数理解析研究所講究録
第 1773 巻 2012 年 68-76

Faster Algorithms for Computer Vision

Toshiyuki Masaki and Takahito Kuno

Graduate School of Systems and Information Engineering,
University of Tsukuba, Ibaraki 305-8573, Japan

Abstract

In this paper, we make a modification to Karl and Hartley's formulation of problems in computer vision [3, 4], and show the resulting norm minimization problem can be solved by solving a sequence of LP problems. We also propose an approximation of the norm minimization problem, which reduces to one LP or SOCP problem.

Key words: Computer vision, multiple view geometry, linear programming, second-order cone programming.

1. Introduction

In recent years, problems dealt with in computer vision are larger in size and require much more computational time to solve than before. For example, while the traditional triangulation problem has only three variables, the number of variables in the latest structure-and-motion problem amounts to several hundreds in some cases. Thus, there is a great need for faster algorithms for large scale problems in the area of computer vision. In response to this, Karl and Hartley [3, 4] developed a framework for solving geometric problems, such as *triangulation*, *camera-resectioning*, *homography-estimation* and *structure-and-motion.* In their framework, those problems are formulated into an L_∞ norm minimization problem and solved by solving a sequence of second-order cone programming (SOCP) problems.

In this paper, we show that, if Karl and Hartley's formulation is slightly modified, the norm minimization problem can be solved by solving a sequence of linear programming (LP) problems. In addition, we propose an approximation of the norm minimization problem, which reduces to one LP or SOCP

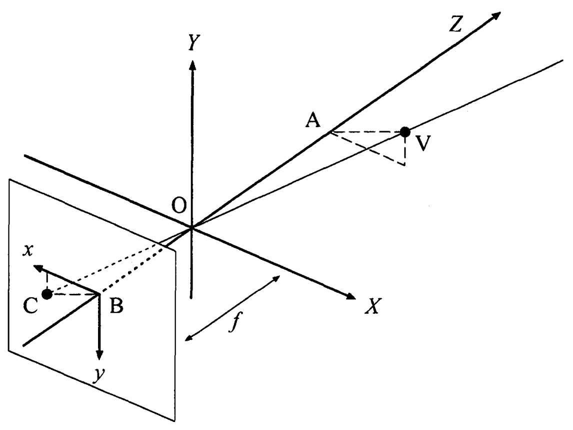

Figure 1: Geometry of a pinhole camera.

problem. Lastly, we report some numerical results, which demonstrate the superiority of our approach to Karl and Hartley's in both efficiency and accuracy.

2. Problems in computer vision

In this section, we first take *triangulation* as a typical example and illustrate how it can be formulated into an optimization problem. We also show that many other problems in computer vision take the same form. Essential to those formulations is the *pinhole camera model.*

Pinhole camera model: The pinhole camera model describes the relationship between the coordinates of a 3D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is a pinhole and no lenses are used to focus light. The geometry related to the mapping of a pinhole camera is illustrated in Figure 1. Let us denote the object point by $\mathrm{V} = (X, Y, Z)^\top$ in the 3D coordinate system with its origin at the camera aperture O. Light emanating from V passes through O and projects an inverted image $\mathbf{v} = (x, y)^\top$ on the image plane, which is parallel to the X-Y plane and located at the focal length f (> 0) from O in the negative direction of the Z axis. Let $\mathrm{A} = (0, 0, Z)^\top$, $\mathrm{B} = (0, 0, -f)^\top$ and $\mathrm{C} = (x, y, -f)^\top$. Since the triangle OAV is similar to the triangle OBC, we have $(x, y)^\top = (f/Z)(X, Y)^\top$,

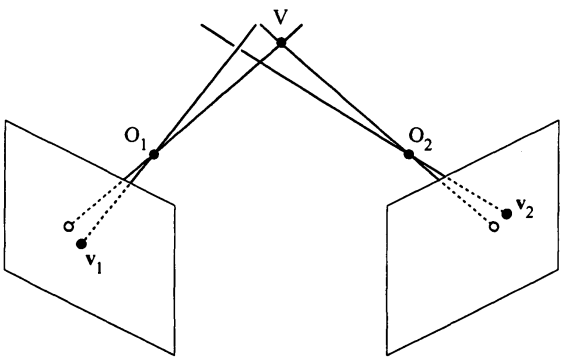

Figure 2: Triangulation using two cameras.

or equivalently

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix}$$

in homogeneous coordinates. It should be also noted that the image $\mathbf{v}$ is invariant under scaling of V. We denote this by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1}$$

and say that $(x, y, 1)^{\mathrm{T}}$ is *equivalent, or proportional, to* $(X, Y, Z/f)^{\mathrm{T}}$. The 3×4 matrix in (1) is called the *camera matrix.*

Triangulation: Triangulation (or *reconstruction*) is the process of determining the 3D coordinates of the object V, given its projection onto two, or more, images captured by pinhole cameras. In theory, the triangulation problem is quite trivial. Each image $\mathbf{v}$ of V corresponds to a half-line in the 3D space such that all points on the line are projected to $\mathbf{v}$. Therefore, V must lie on the intersection of those lines, and we must be able to calculate its coordinates analytically from a pair of different images. In practice, however, various types of noise, such as geometric noise from lens distortion or interest point detection error, lead to inaccuracies in the measured image coordinates. As a result, lines associated with different images of V do not always intersect in the 3D space, as in Figure 2.

Suppose that $\mathrm{V} = (X, Y, Z)^\mathsf{T}$ is in an arbitrary 3D coordinate system, and that there are $M$ images $\mathbf{v}_i = (x_i, y_i)^\mathsf{T}$ of V captured by cameras $i = 1, \ldots, M$. Let us denote the $i$th camera matrix by

$$\mathbf{Q}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f_i & 0 \end{bmatrix},$$

where f_i (> 0) is the focal length of camera $i$. Note that V is denoted as $\mathbf{R}_i\mathrm{V} + \mathbf{t}_i$ for some rotation matrix $\mathbf{R}_i$ and a translation vector $\mathbf{t}_i$ in the 3D coordinate system with the origin at the focal point O_i of camera $i$. Hence, from (1), we have

$$\begin{bmatrix} \mathbf{v}_i \\ 1 \end{bmatrix} \sim \mathbf{Q}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{O} & 1 \end{bmatrix} \begin{bmatrix} \mathrm{V} \\ 1 \end{bmatrix}, \quad i = 1, \ldots, M.$$

Let

$$\mathbf{P}_i = \begin{bmatrix} \mathbf{p}_i^1 \\ \mathbf{p}_i^2 \\ \mathbf{p}_i^3 \end{bmatrix} = \mathbf{Q}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{O} & 1 \end{bmatrix},$$

which is referred to as the *normalized camera matrix*, or simply as the *camera matrix*. The coordinates of the image $\mathbf{v}_i = (x_i, y_i)^\mathsf{T}$ is then given as

$$x_i = \frac{\mathbf{p}_i^1\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}}, \quad y_i = \frac{\mathbf{p}_i^2\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}},$$

where $\mathrm{U} = (\mathrm{V}^\mathsf{T}, 1)^\mathsf{T}$, if there is no noise. As mentioned above, however, this is not the case in practice, and we need to determine the coordinates $(X, Y, Z)$ of V so as to minimize the 2D *residual error*, which is defined as follows in terms of L_p norm:

$$\gamma_i = \left\| \frac{\mathbf{p}_i^1\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}} - x_i, \frac{\mathbf{p}_i^2\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}} - y_i \right\|_p, \quad i = 1, \ldots, M.$$

If we measure the magnitude of $\gamma = (\gamma_1, \ldots, \gamma_M)^\mathsf{T}$ in L_q norm, then triangulation reduces to an optimization problem with $(X, Y, Z)$, the first three components of U, as the variables:

$$\left| \begin{array}{ll} \text{minimize} & \|\gamma\|_q \\ \text{subject to} & \left\| \frac{\mathbf{p}_i^1\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}} - x_i, \frac{\mathbf{p}_i^2\mathrm{U}}{\mathbf{p}_i^3\mathrm{U}} - y_i \right\|_p = \gamma_i, \quad i = 1, \ldots, M. \end{array} \right. \tag{2}$$

Table 1: Known and unknown parameters of computer vision problems.

problem	*# cameras*	*# objects*	*known*	*unknown*
triangulation	M	1	$\mathbf{P}_i, \mathbf{v}_{i1}$	V_1
camera-resectioning	1	N	$\mathrm{V}_j, \mathbf{v}_{1j}$	$\mathbf{P}_1$
structure-and-motion	M	N	$\mathbf{R}_i, \mathbf{v}_{ij}$	$\mathbf{t}_i, \mathrm{V}_j$

Other problems: Many other computer vision problems can also be formulated into optimization problems similar to (2) except that the number of objects is usually more than one.

Suppose N points $\mathrm{V}_j = (X_j, Y_j, Z_j)^\mathsf{T}$, $j = 1, \ldots, M$, are given in the 3D space. Let $\mathbf{v}_{ij} = (x_{ij}, y_{ij})^\mathsf{T}$ denote the image of V_j captured by camera i. Also let

$$\gamma_{ij} = \left\| \frac{\mathbf{p}_i^1 \mathrm{U}_j}{\mathbf{p}_i^3 \mathrm{U}_j} - x_{ij}, \frac{\mathbf{p}_i^2 \mathrm{U}_j}{\mathbf{p}_i^3 \mathrm{U}_j} - y_{ij} \right\|_p, \quad i = 1, \ldots, M; j = 1, \ldots N,$$

where $\mathrm{U}_j = (\mathrm{V}_j^\mathsf{T}, 1)^\mathsf{T}$. In this case, the problem is written as follows

$$\left| \begin{array}{ll} \text{minimize} & \|\gamma\|_q \\ \text{subject to} & \left\| \frac{\mathbf{p}_i^1 \mathrm{U}_j}{\mathbf{p}_i^3 \mathrm{U}_j} - x_{ij}, \frac{\mathbf{p}_i^2 \mathrm{U}_j}{\mathbf{p}_i^3 \mathrm{U}_j} - y_{ij} \right\|_p = \gamma_{ij}, \quad \left\{ \begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N, \end{array} \right. \end{array} \right. \tag{3}$$

where $\gamma = (\gamma_{11}, \ldots, \gamma_{MN})^\mathsf{T}$. This seems a straightforward extension of (1), but describes various types of problems in computer vision depending on what parameters are known or unknown. Table 1 shows three major examples; e.g., the number of variables amounts to $3(M+N)$ in (3) associated with a structure-and-motion problem while that is only three in the case of triangulation.

3. Solution approaches

In [4], Kahl and Hartley proposed a bisection algorithm to solve (3) with $p = 2$ and $q = \infty$. After illustrating their algorithm, we introduce here a practical approximation approach to (3).

Bisection approach: In the usual applications of computer vision, we may assume that $\mathbf{p}_i^3 \mathrm{U}_j > 0$ for all i, j. We can therefore rewrite (3) into the following

Algorithm 1 Bisection algorithm for the case where $p = 2$ and $q = \infty$.

```
Require: an interval $[\gamma_\ell, \gamma_u]$ known to contain the optimal value $r^*$ and a
  tolerance $\epsilon > 0$.
  repeat
    $\gamma \leftarrow (\gamma_\ell + \gamma_u)/2$;
    check if (5) is feasible or not, by solving an associated SOCP problem;
    if (5) is feasible then
      $\gamma_u \leftarrow \gamma$
    else
      $\gamma_\ell \leftarrow \gamma$
    end if
  until $\gamma_u - \gamma_\ell \leq \epsilon$;
```

when $p = 2$ and $q = \infty$:

$$\left|\begin{array}{ll} \text{minimize} & \gamma \\ \text{subject to} & \|\mathbf{p}_i^1 \mathrm{U}_j - x_{ij}\mathbf{p}_i^3 \mathrm{U}_j, \mathbf{p}_i^2 \mathrm{U}_j - y_{ij}\mathbf{p}_i^3 \mathrm{U}_j\|_2 \leq \gamma \mathbf{p}_i^3 \mathrm{U}_j, \quad \left\{\begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N. \end{array}\right. \end{array}\right. \tag{4}$$

As is shown in Algorithm 1, the bisection algorithm solves (4) by checking repeatedly if the following system of inequalities is feasible for a fixed r in a given interval $[\gamma_\ell, \gamma_u]$:

$$\left\|\mathbf{p}_i^1 \mathrm{U}_j - x_{ij}\mathbf{p}_i^3 \mathrm{U}_j, \mathbf{p}_i^2 \mathrm{U}_j - y_{ij}\mathbf{p}_i^3 \mathrm{U}_j\right\|_2 \leq \gamma \mathbf{p}_i^3 \mathrm{U}_j, \quad \left\{\begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N. \end{array}\right. \tag{5}$$

Since the right-hand-sides turn into constants, the feasibility of (5) can be checked by solving an SOCP problem with (5) as constraints.

As a natural extension of this Kahl and Hartley's approach, we can apply the bisection algorithm to (3) with another combination of p and q. For example, if we choose $p = 1$ and $q = \infty$, then (3) is rewritten as

$$\left|\begin{array}{ll} \text{minimize} & \gamma \\ \text{subject to} & |\mathbf{p}_i^1 \mathrm{U}_j - x_{ij}\mathbf{p}_i^3 \mathrm{U}_j| + |\mathbf{p}_i^2 \mathrm{U}_j - y_{ij}\mathbf{p}_i^3 \mathrm{U}_j| \leq \gamma \mathbf{p}_i^3 \mathrm{U}_j, \quad \left\{\begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N. \end{array}\right. \end{array}\right. \tag{6}$$

Note that the inequality

$$\left|\mathbf{p}_i^1 \mathrm{U}_j - x_{ij}\mathbf{p}_i^3 \mathrm{U}_j\right| + \left|\mathbf{p}_i^2 \mathrm{U}_j - y_{ij}\mathbf{p}_i^3 \mathrm{U}_j\right| \leq \gamma \mathbf{p}_i^3 \mathrm{U}_j$$

is equivalent to a set of inequalities

$$\left.\begin{array}{r}(\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j) + (\mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j) \le \gamma \mathbf{p}_i^3 U_j \\ (\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j) - (\mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j) \le \gamma \mathbf{p}_i^3 U_j \\ -(\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j) + (\mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j) \le \gamma \mathbf{p}_i^3 U_j \\ -(\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j) - (\mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j) \le \gamma \mathbf{p}_i^3 U_j. \end{array}\right\} \tag{7}$$

In other words, the constraints of (6) can be thought of as linear constraints once the value of γ is fixed. Hence, Algorithm 1 can solve (6) by checking the feasibility of (7) for all i, j, instead of (5). This can be done by solving an LP problem with (7) for all i, j as constraints. Similarly, we can solve (3) with $p = \infty$ and $q = \infty$ using the bisection algorithm.

Approximation approach: The reason why (3) cannot be solved directly as an SOCP or LP problem is that the product of two variables appears in the right-hand-sides of the constraints when (3) is rewritten as

$$\left|\begin{array}{ll}\text{minimize} & \|\gamma\|_q \\ \text{subject to} & \|\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j, \mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j\|_p = \gamma_{ij}\mathbf{p}_i^3 U_j, \quad \left\{\begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N. \end{array}\right.\end{array}\right. \tag{8}$$

Introducing new variables δ_{ij} and letting

$$\delta_{ij} = \gamma_{ij}\mathbf{p}_i^3 U_j, \quad i = 1, \ldots, M; j = 1, \ldots, N,$$

we have

$$\left|\begin{array}{ll}\text{minimize} & \|\boldsymbol{\delta}\|_q \\ \text{subject to} & \|\mathbf{p}_i^1 U_j - x_{ij}\mathbf{p}_i^3 U_j, \mathbf{p}_i^2 U_j - y_{ij}\mathbf{p}_i^3 U_j\|_p = \delta_{ij}, \quad \left\{\begin{array}{l} i = 1, \ldots, M \\ j = 1, \ldots N, \end{array}\right.\end{array}\right. \tag{9}$$

where $\boldsymbol{\delta} = (\delta_{11}, \ldots, \delta_{MN})^{\mathsf{T}}$. It is easy to see that (9) reduces to an SOCP problem if $p = 2$ and $q = \infty$, and is an LP problem if $p = 1, \infty$ and $q = \infty$. Furthermore, (9) can serve as a good approximation for (8) (and hence (3)).

Proposition 1 *If the optimal value of (8) vanishes, then any optimal solution of (8), together with $\boldsymbol{\delta} = \mathbf{O}$, solves (9) to optimality. Conversely, if the optimal value of (9) vanishes, then any optimal solution of (9), together with $\boldsymbol{\gamma} = \mathbf{O}$, solves (8) to optimality.* □

Table 2: CPU time (in seconds).

	$p \setminus q$	∞	1	2
bisection	∞	0.066	0.117	0.729
approximation	∞	0.003	0.003	0.075
	1	0.009	0.004	0.080

Table 3: 3D residual error (in L_2 norm).

	$p \setminus q$	∞	1	2
bisection	∞	0.017	0.018	0.016
approximation	∞	0.036	0.036	0.035
	1	0.014	0.014	0.013

Let us discuss the case where $q = \infty$ a little more closely. Let $(\overline{\gamma}_{ij}, \overline{\mathbf{p}_i^3\mathrm{U}_j})$ and $(\widehat{\delta}_{ij}, \widehat{\mathbf{p}_i^3\mathrm{U}_j})$ denote the values of $(\gamma_{ij}, \mathbf{p}_i^3\mathrm{U}_j)$ for optimal solutions of (8) and (9), respectively, and define

$$\widehat{\gamma} = \widehat{\delta}/\widehat{\mathbf{p}_k^3\mathrm{U}_\ell}, \quad \overline{\delta} = \overline{\gamma}\,\overline{\mathbf{p}_r^3\mathrm{U}_s},$$

where $\overline{\gamma} = \overline{\gamma}_{k\ell} \in \max_{i,j}\{\overline{\gamma}_{ij}\}$ and $\widehat{\delta} = \widehat{\delta}_{rs} \in \max_{i,j}\{\widehat{\delta}_{ij}\}$.

Proposition 2 *When $q = \infty$, the following relations hold:*

$$\overline{\gamma} \leq \widehat{\gamma} \leq L\overline{\gamma}, \quad \widehat{\delta} \leq \overline{\delta} \leq L\widehat{\delta},$$

where $L = \max_{i,j}\{\overline{\mathbf{p}_i^3\mathrm{U}_j}\}/\min_{i,j}\{\widehat{\mathbf{p}_i^3\mathrm{U}_j}\}$. □

4. Numerical results

Lastly, we report some numerical results obtained with MATLAB (version 7.9, R2009b) [5].

The problem used as a benchmark is triangulation. The number of cameras was fixed at $N = 25$, and each camera matrix was of the form:

$$\mathbf{P}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi_i & -\sin\phi_i & 0 \\ 0 & \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_i & 0 & -\sin\theta_i & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_i & 0 & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where z_i and ϕ_i are uniform random numbers on the intervals $[10.0, 100.0]$ and $[0.0, \frac{\pi}{6}]$, respectively, and

$$\theta_i = \frac{2\pi}{n(i-1)}.$$

Each component of the object point V was also generated uniformly at random in the interval $[-5.0, 5.0]$. Associated SOCP and LP problems were solved using SeDuMi [7] and GLPK[1], respectively. Table 2 summarizes the average CPU time in seconds taken to solve ten instances for each p, q. Table 3 shows the average residual error between actual and calculated values of V in L_2 norm. These results suggest that our approach is superior to Karl and Hartley's in both efficiency and accuracy.

References

[1] GLPK(GNU Linear Programming Kit) *http://www.gnu.org/software/glpk/.*

[2] Hartley, R.I. and Sturm, P. , Triangulation In *COMPUTER ANALYSIS OF IMAGES AND PATTERNS*, pages 190-197, 1995.

[3] Kahl, F. , Multiple view geometry and the L_∞-norm. In *Int. Conf. Computer Vision*,pages 1002-1009, Beijing, China, 2005.

[4] Kahl, F. and Hartley, R.I. , Multiple View Geometry Under the L_∞-norm. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1603-1617, September 2008.

[5] MATLAB *www.mathworks.com/products/matlab/.*

[6] Olsson, C. and Kahl, F. , Generalized Convexity in Multiple View Geometry. In *JOURNAL OF MATHEMATICAL IMAGING AND VISION*, pages 35-51, September 2010.

[7] SeDuMi *http://sedumi.ie.lehigh.edu/.*

[8] Seo, Y. and Hartley, R.I. , Sequential L_∞ Norm Minimization for Triangulation. In *Computer Vision - ACCV 2007*, vol. 4844, pages 322-331, 2007.