, new practical formulations similar to this norm are also promising [23]. In this paper, we limit our scope on the L_∞ norm.

3 Problem Background

In this section, we review some parametric formulations [10] on L_∞ problems in multiview geometry and give a brief description of the underlying idea of our method.

In [8][21], it has been shown that many multiview geometry problems (triangulation, known-rotation) could be written in the following minimax form,

$$\min_{\boldsymbol{x}} \max_i \frac{\|(\boldsymbol{a}_{i1}^T\boldsymbol{x}+b_1, \boldsymbol{a}_{i2}^T\boldsymbol{x}+b_2)\|}{\boldsymbol{a}_{i3}^T\boldsymbol{x}+b_3} \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{a}_{i3}^T\boldsymbol{x}+b_3 > 0, \quad i=1,...,m \tag{2}$$

where $\boldsymbol{x}$ and $\boldsymbol{a}_{ij}, j=1,2,3$ belong to $\mathbb{R}^n$. The norm $\|\cdot\|$ is usually L_∞, L_1, or L_2 norm. Using L_∞ or L_1 norm as reprojection error provides four linear constraints from a measurement, and the L_2 norm error gives a second-order cone constraint from a measurement. Constraints (2) come from the requirement of the depth of a 3D point should be positive. We use $\mathcal{X}$ to denote the polyhedron which is the feasible region of (2). Then (1-2) is a special case of a more general version (Let $\boldsymbol{g}_2(\boldsymbol{x})$ be the denominator of (1) and $\boldsymbol{g}_1(\boldsymbol{x})$ be the numerator of (1), then (1)-(2) is converted to O.)

$$O : \min_{\boldsymbol{x},\gamma} \; \gamma \tag{3}$$

$$\text{s.t.} \; \boldsymbol{g}_1(\boldsymbol{x}) - \gamma \boldsymbol{g}_2(\boldsymbol{x}) \preceq \mathbf{0} \tag{4}$$

$$\boldsymbol{x} \in \mathcal{X} \;\; \gamma \geq 0, \tag{5}$$

where $\boldsymbol{g}_1(\boldsymbol{x}) - \gamma \boldsymbol{g}_2(\boldsymbol{x}) \preceq \mathbf{0}$ is a short form of m inequalities $g_{1i}(\boldsymbol{x}) - \gamma g_{2i}(\boldsymbol{x}) \leq 0, \forall i = 1,\ldots,m$. Also, $\boldsymbol{g}_1(\boldsymbol{x})$ is convex and $\boldsymbol{g}_2(\boldsymbol{x})$ is concave. Thus the set $S_\gamma = \{\boldsymbol{x} | \boldsymbol{g}_1(\boldsymbol{x}) - \gamma \boldsymbol{g}_2(\boldsymbol{x}) \preceq \mathbf{0}\}$ is a convex set for fixed γ. For the bisection algorithm, each iteration solves the following convex problem,

$$P_\gamma : \text{Find } \boldsymbol{x} \tag{6}$$

$$\text{s.t.} \; \boldsymbol{g}_1(\boldsymbol{x}) - \gamma \boldsymbol{g}_2(\boldsymbol{x}) \preceq \mathbf{0} \tag{7}$$

$$\boldsymbol{x} \in \mathcal{X}. \tag{8}$$

Previously proposed algorithms need to solve a series of feasibility problem P_γ, and P_γ is formulated as Q_γ,

$$Q_\gamma : \min_{\boldsymbol{x},w} w \tag{9}$$

$$\text{s.t. } \boldsymbol{g}_1(\boldsymbol{x}) - \gamma \boldsymbol{g}_2(\boldsymbol{x}) \preceq w\mathbf{1} \tag{10}$$

$$\boldsymbol{x} \in \mathcal{X}. \tag{11}$$

The problem Q_γ could be solved by primal-dual interior point methods. We use $(\boldsymbol{x}^*, w^*)$ to represent the solution of the problem Q_{γ^*} for a given γ^*. If $w^* \leq 0$, then the problem P_{γ^*} is feasible, otherwise it is infeasible.

Previously, it has been thought we could not vary γ in Newton steps due to it is not on the *central path*. For methods like bisection, Dinkelbach, or Gugat, they need to solve problems of Q_γ independently. Suppose a sequence of γ for problem Q_γ is $(\gamma_1, \gamma_2, ...)$. The solution of Q_{γ_k} will not be used for $Q_{\gamma_{k+1}}$ in these methods. Therefore every single convex optimization problem Q_{γ_k} needs a standard iterative procedure to get the solution. Although we could stop the iteration once we get $w \leq 0$ in bisection, it cannot save much iterations when γ is near optimal. This is because the feasible region becomes small with γ approximating to optimal, and the minimized w will be near to zero. In Dinkelbach or Gugat algorithm, the minimized w is solved such that γ could be reduced as much as possible from the solution of Q_γ. In the kth iteration of Dinkelbach algorithm, $(\boldsymbol{x}_k, w_k)$ is obtained by solving Q_{γ_k}, then γ is updated as

$$\gamma_{k+1} = \max_i \frac{g_{1i}(\boldsymbol{x}_k)}{g_{2i}(\boldsymbol{x}_k)}. \tag{12}$$

This updating is aggressive and make $\boldsymbol{x}_k$ locate on the boundary of the feasible region of $P_{\gamma_{k+1}}$. Thus $\boldsymbol{x}_k$ is not used for the initialization of $Q_{\gamma_{k+1}}$.

Intuitively, if we relax γ_{k+1} a little bit to a bigger value $\gamma_{k+1} + \epsilon$, the feasible region of $Q_{\gamma_{k+1}+\epsilon}$ will be expanded and $\boldsymbol{x}_k$ will be an interior point for the problem $P_{\gamma_{k+1}+\epsilon}$, and $(\boldsymbol{x}_k, 0)$ will be an interior point for the problem $Q_{\gamma_{k+1}+\epsilon}$. This is our original idea on how to connect the two optimization problems Q_{γ_k} and $Q_{\gamma_{k+1}}$. The relaxation and updating strategy will be further explored in subsection 4.2.

4 Our Method

In this section, we will propose a method to vary γ in Newton-like steps of primal-dual interior point algorithm.

The key of our method is how to get an updating step toward the optimal solution from current point $\boldsymbol{x}_k$ in the problem O. After moving forward a step, we get a new upper bound from the updated location of $\boldsymbol{x}$. A direct way to get a good step is computing the optimal solution of Q_{γ_k}, but it costs lots of Newton-like iterations. Among these iterations, the first step is usually the largest and makes a significant progress towards the optimal solution, and other steps will move slowly compared with the first step. The optimal solution of Q_{γ_k} usually has a distance with the optimal solution of O when the feasible region is not small. Thus we update $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ after the first Newton-like step which is computed based on the formulation of Q_{γ_k}.

Next we will propose to use a dual infeasible step for Q_γ, and the updating strategy for γ will be introduced after this.

4.1 Dual Infeasible Step for Q_γ

To introduce how we compute an updating step from a point $(\boldsymbol{x}, w)$ for problem Q_γ, let us consider a more general convex programming problem.

$$Q_g : \min_{\mathbf{x}} f_0(\mathbf{x}) \tag{13}$$

$$\text{s.t. } \boldsymbol{f}(\mathbf{x}) \preceq \mathbf{0}, \tag{14}$$

where f_0 is a convex objective function, $\boldsymbol{f}(\mathbf{x})$, which is composed of m functions$(f_1, ..., f_m)$, is a mapping from a $n+1$ dimensional variable $\mathbf{x}$ to a m dimensional variable. The problem Q_γ is a special case of the problem Q_g, and here the symbol $\mathbf{x}$ in Q_g represents $(\boldsymbol{x}, w)$ in Q_γ. In the following context, $\mathbf{x}$ is interchangeable with $(\boldsymbol{x}, w)$. The Lagrangian function of Q_g is

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f_0(\mathbf{x}) + \boldsymbol{\lambda}^T(\boldsymbol{f}(\mathbf{x})). \tag{15}$$

We also assume constraints (14) are convex and KKT conditions are applicable, then an optimal solution$(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ satisfies

$$Q_g \text{ KKT} : \quad \frac{\partial L}{\partial \mathbf{x}^*} = \mathbf{0} \tag{16}$$

$$\boldsymbol{\lambda}^* \circ (\boldsymbol{f}(\mathbf{x}^*)) = \mathbf{0} \tag{17}$$

$$\boldsymbol{f}(\mathbf{x}^*) \preceq \mathbf{0} \tag{18}$$

$$-\boldsymbol{\lambda}^* \preceq \mathbf{0}, \tag{19}$$

where the symbol $\circ$ is a Hadamard (entry-wise) product operator. Given an initial value of $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$, we want to solve KKT equations for the updating of $(\mathbf{x}, \boldsymbol{\lambda})$.

Next, we customize the long-step strategy in [14] for our procedure.

Step with Centering-Corrector. For the notation, the superscript $\cdot^a$ denotes *affine scaling* step. Here the *affine scaling* means the different scaling along the steps of primal and dual variables. The symbol Δ with a variable means the updating for the variable. From KKT conditions of Q_g, we want to make the updating steps satisfy the conditions

$$\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}_0 + \Delta\mathbf{x}^a, \boldsymbol{\lambda}_0 + \Delta\boldsymbol{\lambda}^a) = \mathbf{0} \tag{20}$$

$$(\lambda_{0i} + \Delta\lambda_i^a) f_i(\mathbf{x}_0 + \Delta\mathbf{x}^a) = 0, \; i = 1, ..., m. \tag{21}$$

The *surrogate duality gap* $\hat{\mu}$ for any $\mathbf{x}$ satisfies $\boldsymbol{f}(\mathbf{x}) \prec \mathbf{0}$ and $\boldsymbol{\lambda} \succeq \mathbf{0}$, is

$$\hat{\mu} = -\boldsymbol{\lambda}^T \boldsymbol{f}(\mathbf{x})/m. \tag{22}$$

It would be the duality gap μ if $\boldsymbol{\lambda}$ were dual feasible.

Using the first order approximation for (20-21), we could get the following equations,

$$\begin{bmatrix} \nabla^2 f_0(\mathbf{x}) + \Sigma_{i=1}^m \lambda_i \nabla^2 f_i(\mathbf{x}) & D\boldsymbol{f}(\mathbf{x})^T \\ -\mathrm{diag}(\boldsymbol{\lambda}) D\boldsymbol{f}(\mathbf{x}) & -\mathrm{diag}(\boldsymbol{f}(\mathbf{x})) \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}^a \\ \Delta \boldsymbol{\lambda}^a \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_c \end{bmatrix}, \tag{23}$$

where

$$\boldsymbol{r}_d = \nabla f_0(\mathbf{x}_0) + J^T \boldsymbol{\lambda}_0 \tag{24}$$

is called *dual residual*, and $\boldsymbol{r}_c$ is called *centrality residual* by adding a centering corrector term,

$$\boldsymbol{r}_c = -\boldsymbol{f}(\mathbf{x}_0) \circ \boldsymbol{\lambda}_0 - \frac{\hat{\mu}}{t}\mathbf{1}, \tag{25}$$

where the weights between centering and reducing duality gap are controlled by t. For L_1 or L_∞ norm residual, Q_γ is a LP, the affine scaling step is very efficient and we set a large $t = 10$ in this case. For L_2 norm residual, Q_γ is a SOCP, and make the current step towards more to current central path achieving fast convergence, we set $t = 2$ for this case.

Let $H = \nabla^2 f_0(\mathbf{x}) + \Sigma_{i=1}^m \lambda_i \nabla^2 f_i(\mathbf{x}), J = D\boldsymbol{f}(\mathbf{x})$, $C = -\mathrm{diag}(\boldsymbol{f}(\mathbf{x}))^{-1}$, $S = -\mathrm{diag}(\lambda) D\boldsymbol{f}(\mathbf{x}) C$, then we have

$$(H - J^T S J)\Delta \mathbf{x}^a = -\boldsymbol{r}_d + J^T C \boldsymbol{r}_c. \tag{26}$$

After get $\Delta \mathbf{x}^a$ by solving this normal equations, $\Delta \boldsymbol{\lambda}^a$ is got by back substitution

$$\Delta \boldsymbol{\lambda}^a = -C\boldsymbol{r}_c - SJ\Delta \mathbf{x}^a. \tag{27}$$

The affine step lengths along directions of $\Delta \mathbf{x}^a$ and $\Delta \boldsymbol{\lambda}^a$ are

$$\alpha_{\mathbf{x}}^a = \underset{\alpha}{\mathrm{argmax}}\{\alpha \in [0,1] | \boldsymbol{f}(\mathbf{x} + \alpha \Delta \mathbf{x}^a) \preceq \mathbf{0}\}, \tag{28}$$

$$\alpha_{\boldsymbol{\lambda}}^a = \underset{\alpha}{\mathrm{argmax}}\{\alpha \in [0,1] | \boldsymbol{\lambda} + \alpha \Delta \boldsymbol{\lambda}^a \succeq \mathbf{0}\}. \tag{29}$$

Equations (23) may be solved efficiently when $H - J^T S J$ is sparse in some computer vision applications. Using the sparsity in L_∞ norm minimization for structure and motion reconstruction has been proposed in [24,25].

4.2 Relaxation

The central path $\mathcal{C}$ of Q_γ is a set of primal dual points $(\mathbf{x}, \boldsymbol{\lambda})$ satisfy

$$-\boldsymbol{\lambda} \circ \boldsymbol{f}(\mathbf{x}) = \mu \mathbf{1}. \tag{30}$$

It is unnecessary that a sequence of $\mathbf{x}$ generated in iterations flows exactly along the central path. We only need to make the sequence along a neighborhood of central path for convergence, and this is the main reason attributed to the success of primal-dual method. We use a very large neighborhood around the central path which is represented as $\mathcal{N}_{-\infty}(\zeta)$,

$$\mathcal{N}_{-\infty}(\zeta) = \{(\mathbf{x}, \boldsymbol{\lambda}) | -\lambda_i f_i(\mathbf{x}) \geq \zeta\mu, \text{for all } i = 1, ..., m\}, \tag{31}$$

where $\zeta \in (0,1)$. For each iteration, we need to set $(\gamma, \mathbf{x}, \boldsymbol{\lambda})$ to guarantee $(\mathbf{x}, \boldsymbol{\lambda})$ in a region $\mathcal{N}_{-\infty}(\zeta)$ such that the procedure could converge. The region size of $\mathcal{N}_{-\infty}(\zeta)$ is controlled by the parameter ζ. A simple method of relaxation is to add a small value on γ, and then ζ value could be computed from the γ relaxation by equation (31). This strategy will generate neighborhoods for a sequence of different ζ value.

Back to the problem Q_γ, we set w to zero for each new iteration. In the kth iteration, $(\mathbf{x}_k, \lambda_k)$ comes from the step

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha^a_{\mathbf{x}_{k-1}} \Delta\mathbf{x}^a_{k-1}, \tag{32}$$

$$\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_{k-1} + \alpha^a_{\boldsymbol{\lambda}_{k-1}} \Delta\boldsymbol{\lambda}^a_{k-1}. \tag{33}$$

For the γ sequence, we set

$$\gamma_k = \min\{\max_i \frac{g_{1i}(\boldsymbol{x}_k)}{g_{2i}(\boldsymbol{x}_k)}, \gamma_{k-1}\}, \tag{34}$$

which makes P_{γ_k} be always feasible. Assume we relax γ_k to

$$\gamma_k^{'} = \gamma_k + \Delta\gamma_k, \tag{35}$$

then new steps for $(\mathbf{x}_k, \boldsymbol{\lambda}_k)$ could be got by the equations (23) which is derived from $Q_{\gamma_k^{'}}$.

The relax strategy described above works well in our practice. However, we found a more sophisticated strategy. For the problem Q_{γ_k}, if the γ_k is near optimal, the solution of w will be close to zero. The finally optimal w is pre-known to be zero. The point $((\boldsymbol{x}_k, 0), \boldsymbol{\lambda}_k)$, where $\boldsymbol{x}_k \in \mathcal{X}$, is not on the region of $\mathcal{N}_{-\infty}(\zeta)$, because $(\boldsymbol{x}_k, 0)$ makes some constrains be strictly equal to zero from last updating. Instead of relaxing on γ, we could modify w value so that $((\boldsymbol{x}_k, w), \boldsymbol{\lambda}_k)$ belongs to $\mathcal{N}_{-\infty}(\zeta)$. The modification on $\mathbf{x}$ has no influence on $\boldsymbol{x}$. The $(\boldsymbol{x}_k, w_k)$ is feasible for any $w_k \geq 0$. In each step, if $\boldsymbol{\lambda}_k$ is dual feasible,

$$w_k^{'} = \zeta\mu_k / \max_i \lambda_i + \max_i(f_i(\boldsymbol{x}_k, 0)), \tag{36}$$

which guarantee $((\boldsymbol{x}_k, w_k^{'}), \boldsymbol{\lambda}_k) \in \mathcal{N}_{-\infty}(\zeta)$. However, in each iteration we have a dual residual

$$\boldsymbol{r}_{\boldsymbol{\lambda}_k} = \nabla f_0(\mathbf{x}_k) + J^T \boldsymbol{\lambda}_k \tag{37}$$

which is not equal to $\mathbf{0}$. Thus we use the surrogate gap to replace the unknown duality gap in (36) to approximate $w_k^{'}$,

$$w_k = \zeta\hat{\mu}_k / \max_i \lambda_i + \max_i(f_i(\boldsymbol{x}_k, 0)). \tag{38}$$

This cannot guarantee $((\boldsymbol{x}_k, w_k), \boldsymbol{\lambda}_k) \in \mathcal{N}_{-\infty}(\zeta)$, but it will make $((\boldsymbol{x}_k, w_k), \boldsymbol{\lambda}_k)$ in a similar neighborhood with an unknown ζ parameter.

The term $\max_i(f_i(\boldsymbol{x}_k, 0))$ means the maximum infeasibility for $\boldsymbol{x}_k$ in constraints. In the initialization, it may be bigger than zero for a random starting point, such as a point behind a camera in the triangulation problem. After the initialization, the point will move to feasible region quickly and the maximum infeasibility will always be zero.

4.3 Algorithm

Now we are ready to present our complete method in Algorithm 1.

Stop Criterion. The condition in line 27 is the stop criterion. When $\boldsymbol{r}_d$ is very small, the surrogate gap $\hat{\mu}$ is close to the duality gap. Therefore, if $\hat{\mu}$ is near to zero, current solution $(\boldsymbol{x}^*, w^*)$ of Q_{γ^*} will be near optimal. Under this condition, if w^* is near to zero, then the γ^* will be near optimal. This criterion guarantees the solution $(\mathbf{x}, \gamma)$ arrives at a desired precision through controlling ϵ_1 and ϵ_2.

Initialization. Varying γ in the procedure could take advantage of a good initial point. Previous methods use a dummy initial point. We make the solution approaches to the global optimum gradually. We recommend to use an algebraic solution for an initial point. The time complexity of an algebraic solution is comparable with one Newton step in other geometrically meaningful solutions. In our experiments, we solve the equations

$$\boldsymbol{g}_1(\boldsymbol{x}) = \mathbf{0}. \tag{39}$$

This is over-determined, and we compute its closed-form linear least-squares solution. However, an algebraic solution may not in χ of Q_γ in very rare cases. For this situation, we recommend to employ standard primal-dual algorithm to solve the feasibility problem

$$\text{Find} \quad \boldsymbol{x} \tag{40}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{X}, \tag{41}$$

and the algebraic solution is used as its initial point.

Our algorithm is always dual infeasible until stopped. It is always primal feasible if it starts from a feasible point. Otherwise it will take several steps to arrive at a primal feasible point, and in subsequent steps it will be always feasible.

Outputs. We output both $\boldsymbol{x}$ and dual variables $\boldsymbol{\lambda}$. The dual variables $\boldsymbol{\lambda}$ are useful. The top non-zero values (at most $n+1$) of $\boldsymbol{\lambda}$ correspond to the supporting set of constrains. Furthermore, the max number of supporting set can also be clearly explained from the complementary slackness between constraints and dual variables.

Scaling. The Q_γ we deal with is a scaled version by $\boldsymbol{g}_2(\boldsymbol{x}_k)$ which is the same in Dinkelbach type II algorithm.

Convergence Analysis. The code in lines 12-14 is an adjustment strategy. In the case of making a small step on the last iteration, it guarantees the relaxed amount r no less than c_1 so that the relaxed point $((\boldsymbol{x}, w), \boldsymbol{\lambda})$ has an enough large region to move. Assume the relaxed point $((\boldsymbol{x}, w), \boldsymbol{\lambda})$ locates on the boundary of $N_{-\infty}(\zeta)$, then the relation between relaxed amount r and ζ is proportional

$$r \propto \zeta. \tag{42}$$

With a bigger ζ, the next iteration will keep the point move along more towards the central path. In experiments, we set $c_1 = 10^{-4}$ for L_1 and L_∞ norm residual, and $c_1 = 10^{-2}$ for L_2 norm residual. The condition $w_k \geq c_2$ in line 15 prevents equations (23) from getting a bad conditional number (we set $c_2 = 10^{-6}$). Lines 9-30 is a loop

Algorithm 1: Relax algorithm

```
Input: $\boldsymbol{x}_0$, $u_0$(upper bound of $\gamma$), functions $\boldsymbol{g}_1(\boldsymbol{x})$ and $\boldsymbol{g}_2(\boldsymbol{x})$
Output: $\boldsymbol{x}$, $\boldsymbol{\lambda}$, $\gamma$
begin
    $k \leftarrow 0,\ w_k \leftarrow 0,\ \boldsymbol{\lambda}_k \leftarrow \mathbf{1}, \alpha_{prev} \leftarrow 1$;
    $\gamma_k \leftarrow u_0$;
    if $x_0 \in \mathcal{X}$ then
        $\gamma_k \leftarrow \min\{\max_i g_{1i}(\boldsymbol{x}_k)/g_{2i}(\boldsymbol{x}_k), \gamma_k\}$;
    end
    while true do
        Setting up coefficient for $Q_{\gamma_k}$;
        repeat
            $\hat{\mu}_k \leftarrow \boldsymbol{\lambda}_k^T|\boldsymbol{f}(\mathbf{x}_k)|$;
            $r \leftarrow \zeta\hat{\mu}_k/\max_i \lambda_i$;
            if $\alpha_{prev} < \alpha_T$ then
                $r \leftarrow \max\{r, c_1\}$;
            end
            $r \leftarrow \max\{w_k, c_2\}$;
            $w_k \leftarrow r + \max_i(f_i(\boldsymbol{x}_k, 0))$;
            Compute $\Delta\mathbf{x}_k^a, \Delta\boldsymbol{\lambda}_k^a$ from (23);
            Compute $\alpha_{\mathbf{x}_k}^a, \alpha_{\boldsymbol{\lambda}_k}^a$ from (28-29);
            $\alpha_{prev} \leftarrow \alpha_{\mathbf{x}_k}^a$;
            $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \beta\alpha_{\mathbf{x}_k}^a \Delta\mathbf{x}_k^a$;
            $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \beta\alpha_{\boldsymbol{\lambda}_k}^a \Delta\boldsymbol{\lambda}_k^a$;
            $\mathbf{x}_k \leftarrow \mathbf{x}_{k+1},\ \boldsymbol{\lambda}_k \leftarrow \boldsymbol{\lambda}_{k+1}$;
            $\gamma_{k+1} \leftarrow \gamma_k$;
            if $\boldsymbol{x}_k \in \mathcal{X}$ then
                $\gamma_{k+1} \leftarrow \max_i g_{1i}(\boldsymbol{x}_k)/g_{2i}(\boldsymbol{x}_k)$;
            end
            if $|w_k| < \epsilon_1$ and $\hat{\mu}_k/m < \epsilon_2$ and $\gamma_{k+1} - \gamma_k < \epsilon_1$ and $\|\boldsymbol{r}_d\|_2/m < \epsilon_1$ then
                return $\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \gamma_k$;
            end
        until $\gamma_{k+1} < \gamma_k$;
        $k \leftarrow k + 1$;
    end
end
/* In our current implementation, we set $\zeta = 0.1$. Parameter $\beta$
   is close to one, and we set $\beta = 0.995$ to prevent the
   duality pair $-f_i\lambda_i$ from being strictly equal to zero. The
   parameter $\alpha_T$ is set to 0.1 for $L_1$ and $L_\infty$ norm residual,
   and it is set to 0.2 for $L_2$ norm residual.                        */
```

to make sure from current point $\boldsymbol{x}_k$ to move a point with a smaller γ. It is observed that the loop usually takes one iteration to reduce γ value. It may take more than one iteration from an initial point with an unrealistic large γ value ($\gamma = \infty$ if $\boldsymbol{x} \notin \mathcal{X}$, and we set $\gamma = u_0$ if $\gamma > u_0$), then the loop will work like the long-step path following algorithm (see [26] or chapter 5 in [14]) to get a point with reduced γ. The process of our algorithm produces a non-increasing sequence of γ, each iteration is a Newton-like step piercing into the current feasible region of P_γ. The algorithm is always convergent in our experiments, but rigorous proof is not pursued in this paper.

5 Experiments

In this section we report some experiments carried out to evaluate our method. Currently, the algorithm is implemented with Matlab.

Three algorithms were compared. Bisection refers to the implementation of Bisect II from [10]. Dinkelbach II refers to Dinkelbach procedures of type II. Gugat refers to Gugat's algorithm [12]. Kernel solvers of the three algorithms are implemented using SeDuMi [15]. In experiments, we found our algorithm can always get a precision of optimal γ at least ϵ_1 by setting $\epsilon_2 = 10^{-2}\epsilon_1$. We present experimental results with $\epsilon_1 = 10^{-4}$. Parameters of compared algorithms are set such that the precision of γ arrives at ϵ_1. In algorithm 1, the algebraic solution of (39) is used to give an initial point. For other algorithms, algebraic solution is used to provide an upper bound on γ value. In the implementation, the setting up of coefficients of linear or second order cone constraints is significantly efficient compared with the solver runtime, thus only the time used by the solver is noted for wiping out the influence of different implementations of the setting up. The time to get the initial algebraic solution is also not included.

Purely runtime may be influenced by computer configuration, coding language, and compiling optimization. Besides comparing execution time, we compare the total iterations for solving linear equations. It is generally recognized that the main computational cost of mathematical optimization is spent on solving linear equations [27] (See chapter 10). Different algorithms such as the Gugat and Dinkelbach need to solve linear equations with the same structure and time complexity of (23). Therefore the most important criterion for comparing the efficiency is to check how many iterations needed to get the global optimum.

5.1 Experiment on Triangulation

We use simulated data in this experiment. We randomly generate m views for a random 3D point, where $m = 50$ in the following report. The 3D point is located within a cube in front of all m views.

Our algorithm is always convergent from any initial point. Figure 1 is a running case using 20 different random initial points with the L_2 norm for measurement residual.

Figure 2 and 3 show the runtime and iterations of 100 random running cases for three kinds of residual norms. Algorithm 1 significantly outperforms previous methods on both the time and iterations. It only needs around 7 iterations to jump from algebraic solution to L_∞ solution with high precision. The runtime is in log scale. It shows that algorithm 1 is ten times faster than previous solvers.

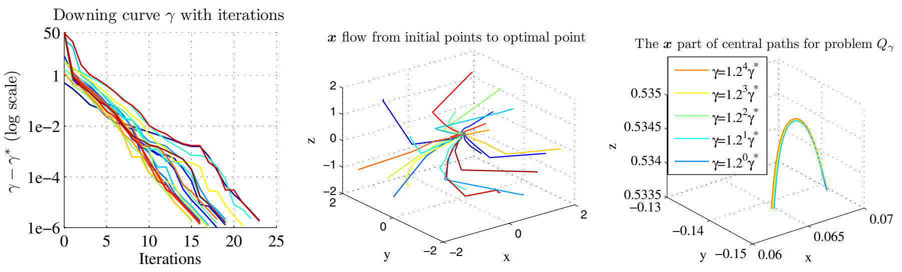

Fig. 1. This is a random generated example with an optimal $\gamma^* = 0.0212$. The initial upper bound of γ is $u_0 = 50$. The γ value is going down for each iteration; $\boldsymbol{x}$ converges to optimal point for 20 random initial points; and the $\boldsymbol{x}$ part of central paths are close each other with the downing of γ to optimal γ^*.

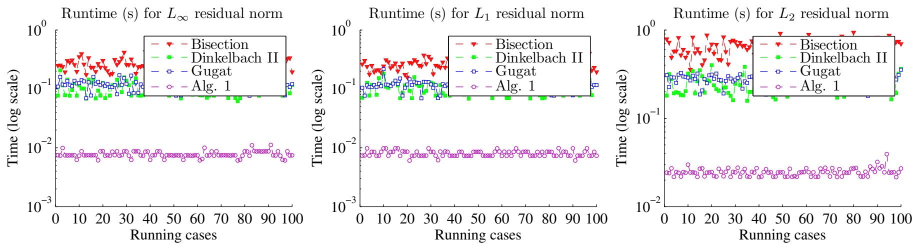

Fig. 2. Runtime for 100 random running cases

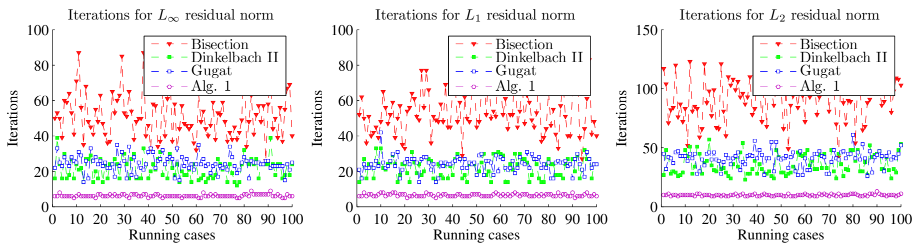

Fig. 3. Iterations for 100 random running cases

5.2 Structure and Motion Recovery with Known Camera Orientation

For SOCP formulation, Sedumi cannot solve problems in a very large scale (out of memory error). Here, we present experiments on two real standard data set, Dino and Oxford which are publicly available from the Oxford Visual Geometry Group. The Dino data set contains 36 cameras and 127 3D-points. The Oxford data set contains 11 cameras and 737 3D-points. For *Gauge freedom*, it is removed through fixing the first 3D point location and assigning the third coordinate of the first camera location to 1 as a scale of the reconstruction. The problem Q_γ running on these data sets is an optimization over 485 variables and 2240 variables for the Dino and Oxford data sets respectively. Table 1 and 2 list the compared results on runtime and iterations. Algorithm 1 saves both runtime and iterations.

Table 1. Running iterations and time on the Dino experiment. The data set contains 36 cameras and 127 3D-points which are visible in at least 2 images. The total number of measurements is 2667. Under each algorithm, the number of iterations is on the left column and the runtime is on the right column in seconds. The number in the parentheses indicates the number of times Q_γ was solved.

Residual norm	Bisection		Dinkelbach II		Gugat		Alg. 1	
L_∞	190(6)	8.22s	91(3)	3.83s	58(2)	2.46s	27	1.09s
L_1	354(14)	14.04s	75(3)	3.21s	46(2)	1.82s	28	0.92s
L_2	295(10)	62.33s	80(3)	17.32s	56(2)	11.84s	31	1.07s

Table 2. Running iterations and time on the Oxford experiment. The data set contains 11 cameras and 737 3D-points which are visible in at least 2 images. The total number of measurements is 4035. Under each algorithm, the number of iterations is on the left column and the runtime is on the right column in seconds. The number in the parentheses indicates the number of times Q_γ was solved. f denotes numerical failure.

Residual norm	Bisection		Dinkelbach II		Gugat		Alg. 1	
L_∞	164(7)	8.54s	78(4)	4.14s	88(4)	4.38s	31	1.40s
L_1	243(10)	13.15s	111(5)	5.68s	96(4)	5.44s	32	1.72s
L_2	172(7)	122.93s	172(6)	121.36s	f	f	33	1.73s

Oxford Data Set. It is very interesting that setting γ from 0.32 to 2.00 for the SOCP formulation of Q_γ, then the minimized w value is always positive and close to zero. Figure 4 is the curve of minimized w value with varying γ for the problem Q_γ running on the Oxford data set. This makes termination criterion $|w| < \epsilon$ of Gugat's algorithm failed. For this data set, the problem P_γ is infeasible for $0.32 < \gamma < 2.00$, but there exists a solution for Q_γ with huge depth of range of camera and structure configuration and with the objective function w is minimized near to zero. In the original Gugat's algorithm, the stop criterion is either $|w| < \epsilon_2$ or the gap between upper bound and lower bound of γ is less than ϵ_1. Thus the algorithm will stop running on this data set once it predicates $0.32 < \gamma < 2.00$. We fixed this through changing $|w| < \epsilon$ with an additional requirement, which is the new estimated γ should not be far away from the

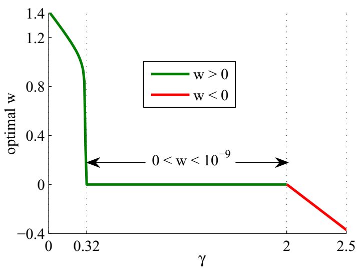

Fig. 4. The optimized result of w for the Oxford data set in the second-order cone formulation of Q_γ with varying γ from 0 to 2.50

current upper bound of γ. With the new criterion, Gugat's algorithm gets a solution with 1e-2 precision on γ and stopped due to numerical error of SeDuMi.

6 Conclusions

In this paper, we have demonstrated a fast specific solver for L_∞ problems in multiview geometry. The method is general and applicable to all the different L_∞ problems and it may also be used for generalized fractional programming in other fields.

In future, we will research on using finite termination techniques to get optimal solution of both γ and $\boldsymbol{x}$ when γ and $\boldsymbol{x}$ are advanced enough.

Acknowledgments. We thank Sameer Agarwal for providing us the source code of L_∞ toolbox [10]. Yihong Wu was supported by the National Basic Research Program of China (973 Program) under grant 2012CB316302. Zhijun Dai, Fengjun Zhang, and Hongan Wang were supported by the National Science Foundation of China (61203276, 61173059, 61135003).

References

1. Hartley, R., Kahl, F.: Optimal Algorithms in Multiview Geometry. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part I. LNCS, vol. 4843, pp. 13–34. Springer, Heidelberg (2007) 1
2. Hartley, R., Schaffalitzky, F.: L_∞ minimization in geometric reconstruction problems. In: CVPR, pp. 504–509 (2004) 2
3. Micusik, B., Pflugfelder, R.: Localizing non-overlapping surveillance cameras under the L-infinity norm. In: CVPR, pp. 2895–2901 (2010) 2
4. Sim, K., Hartley, R.: Recovering camera motion using L_∞ minimization. In: CVPR, pp. 1230–1237 (2006) 2
5. Heller, J., Havlena, M., Pajdla, T., Sugimoto, A.: Structure-from-motion based hand-eye calibration using L_∞ minimization. In: CVPR, pp. 3497–3503 (2011) 2
6. Kahl, F.: Multiple view geometry and the L_∞-norm. In: ICCV, pp. 1002–1009 (2005) 2
7. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. In: ICCV, pp. 986–993 (2005) 2

8. Olsson, C., Eriksson, A., Kahl, F.: Efficient optimization for L_∞-problems using pseudoconvexity. In: ICCV, pp. 1–8 (2007) 2,3
9. Dinkelbach, W.: On nonlinear fractional programming. Management Science, 492–498 (1967) 2
10. Agarwal, S., Snavely, N., Seitz, S.: Fast algorithms for L_∞ problems in multiview geometry. In: CVPR, pp. 1–8 (2008) 2,3,10,13
11. Frenk, J., Schaible, S.: Fractional programming. In: Handbook of Generalized Convexity and Generalized Monotonicity, pp. 335–386 (2005) 2
12. Gugat, M.: A fast algorithm for a class of generalized fractional programs. Management Science, 1493–1499 (1996) 2,10
13. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica, 373–395 (1984) 2
14. Wright, S.: Primal-dual interior-point methods, vol. 54. SIAM (1997) 2,5,10
15. Sturm, J.: Using SeDuMi 1. 02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software 11–12, 625–653 (1999) 2,10
16. Freund, R., Jarre, F.: An interior-point method for fractional programs with convex constraints. Mathematical Programming 67(1), 407–440 (1994) 2
17. Nesterov, Y., Nemirovskii, A.: An interior-point method for generalized linear-fractional programming. Mathematical Programming 69(1), 177–204 (1995) 2
18. Seo, Y., Hartley, R.: A fast method to minimize L_∞ error norm for geometric vision problems. In: ICCV, pp. 1–8 (2007) 2
19. Li, H.: Efficient reduction of L-infinity geometry problems. In: CVPR, pp. 2695–2702 (2009) 2,3
20. Olsson, C., Enqvist, O., Kahl, F.: A polynomial-time bound for matching and registration with outliers. In: CVPR, pp. 1–8 (2008) 2
21. Olsson, C., Eriksson, A., Hartley, R.: Outlier removal using duality. In: CVPR, pp. 1450–1457 (2010) 3
22. Dai, Y., Li, H., He, M., Shen, C.: Smooth approximation of L_∞-norm for multi-view geometry. In: Digital Image Computing: Techniques and Applications, pp. 339–346 (2009) 3
23. Zach, C., Pollefeys, M.: Practical Methods for Convex Multi-view Reconstruction. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 354–367. Springer, Heidelberg (2010) 3
24. Lee, H., Lee, S., Seo, Y.: Sparse second-order cone programming for 3d reconstruction. In: International Workshop on Advanced Image Technology (2009) 6
25. Seo, Y., Lee, H., Lee, S.: Sparse Structures in L-Infinity Norm Minimization for Structure and Motion Reconstruction. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 780–793. Springer, Heidelberg (2008) 6
26. Kojima, M., Mizuno, S., Yoshise, A.: A primal-dual interior point algorithm for linear programming. In: Progress in Mathematical Programming: Interior-Point and Related Methods, pp. 29–47. Springer-Verlag New York, Inc. (1988) 10
27. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004) 10