Mathematics-in-Industry Case Studies Journal, Volume 1, pp. 81-98 (2009)

Self Calibration of a Pinhole Camera

Paul Armand [*] Arthemy Kiselev [†] Odile Marcotte [‡] Dominique Orban [§]
Vincent Zalzal [¶]

Abstract. We present a procedure for self calibration of a pinhole camera subject to radial distortion. Radial distortion parameters are estimated using a nonlinear least-squares approach based on the knowledge that the observed image originates from a set of equidistant control points lying on straight lines. The pinhole perspective effect is corrected by minimizing an error in the least-squares sense subject to nonlinear constraints. We illustrate the two-stage self-calibration procedure with numerical results based on state-of-the-art computing tools in nonlinear optimization.

Keywords. Camera Self Calibration, Least Squares, Nonlinear Optimization, Pinhole Perspective, Radial Distortion

1 Introduction

In this paper we address a problem arising in camera calibration whose features are noticeably different from the usual ones. We shall describe our problem precisely before discussing the features that set it apart. The goal of camera calibration is to discover the relationship between points on an object observed by a camera—the *world points*, or *original points*—and the images of those points in a reference frame tied to the camera. The world points are assumed to lie in a plane called the *world plane*. Their coordinates are unknown, but the set of points is partitioned into a number of intersecting straight lines.

The world points are observed by a pinhole camera, and their images lie in the *focal plane* of the camera. Note that the focal plane is not, in general, parallel to the world plane; we thus have to take perspective into account. We assume that the distortion of the lens is radial, and because no measurement is perfectly accurate, we also assume the presence of noise. Hence the function that

[*]Département de Mathématiques, Université de Limoges, Limoges, France, armand@unilim.fr

[†]Institute of Mathematics, Utrecht University, Netherlands

[‡]Centre de recherches mathématiques and Université du Québec à Montréal, Montréal (Québec) Canada, odile.marcotte@gerad.ca

[§]Département de mathématiques et génie industriel, École Polytechnique de Montréal, and GERAD, Montréal (Québec) Canada, dominique.orban@gerad.ca

[¶]Matrox Electronic Systems, Dorval (Québec) Canada, vzalzal@matrox.com

maps a world point to its image in the focal plane depends upon three factors: the perspective, the distortion and the noise. In the following, we assume that these three factors act upon a world point in turn. First, a perspective projection maps a world point into the focal plane. Then, distortion and noise are applied to the result to obtain the image of the world point. We do not assume that the center of distortion is known.

The camera calibration problem is to determine the world points given their images, i.e., in a sense, to reconstruct the observed object. This problem was submitted to us in the context of the First Montréal Industrial Problem Solving Workshop [16]. In practice, the number of lines will be at most 30, and there will be between 8 and 25 points per line. The effect of the perspective will generally be much more important than the distortion. In fact, we shall see that the position of the world plane cannot be determined exactly, but all the possible world planes are parallel and all the possible reconstructed objects are homothetic. For the applications considered, this is a satisfactory outcome. Figure 1 gives an illustration of a set of observed points along with the reconstructed set of original points.

The problem we have just stated is different from those considered in the literature for three reasons. First, the original points are not known. Second, even the relative positions of lines are unknown, whereas most calibration techniques use structured calibration patterns (see below). Third, in most cases, the total set of points is not even "dense" in the world plane, i.e., there will be many points in some regions of that plane and relatively few points in other regions.

Ever since the seminal article [10], many authors have used the so-called "plumbline calibration" [15], in which the object to be reconstructed consists of vertical parallel lines (which may become horizontal after a 90 degree rotation). Lenz and Tsai [12], Barista *et al.* [9], and others have used calibration patterns (or grids) consisting of squares or disks that are equally spaced in the world plane. The authors of [13] use a calibration grid, [11] a set of random points and [14] a dense correspondence between the calibration plane and the world plane obtained through a structured light algorithm. In the vast literature on camera calibration, it seems that there are few articles (if any) considering a calibration based on lines that do not form a grid.

1.1 Notation and Assumptions

The number of straight lines will be denoted by N and the number of points in the i-th line by n_i, $i = 1, \ldots, N$. Those points will sometimes be referred to as *control points*. Let $d(u_1, u_2)$ be the Euclidean distance between two points u_1 and u_2. We assume that on line i, the points are equally spaced. Formally, the points on line i are labeled u_j^i, $j = 1, \ldots, n_i$ and we assume that $d(u_{j+1}^i, u_1^i) = jd(u_2^i, u_1^i)$ for $j = 1, \ldots, n_i - 1$. This implies, of course, that $d(u_{j+1}^i, u_j^i)$ is constant for all $j = 1, \ldots, n_i - 1$.

If we only consider the perspective projection, u_j^i is mapped onto $\bar{u}_j^i$ in the focal plane. Next, distortion and noise are applied to $\bar{u}_j^i$. The (unknown) center of distortion is denoted $\bar{u}_0$. Once the distortion and noise are taken into account, the images are denoted $\hat{u}_j^i$. Note also that the indices

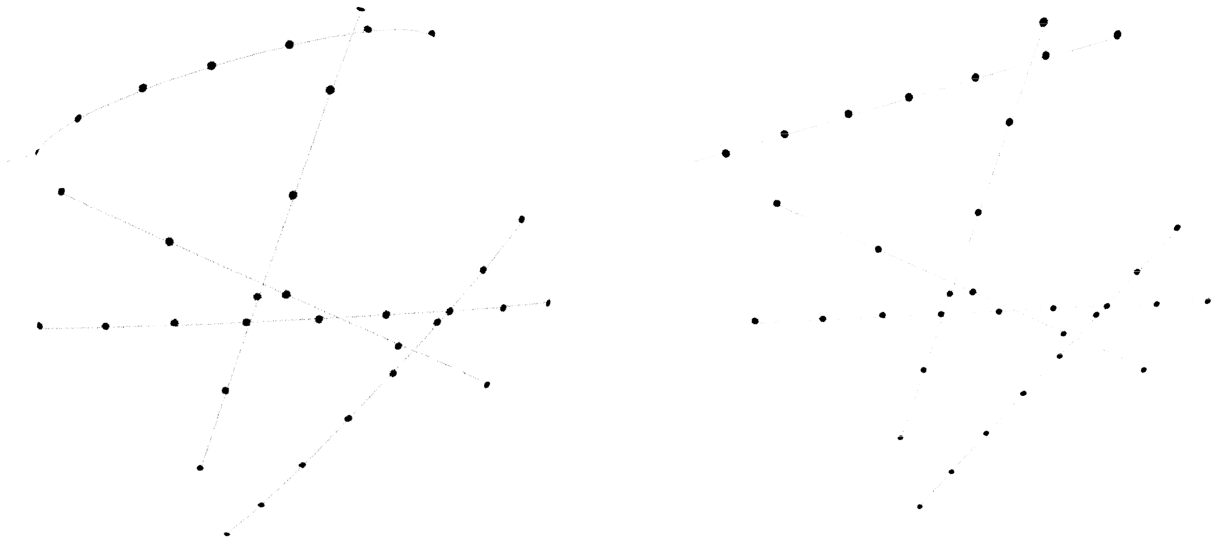

Figure 1: *On the left, the observed data is seen in perspective and with a certain amount of distortion, which may include random perturbations modeled as noise. On the right, the original data originates from a set of straight lines. On each straight line, the original points are equally spaced, but this spacing may not be the same from one line to the next. A typical situation has at most* 30 *lines and between* 8 *and* 25 *points on each.*

are "preserved", that is, we know that $\hat{u}^2_5$ (for instance) is the image of the fifth point on the second line.

The coordinates of a world point in world space will be denoted by (x, y, z). Its pinhole projection lies in a plane and is denoted $(\bar{x}, \bar{y})$. The noisy distortion of the latter has coordinates $(\widehat{x}, \widehat{y})$. Unless stated otherwise, all vectors are considered to be column vectors. We assume, without loss of generality, that z^i_1, the z component of the first point on each straight line, is nonzero.

2 Radial Distortion Correction

A radial distortion can be conveniently expressed in polar coordinates (r, θ) as

$$g(r, \theta) = (r(1 + \kappa r^2), \theta), \tag{1}$$

where $\kappa \in \mathbb{R}$ is an unknown parameter and the coordinates (r, θ) are measured from some unknown origin—the center of distortion, whose Cartesian coordinates will be denoted by $(\bar{x}_0, \bar{y}_0)$. Here, we model radial distortion up to the second degree, but more sophisticated models are possible, such as, for instance,

$$g(r,\theta) = (r(1+\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \cdots), \theta).$$

Only even powers of r appear in the model. In many applications, (1) is sufficient, and we will use it to generate our synthetic data.

Changing to Cartesian coordinates, we have

$$\bar{x} = \bar{x}_0 + r\cos\theta, \qquad \text{and} \qquad \bar{y} = \bar{y}_0 + r\sin\theta.$$

Thus, the radial distortion may be written as

$$g(\bar{x},\bar{y}) = \begin{bmatrix} \bar{x}_0 \\ \bar{y}_0 \end{bmatrix} + \left(1 + \kappa \left\| \begin{bmatrix} \bar{x} - \bar{x}_0 \\ \bar{y} - \bar{y}_0 \end{bmatrix} \right\|^2 \right) \begin{bmatrix} \bar{x} - \bar{x}_0 \\ \bar{y} - \bar{y}_0. \end{bmatrix}. \tag{2}$$

In the mathematical analysis that follows, we are assuming that the points are equally spaced, because this assumption is an integral part of the original problem submitted to us. The same analysis, however, can be used to solve the case where the points are not equally spaced. Note also that there is a close connection between our analysis and the so-called *anharmonic ratio*, first introduced by Pappus and used by Pascal and Desargues in their study of conic sections, cf. [17]. Assuming that the N sets of points $(\bar{x}^i_j, \bar{y}^i_j)$, $i = 1, \ldots, N$, $j = 1, \ldots, n_i$, represent the images of N original straight lines on which the points are equally spaced, their spacing must follow a precise pattern. To see this, let us focus on a single straight line to simplify the exposition and denote the equally-spaced points on this original straight line by

$$x_{j+1} = x_1 + j\Delta_x, \quad y_{j+1} = y_1 + j\Delta_y, \quad z_{j+1} = z_1 + j\Delta_z, \quad j = 0, 1, \ldots, n_i - 1,$$

where $\Delta = (\Delta_x, \Delta_y, \Delta_z)$ is the direction of the straight line. Under perspective projection, the images of those points are given by

$$\begin{bmatrix} \bar{x}_{j+1} \\ \bar{y}_{j+1} \end{bmatrix} = \frac{1}{z_{j+1}} \begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} = \frac{1}{z_{j+1}} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + j\frac{1}{z_{j+1}} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} = \frac{z_1}{z_{j+1}} \begin{bmatrix} \bar{x}_1 \\ \bar{y}_1 \end{bmatrix} + j\frac{z_1}{z_{j+1}} \begin{bmatrix} \Delta_x/z_1 \\ \Delta_y/z_1 \end{bmatrix}. \tag{3}$$

We introduce $\bar{\Delta}_{x_j}$ and $\bar{\Delta}_{y_j}$, defined by the relations:

$$\bar{x}_{j+1} = \bar{x}_1 + j\bar{\Delta}_{x_j}, \qquad \text{and} \qquad \bar{y}_{j+1} = \bar{y}_1 + j\bar{\Delta}_{y_j}, \qquad j = 1, \ldots, n_i - 1, \tag{4}$$

for some values $\bar{\Delta}_{x_j}$ and $\bar{\Delta}_{y_j}$ which are no longer constant but depend on j.

From (3) and (4), the x coordinate is obtained by solving

$$\left(\bar{x}_1 + j\frac{\Delta_x}{z_1}\right)\frac{z_1}{z_{j+1}} = \bar{x}_1 + j\bar{\Delta}_{x_j}, \quad j = 1, 2, \ldots, n_i - 1.$$

Therefore,

$$\bar{\Delta}_{x_j} = \frac{1}{z_{j+1}}\left(\frac{\bar{x}_1}{j}(z_1 - z_{j+1}) + \Delta_x\right), \quad j = 1, 2, \ldots, n_i - 1,$$

with a similar expression for $\bar{\Delta}_{y_j}$. We may now compute the ratio

$$\begin{aligned}\frac{\bar{\Delta}_{x_{j+1}}}{\bar{\Delta}_{x_j}} &= \left(\frac{z_{j+1}}{z_{j+2}}\right)\frac{j\bar{x}_1(z_1 - z_{j+2}) + j(j+1)\Delta_x}{(j+1)\bar{x}_1(z_1 - z_{j+1}) + j(j+1)\Delta_x} \\ &= \left(\frac{z_1 + j\Delta_z}{z_1 + (j+1)\Delta_z}\right)\frac{j\bar{x}_1(j+1)(-\Delta_z) + j(j+1)\Delta_x}{(j+1)\bar{x}_1 j(-\Delta_z) + j(j+1)\Delta_x} \\ &= \frac{z_1 + j\Delta_z}{z_1 + (j+1)\Delta_z}.\end{aligned}$$

This last expression is simply z_{j+1}/z_{j+2}, but since those quantities are unknown, it is useful to look at it as a ratio in which z_1 and Δ_z are constants for a given straight line. From our assumption that $z_1 \neq 0$ we can simplify the above ratio through dividing both the numerator and the denominator by z_1, and obtain the general form

$$\frac{\bar{\Delta}_{x_{j+1}}}{\bar{\Delta}_{x_j}} = \frac{1 + j\alpha}{1 + (j+1)\alpha}, \quad j = 1, 2, \ldots, n_i - 1,$$

for some unknown α that is constant for each straight line. The corresponding equation in the y component is

$$\frac{\bar{\Delta}_{y_{j+1}}}{\bar{\Delta}_{y_j}} = \frac{1 + j\alpha}{1 + (j+1)\alpha}, \quad j = 1, 2, \ldots, n_i - 1,$$

with *the same* α.

The unknowns κ, $(\bar{x}_0, \bar{y}_0)$, α_i (one for each line) and the coordinates of the control points seen in perspective $(\bar{x}^i_j, \bar{y}^i_j)$ are identified by solving the nonlinear least-squares problem with nonlinear constraints

$$\begin{array}{lll} \underset{\bar{x},\bar{y},\bar{x}_0,\bar{y}_0,\kappa,\alpha}{\text{minimize}} & \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{n_i} \|g(\bar{x}^i_j, \bar{y}^i_j) - (\hat{x}^i_j, \hat{y}^i_j)\|^2 & \\ \text{subject to} & \|\bar{v}^i_{j+1}\|\bar{v}^i_j = \|\bar{v}^i_j\|\bar{v}^i_{j+1}, & i = 1, \ldots, N, \quad j = 1, \ldots, n_i - 1, \\ & (1 + (j+1)\alpha_i)\bar{\Delta}^i_{j+1} = (1 + j\alpha_i)\bar{\Delta}^i_j, & i = 1, \ldots, N, \quad j = 1, \ldots, n_i - 1, \end{array} \tag{5}$$

where the variables α_i set the spacing pattern of the points on the straight lines as they are seen in perspective, and

$$\bar{v}^i_j = \begin{bmatrix} \bar{x}^i_{j+1} - \bar{x}^i_j \\ \bar{y}^i_{j+1} - \bar{y}^i_j \end{bmatrix}, \quad \text{and} \quad \bar{\Delta}^i_j = \begin{bmatrix} \bar{\Delta}^i_{x_j} \\ \bar{\Delta}^i_{y_j} \end{bmatrix}, \quad i = 1, \ldots, N, \ j = 1, \ldots, n_i - 1.$$

The first set of constraints of (5) imposes that the control points be aligned by requiring that the two consecutive normalized vectors from one control point to its successor be identical. The second set of constraints imposes the precise spacing pattern. Note that, using (4), the vectors $\bar{\Delta}^i_j$ and $\bar{v}^i_j$ are related by

$$j\bar{\Delta}^i_j = \sum_{k=1}^{j} \bar{v}^i_k, \quad \text{and} \quad \bar{v}^i_{j+1} = (j+1)\bar{\Delta}^i_{j+1} - j\bar{\Delta}^i_j.$$

It is therefore possible to eliminate either $\bar{\Delta}_j^i$ or $\bar{v}_j^i$ from the model altogether. Alternatively, either of the two relations above can be added to the model so as to formulate it in terms of both families of vectors. This is what we elected to do and we let the preprocessor of the modeling language simplify the model as much as possible.

3 Pinhole Perspective Correction

Given a point (x, y, z) in world space, its projection in perspective onto the focal plane is given by

$$\bar{x} = \frac{f}{z}\,x, \quad \text{and} \quad \bar{y} = \frac{f}{z}\,y, \tag{6}$$

where without loss of generality, it is assumed that all points in world space have $z > 0$ and $f > 0$ is the focal length of the camera. A simple two-dimensional illustration of pinhole perspective is given in Figure 2. In what follows, we will assume that entities are computed relative to $f = 1$. Given N sets of colinear points in the focal plane, the world points can be recovered by solving the following least-squares problem with nonlinear constraints

$$\begin{array}{lll} \underset{x,y,z,r,q,a,b,c}{\text{minimize}} & \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{n_i-2}(r_j^i)^2 + \sum_{i=1}^{N}\sum_{j=1}^{n_i}(q_j^i)^2 & \\ \text{subject to} & v_{j+1}^i = v_j^i + r_j^i, & i = 1,\dots,N,\ j = 1,\dots,n_i-2 \\ & a x_j^i + b y_j^i + c z_j^i = 1 + q_j^i, & i = 1,\dots,N,\ j = 1,\dots,n_i \\ & x_j^i = \bar{x}_j^i z_j^i,\ y_j^i = \bar{y}_j^i z_j^i, & i = 1,\dots,N,\ j = 1,\dots,n_i \\ & z_1^1 = 1, & \end{array} \tag{7}$$

where $v_j^i = u_{j+1}^i - u_j^i$ for all i and j. Note that with this notation, the first set of constraints can be rewritten

$$u_{j+2}^i - 2u_{j+1}^i + u_j^i = r_j^i, \quad i = 1,\dots,N,\ j = 1,\dots,n_i-2.$$

It is the latter formulation that we use in the implementation of (7).

In (7), it is assumed that points in the world space are coplanar, but the plane equation, defined by a, b, and c, is unknown. Because the data may be noisy, we may not be able to determine precisely a plane (up to parallelism) from which it originates. Therefore, we have introduced error coefficients, r_j^i and q_j^i, which are minimized in the least-squares sense. The error term r_j^i measures the misalignment of the reconstructed points while the error term q_j^i measures the lack of co-planarity of the straight lines. Note however that (7) is not a least-squares problem in the usual sense of the term since it has quadratic constraints.

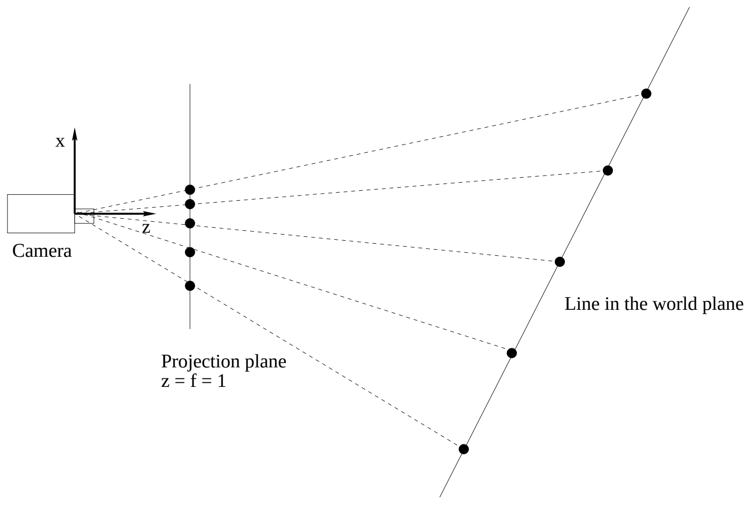

Figure 2: *A schematic two-dimensional view of pinhole perspective in the reference frame of the camera.*

4 Numerical Experiments

Since our approach to solving the camera-calibration problem involves nonlinear programming problems, including general nonlinear constraints, we naturally use nonlinear programming tools. Tools of choice for the modeler include the AMPL modeling language [2] and the NEOS Server for Optimization [3] available at http://neos.mcs.anl.gov. AMPL is a language in which users can easily represent linear and nonlinear optimization problems, including problems with integer or binary variables and problems involving sets or semi-definite matrices. As we illustrate in this section, representing problems such as (5) or (7) is easy and involves minimal syntax. The reader can compare the problem written in mathematical form and the model used to implement it, and realize how strikingly similar they are. A problem modeled in AMPL takes the form of two text files. The first, called the *model file*, describes the mathematical structure of the problem. The second, called the *data file*, instantiates a problem by assigning values to all the parameters declared in the model file. For the application considered here, these parameters are N, n_i, $\hat{x}^i_j$, and $\hat{y}^i_j$ for all i and j, and the initial guess. Several instances of the same problem can be obtained by using several data files—for example, this allows us to examine the influence of the initial guess or to test whether it is sufficient to consider three points per line only.

Nonlinear optimization algorithms typically require first and possibly second derivatives of all the functions involved—objective and constraints—in order to make progress. In the past, users had to hard-code those derivatives before passing their model to a solver. Fortunately, AMPL is able to supply those derivatives by means of a computational technique known as *automatic differentiation* (AD). AD should not be confused with symbolic differentiation. In most cases, it is far too expensive to compute analytic derivatives for all objectives and constraints. Rather, for given values of the variables, AD supplies the *values* of the first and second derivatives, to an accuracy of the order of machine precision, at a cost that is a small multiple of the cost of evaluating the function being differentiated [5].

The NEOS Server for Optimization is a portal allowing users to submit models of optimization problems, e.g., written in the AMPL modeling language, to one of the solvers provided. A number of solvers are available for a few categories of optimization problems—unconstrained problems, bound-constrained problems, linear problems, quadratic problems, general nonlinear problems, etc. The NEOS Server is able to serve a certain number of (near-)simultaneous submissions by means of grid computing. Globally, workstations are registered with the NEOS Server and have a few solvers installed on them. Whenever idle, those workstations become available for NEOS jobs. If a user submits a problem and requests a solver available on an idle workstation, the Server forwards the problem to that workstation, waits for the result, and forwards the latter—including the solver output—back to the user. This scheme is particularly convenient when testing solvers before purchasing one and alleviates the burden of installing a solver on a local machine.

The two problems presented in the previous section fall under the category of general nonlinear problems. They both have an important feature, which is that their only constraints are equality constraints. However, those constraints are nonlinear. In this respect, several choices are available in terms of algorithmic frameworks. Among them, we may choose the augmented Lagrangian method, the sequential quadratic programming (SQP) method, or the exterior penalty method. We refer the reader to [6] and to references therein for more information on each type of method. Our choice is to use the SQP method since it emerges as the current standard among methods for numerical optimization. This technique is briefly described next.

4.1 Sequential Quadratic Programming

The intent of this section is not to give a complete description of the SQP method but rather to outline its salient features. For a more complete discussion and pointers to fundamental references, the reader is referred to [6].

The basic nonlinear problem that optimization algorithms are able to solve effectively is the quadratic program, i.e., a problem whose objective is quadratic and whose constraints are all linear. The general form of a quadratic problem is thus

$$\begin{array}{ll} \underset{d\in\mathbb{R}^n}{\text{minimize}} & g^T d + \tfrac{1}{2} d^T H d \\ \text{subject to} & Ad = b, \end{array} \tag{8}$$

where $g \in \mathbb{R}^n$ is a fixed vector, $H \in \mathbb{R}^{n\times n}$ is a symmetric matrix, $A \in \mathbb{R}^{m\times n}$ is the constraint matrix and $b \in \mathbb{R}^m$ is fixed.

Given a nonlinear optimization problem with equality constraints

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0, \end{array} \tag{9}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice-continuously differentiable functions, the SQP method solves a sequence of quadratic programs to solve the general problem. This procedure is akin to Newton's method. The quadratic program solved at iterate (x, y) is defined by

$$g = \nabla f(x), \quad H = \nabla_{xx} L(x, y), \quad A = J(x), \quad \text{and} \quad b = -c(x),$$

where $L(x, y) = f(x) + y^T c(x)$ is the Lagrangian of (9), $y \in \mathbb{R}^m$ is the vector of *Lagrange multipliers*, and $J(x) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix of c at x, i.e., the matrix whose i-th row is $\nabla c_i(x)^T$.

A solution d to the quadratic program encountered at the current iteration gives a search direction along which an improved iterate x_+ will be sought. The quadratic program allows us to update our current estimate of the optimal variables, and an additional mechanism is usually required to update our current estimate of the optimal Lagrange multipliers. The choice of the next iterate is carried out by means of a merit function or a filter, i.e., a mechanism able to measure the worth of a given candidate. We will not go into further detail here as the discussion would divert us from our initial goal. Again, the reader will find more information on such concepts in any good nonlinear optimization textbook, such as the one given above.

On the NEOS Server, several implementations of the SQP method are available. We elected to use LOQO [1], a solver initially written for quadratic programs and later extended to general nonlinear optimization. In reality, LOQO is also able to treat inequality constraints by implementing a so-called *interior-point method*. However, in the absence of inequality constraints, LOQO reduces to the SQP method.

4.2 Synthetic Data Generation

Two sets of synthetic data $(\widehat{x}_j^i, \widehat{y}_j^i)$ were used in our tests. The first set is a prescribed number of randomly-generated straight lines with a prescribed number of equally-spaced points on them. The lines are generated so that they lie in the same plane. The data are then projected perspectively and perturbed by radial distortion using a random distortion center about the center of mass of the data points and a randomly chosen distortion coefficient lying in a given interval. A parameter allows the user to add random noise to the resulting data. In what follows we always express the noise as a percentage.

The second data set is mostly used to illustrate the perspective correction and is chosen as a set of lines depicting a simple sailboat. Visualizing the sailboat before perspective and the reconstructed sailboat illustrates well the error of reconstruction in the presence of noise.

4.3 Distortion Correction

Problem (5) is expressed as the AMPL model shown in Listing 1. The data for this model, giving precise values to N, n_i, and the $(\widehat{x}_j^i, \widehat{y}_j^i)$, is loaded from the synthetic data file.

Listing 1: *AMPL model file for the elimination of distortion*

```
# Distortion correction model by nonlinearly-constrained least squares
model;

param N, integer > 0;                  # Number of straight lines
param n {1..N}, integer >= 3;          # Number of points per line
param xhat {i in 1..N, j in 1..n[i]}; # x-coord of the points after distorsion
param yhat {i in 1..N, j in 1..n[i]}; # y-coord of the points after distorsion

var x {i in 1..N, j in 1..n[i]};       # x-coord of the reconstructed points
var y {i in 1..N, j in 1..n[i]};       # y-coord of the reconstructed points
var alpha {i in 1..N};         # Spacing on the straight line seen in perspective

var x0;                                # x-coord of the center of distortion
var y0;                                # y-coord of the center of distortion
var K;                                 # Coefficient of distortion

var r2 {i in 1..N, j in 1..n[i]} = (x[i,j]-x0)^2 + (y[i,j]-y0)^2; # = r^2
var gx {i in 1..N, j in 1..n[i]} = (1+K*r2[i,j]) * (x[i,j]-x0) + x0;
var gy {i in 1..N, j in 1..n[i]} = (1+K*r2[i,j]) * (y[i,j]-y0) + y0;
var norm {i in 1..N, j in 2..n[i]} =
    sqrt( (x[i,j]-x[i,j-1])^2 + (y[i,j]-y[i,j-1])^2 );

var delx {i in 1..N, j in 1..n[i]-1};
var dely {i in 1..N, j in 1..n[i]-1};

minimize l2_error:
    sum {i in 1..N, j in 1..n[i]} ((gx[i,j]-xhat[i,j])^2+(gy[i,j]-yhat[i,j])^2);

subject to x_aligned {i in 1..N, j in 2..n[i]-1}:
    norm[i,j] * (x[i,j+1] - x[i,j]) = norm[i,j+1] * (x[i,j] - x[i,j-1]);

subject to y_aligned {i in 1..N, j in 2..n[i]-1}:
    norm[i,j] * (y[i,j+1] - y[i,j]) = norm[i,j+1] * (y[i,j] - y[i,j-1]);

subject to x_spacing {i in 1..N, j in 1..n[i]-2}:
    delx[i,j+1] * (1+(j+1)*alpha[i]) = delx[i,j] * (1+j*alpha[i]);

subject to y_spacing {i in 1..N, j in 1..n[i]-2}:
    dely[i,j+1] * (1+(j+1)*alpha[i]) = dely[i,j] * (1+j*alpha[i]);

subject to def_delx {i in 1..N, j in 1..n[i]-1}:
    j * delx[i,j] = sum {k in 1..j} (x[i,k+1]-x[i,k]);

subject to def_dely {i in 1..N, j in 1..n[i]-1}:
    j * dely[i,j] = sum {k in 1..j} (y[i,k+1]-y[i,k]);
```

The initial guess was chosen as $(\bar{x}, \bar{y}) = (\widehat{x}, \widehat{y})$, $\kappa = 0.01$, and $(\bar{x}_0, \bar{y}_0)$ is chosen as the barycenter of the set of data points. The AMPL data file used to initialize the data is given in Listing 2. The actual data file includes the values of $(\widehat{x}^i_j, \widehat{y}^i_j)$ for all $j = 1, \ldots, n_i$ and all $i = 1, \ldots, N$ but they were omitted from the listing for the sake of brevity.

Listing 2: *AMPL data file for the elimination of distortion*

```
# Data file used to initialize the distortion correction model
data;

param N := 20;

param xhat:   # Initialization omitted for conciseness (20 lines)
param yhat:   # Initialization omitted for conciseness (20 lines)

let {i in 1..N} n[i] := 9;
let {i in 1..N, j in 1..n[i]} x[i,j] := xhat[i,j];
let {i in 1..N, j in 1..n[i]} y[i,j] := yhat[i,j];
let K1 := 1.0e-2;
let x0 := 1/(sum {i in 1..N} n[i]) * sum {i in 1..N, j in 1..n[i]} xhat[i,j];
let y0 := 1/(sum {i in 1..N} n[i]) * sum {i in 1..N, j in 1..n[i]} yhat[i,j];
```

The results of applying LOQO to problem (5) with and without noise in the data are given in Table 2. For these tests, the data was generated with the center of distortion and distortion coefficient given as in Table 1. There are 20 lines in total and 9 points per line, i.e., $N = 20$ and $n_i = 9$ for all $i = 1, \ldots, N$. The noise follows a uniform distribution. In Table 2, the timings are for information only since, as explained in Section 4, the nature of the NEOS Server implies that different problems may have been solved on different machines, with perhaps quite different architectures. The optimality residual is as reported by the solver and the relative error is computed by recovering the final result returned by the solver and comparing it to the initial values used to generate the data. The value reported here is the largest of the relative errors in the x and y components. The residual and reconstructed data are illustrated in Fig. 3.

Table 1: *Original center of distortion and distortion coefficient for various amounts of random noise used to generate the synthetic data.*

Noise	0.0	0.5	1.0	1.5
x_0	0.016743	0.016734	0.016726	0.016717
y_0	0.013640	0.013636	0.013631	0.013627
κ	2.301546	2.302048	2.302550	2.303052

In each case, LOQO stopped the optimization as soon as a scaled optimality residual fell below the threshold of 10^{-5}. It is apparent from Table 2 that the relative error in the noisy case remains quite large.

Table 2: *Sample results of* LOQO *applied to the distortion reduction problem with various amounts of random noise. See Table 1 for the exact data. In this table, the relative error is not given as a percentage.*

Noise	0.0	0.5	1.0	1.5
Solve Time (s)	1.16	0.28	0.75	0.69
Iterations	39	32	46	42
Optimality Residual	5.7e-06	2.0e-05	5.6e-06	2.7e-05
Relative error	1.2e-04	5.5e-03	9.9e-03	1.5e-02
x_0	0.016772	0.017566	0.018237	0.018915
y_0	0.013580	0.011579	0.014405	0.014976
κ	2.300804	2.360069	2.246736	2.221315

4.4 Perspective Correction

Problem (7) is expressed as the AMPL model shown in Listing 3.

Listing 3: *AMPL model file for the correction of perspective*

```
# This model implements perspective correction
model;

param N > 0;                          # Number of straight lines
param n {1..N};                       # Number of points on each straight line
param xbar {i in 1..N, j in 1..n[i]}; # x-coord of points after perspective
param ybar {i in 1..N, j in 1..n[i]}; # y-coord of points after perspective

var x {i in 1..N, j in 1..n[i]};      # x-coord of points before perspective
var y {i in 1..N, j in 1..n[i]};      # y-coord of points before perspective
var z {i in 1..N, j in 1..n[i]};      # z-coord of points before perspective
var a;
var b;
var c;                                # World plane  a*x + b*y + c*z = 1
var resx {i in 1..N, j in 1..n[i]-2};
var resy {i in 1..N, j in 1..n[i]-2};
var resz {i in 1..N, j in 1..n[i]-2};
var resc {i in 1..N, j in 1..n[i]};

minimize l2_error:
    sum{i in 1..N, j in 1..n[i]-2} (resx[i,j]^2 + resy[i,j]^2 + resz[i,j]^2) +
    sum{i in 1..N, j in 1..n[i]} resc[i,j]^2;

subject to perspective_x {i in 1..N, j in 1..n[i]}:
    z[i,j] * xbar[i,j] = x[i,j];

subject to perspective_y {i in 1..N, j in 1..n[i]}:
    z[i,j] * ybar[i,j] = y[i,j];

subject to aligned_x {i in 1..N, j in 1..n[i]-2}:
    x[i,j+2] - 2*x[i,j+1] + x[i,j] = resx[i,j];

subject to aligned_y {i in 1..N, j in 1..n[i]-2}:
    y[i,j+2] - 2*y[i,j+1] +  y[i,j] = resy[i,j];

subject to aligned_z {i in 1..N, j in 1..n[i]-2}:
    z[i,j+2] - 2*z[i,j+1] + z[i,j] = resz[i,j];

subject to lines_coplanar {i in 1..N, j in 1..n[i]}:
    a * x[i,j] + b * y[i,j] + c * z[i,j] - 1 = resc[i,j];

subject to fix_z11 : z[1,1] = 1;
```

The data file used to initialize the model in this case is similar to that for the distortion correction and we omit it here. Since the focal plane and the world plane are not closely related, no particular initializations were performed—all initial guesses were set to zero. An illustration of sample results on our first set of synthetic data is given in Figure 4. When a certain amount of random noise perturbs the original data, the distortion is not corrected exactly. As a result, attempting to subsequently correct the perspective effect results in an image which does not lie in the world plane, but another, secant plane. In our tests, the larger the magnitude of the noise, the larger the angle between the two planes. This effect can perhaps be better visualized on a second data set, the sailboat set, illustrated in Figure 5.

Table 3: *Sample results of* LOQO *applied to the pinhole perspective correction problem with various amounts of random noise.*

Noise	0.0	0.5	1.0	1.5
Solve Time (s)	0.94	0.28	0.27	0.40
Iterations	92	34	35	37
Optimality Residual	6.9e-10	1.8e-07	3.1e-07	5.4e-07
Relative error	2.6e-04	5.0e-03	2.0e-02	3.0e-02

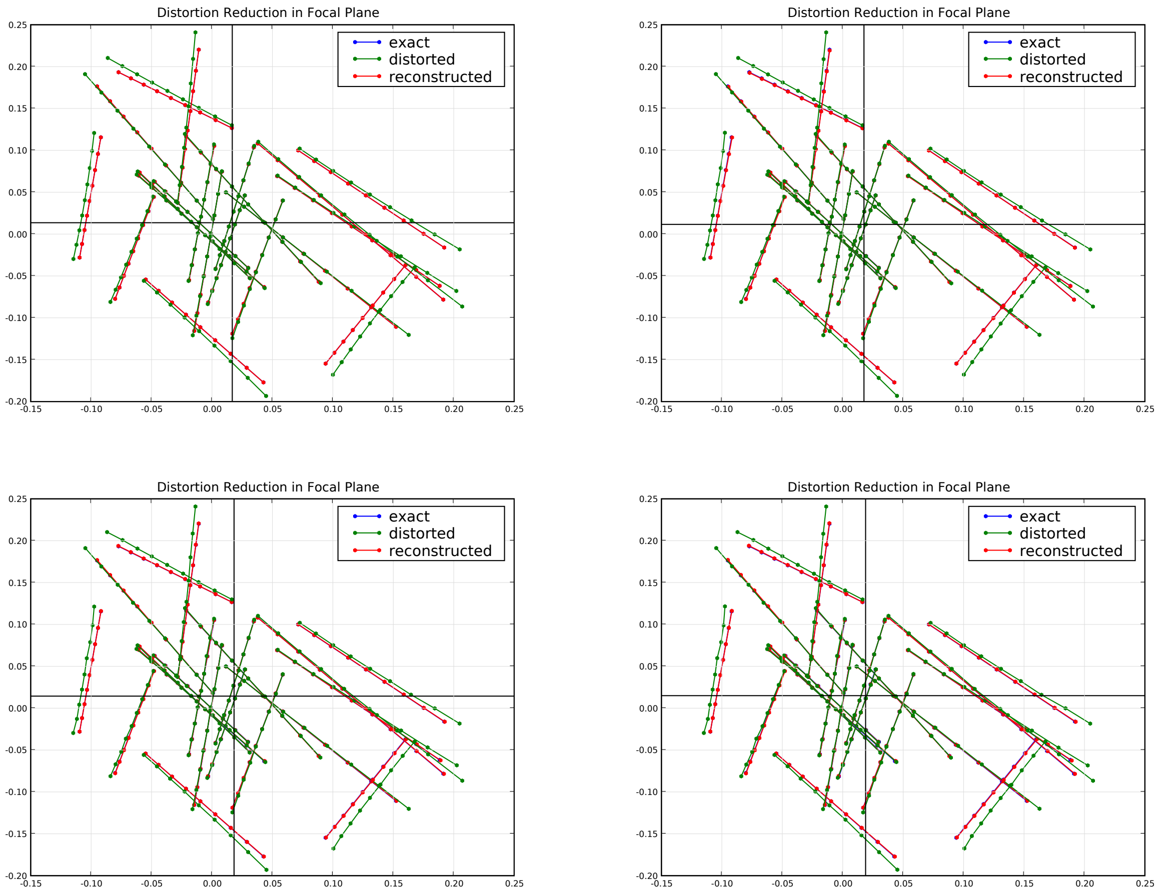

Figure 3: *Radial distortion correction. In the top-left figure, the data was not perturbed with random noise. The correction is essentially exact and the reconstructed points are superposed with the original data and are visually indistinguishable. From left to right and top to bottom, the amount of random noise added to the distorted data is, respectively,* 0.5, 1.0, *and* 1.5. *The correction is no longer exact. The crosshair indicates the computed center of distortion in each case.*

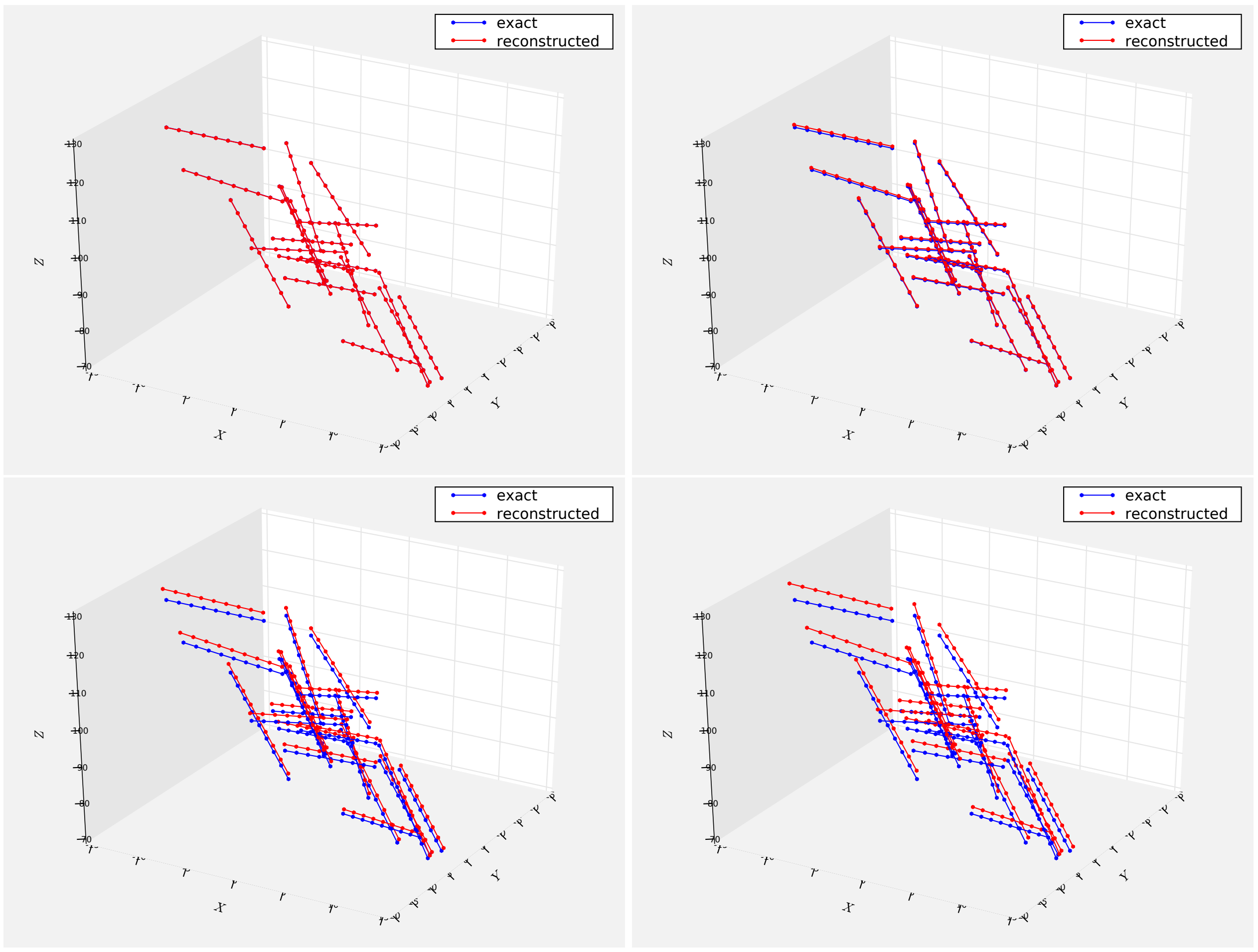

Figure 4: *Pinhole perspective correction. The above pictures represent the initial (red) and reconstructed (blue) data in world space. In the top-left figure, the data was not perturbed with random noise. The correction is essentially exact. From left to right and top to bottom, the amount of random noise added to the distorted data is, respectively,* 0.5, 1.0 *and* 1.5.

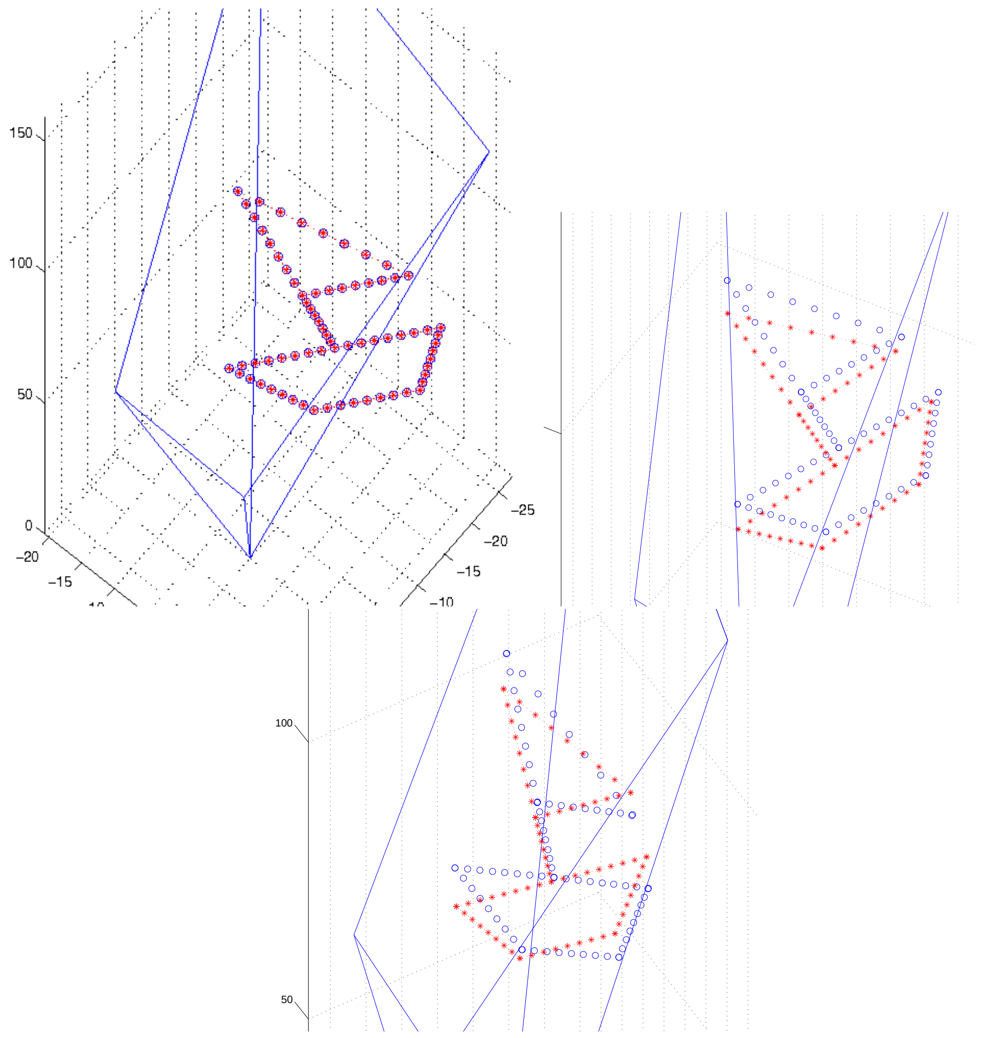

Figure 5: *Pinhole perspective correction. The above pictures represent the initial (red) and reconstructed (blue) data in world space. On the upper left, the data was not perturbed with random noise. The correction is again essentially exact as the reconstructed points are superposed with the original data. On the upper right, a medium amount of uniformly distributed noise was applied to the original data. The reconstructed sailboat is no longer exact. Directly above, a higher amount of random noise was applied.*

5 Conclusion

In this paper, we have presented a two-stage procedure to calibrate a pinhole camera based on the observation of several sets of colinear points. The observed points are subject to radial distortion and their reading can be perturbed by random noise. The procedure relies on solving nonlinear optimization problems with least-squares objectives. For low amounts of noise, the reconstruction is nearly perfect. As the noise level increases, small inaccuracies in the correction of distortion translate to noticeable errors after pinhole projection correction. The reconstruction then becomes less accurate and produces points lying in a plane other than the world plane. An application of this process was the original motivation for the present case study and was supplied to us in the context of automatically asserting conformity of manufactured components.

In future research, we intend to test this procedure on real data, examine techniques of noise removal to be applied prior to distortion correction and investigate the influence of noise distribution on the reconstructed image. We would also like to weaken the assumption on the precise form of the distortion and the perspective.

Acknowledgments

Arthemy Kiselev wishes to thank the Centre de recherches mathématiques for its warm hospitality.

References

[1] Vanderbei, R. J. and Shanno, D. F., An interior point algorithm for nonconvex nonlinear programming *Computational Optimization and Applications* **13** (1999), 231-252. 89

[2] Fourer, R., Gay, D. M., and Kernighan, B. W., *AMPL: A Modeling Language for Mathematical Programming*, 2nd Edition, Duxbury Press 2002. 87

[3] Czyzyk, J., Mesnier, M., and Moré, J. J., The NEOS Server, *IEEE Journal on Computational Science and Engineering* **5** (1998), 68-75. 87

[4] Faugeras, O., *Three Dimensional Computer Vision*, MIT Press 1993.

[5] Griewank, A., Evaluating Derivatives–Principles and Techniques of Automatic Differentiation, SIAM, *Frontiers in Applied Mathematics*, 2000. 88

[6] Nocedal, J. and Wright, S. J., Numerical Optimization, *Springer Series in Operations Research*, Springer-Verlag 1999. 88

[7] Zhang, Z., A Flexible New Technique for Camera Calibration, *Technical Report MSR-TR-98-71*, Microsoft Research 1998.

[8] Heikkilä, J. and Silvén, O., A Four-Step Camera Calibration Procedure with Implicit Image Correction, *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1106-1112 (1997).

[9] Batista, J., Araújo, H., and de Almeida, A. T., Iterative multi-step explicit camera calibration, *Proceedings of the Sixth International Conference on Computer Vision*, 709-714 (1998), 82

[10] Brown, D. C., Close-range camera calibration, *Photogrammetric Engineering and Remote Sensing* **37** (no.8, 1971), 855-866. 82

[11] Hartley, R. I., Self-calibration from multiple views with a rotating camera, *Proceedings of the Third European Conference on Computer Vision*, LNCS **800** (1994), 471-478. 82

[12] Lenz, R. K. and Tsai, R. Y., Techniques for calibration of the scale factor and image center for high accuracy 3-D machine vision metrology, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10** (no.5, 1988), 713-720. 82

[13] Sturm, P. and Maybank, S. J., On plane-based camera calibration: a general algorithm, singularities, applications, *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)* **1** (1999), 1432-1437. 82

[14] Tardif, J-P. and Sturm, P., Calibration of cameras with radially symmetric distortion, *Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS'2005, in parallel with ICCV 2005)* (2005), 44-51. 82

[15] Tardif, J.-P., Sturm, P., and Roy, S., Self-calibration of a general radially symmetric distortion model, *European Conference on Computer Vision*, LNCS **3954** (2006), 186-199. 82

[16] First Montréal Industrial Problem Solving Workshop, CRM, Montréal, Québec, Canada, August, 2007. 82

[17] Courant, R. and Robbins, H., *What Is Mathematics?: An Elementary Approach to Ideas and Methods, 2nd. ed.*, Oxford University Press 1996. 84