J Math Imaging Vis (2009) 35: 36–50
DOI 10.1007/s10851-009-0153-2

An Algebraic Approach to Lens Distortion by Line Rectification

Luis Alvarez · Luis Gómez · J. Rafael Sendra

Published online: 2 May 2009

Abstract A very important property of the usual pinhole model for camera projection is that 3D lines in the scene are projected in 2D lines. Unfortunately, wide-angle lenses (specially low-cost lenses) may introduce a strong barrel distortion which makes the usual pinhole model fail. Lens distortion models try to correct such distortion. In this paper, we propose an algebraic approach to the estimation of the lens distortion parameters based on the rectification of lines in the image. Using the proposed method, the lens distortion parameters are obtained by minimizing a 4 total-degree polynomial in several variables. We perform numerical experiments using calibration patterns and real scenes to show the performance of the proposed method.

Keywords Radial lens distortion · Edge line rectification · Linear regression · Algebraic approach · Minimal residual variance

L. Alvarez
Departamento de Informática y Sistemas, Universidad de Las Palmas de Gran Canaria, Campus de Tafira, 35017 Las Palmas, Spain
e-mail: lalvarez@dis.ulpgc.es

L. Gómez (✉)
Departamento de Ingeniería Electrónica y Automática, Universidad de Las Palmas de Gran Canaria, Campus de Tafira, 35017 Las Palmas, Spain
e-mail: lgomez@diea.ulpgc.es

J.R. Sendra
Departamento de Matemáticas, Universidad de Alcalá, 28871 Alcalá de Henares, Madrid, Spain
e-mail: Rafael.Sendra@uah.es

1 Introduction

Typically, wide angle lenses tend to suffer from barrel distortion and tele lenses from pincushion distortion. Both effects tend to be stronger at the extreme ends of zoom lenses, especially on low-cost compact cameras, web-cam, fish-eye lens, etc.

Lens distortion correction is an important issue in camera calibration where the pinhole model is used (see for instance [1, 2] or [3]). The basic standard model for barrel and pincushion distortion compensation (see for instance [4, 5] or [6]) is a radial distortion model given by the following expression:

$$\begin{pmatrix} \hat{x} - x_c \\ \hat{y} - y_c \end{pmatrix} = L(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix}, \tag{1}$$

where (x, y) are the original point coordinates (distorted), $(\hat{x}, \hat{y})$ are the corrected (undistorted) point coordinates, (x_c, y_c) is the center of the camera distortion model, usually the center of the image (in fact, in this paper we will always take as distortion center, the center of the image), $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ and $L(r)$ is the function which defines the shape of the distortion model. Usually, $L(r)$ is approximated by a Taylor expansion, that is

$$L(r) = k_0 + k_1 r + k_2 r^2 + k_3 r^3 + \cdots ,$$

where the set $\mathbf{k} = (k_0, k_1, \ldots, k_{N_k})^T$ are the distortion parameters. The complexity of the model is given by the number of terms of the Taylor expansion we use to approximate $L(r)$.

In this paper, we use the general approach to determine $L(r)$ by imposing the requirement that the projection of 3D lines in the image has to be 2D straight lines. This approach has been successfully used in [7–11], in these papers, the authors minimize the objective error functions which are expressed in terms of line equations or distance functions. In this paper we propose a new fast technique to obtain the distortion parameter model using a new lens distortion measure error. The main advantage of our formulation is that it yields to a general 4-degree polynomial in the distortion parameters k_i, that can be minimized using powerful techniques of computer algebra. Using the algebraic approach, the global minimum of the distortion measure error function, for the 3 parameter model case (i.e. $\mathbf{k} = (k_0, k_i, k_j)^T$ with $0 < i, j \leq N_k$) can be directly found (bounded) in one step. As a particular application of the proposed method, we also show how to use this technique in an iterative way to estimate lens distortion model with higher number of parameters.

The lens distortion is included in the camera calibration model in the following way: Given a camera defined by a rotation matrix R, a focus $\mathbf{c} = (c_x, c_y, c_z)^T$ and a 3×3 intrinsic matrix parameter A, the projection (x, y) of a 3D point $\mathbf{X} = (X, Y, Z)$ in the camera is given by the following expression

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = sA(R, -Rc) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

where $(\hat{x}, \hat{y})$ is defined in (1) and s is the usual projective factor value. In the case of $L(r) \equiv 1$, the camera model is lens distortion free and the above expression becomes the usual "pinhole" projective model where the projection of a 3D point $\mathbf{X}$ in the camera is given by the interception of the line $\overline{\mathbf{cX}}$ with the retinal plane. We observe that lens distortion correction is performed in pixel image coordinates. In order to the lens distortion model be a radial function in pixels coordinates, we need to assume that the camera CCD sensor has square pixel. This square pixel size assumption is satisfied, in practice, for most modern digital cameras. On the other hand, this square size assumption could be removed if we use normalized coordinates instead of pixel coordinates, but to normalize the point coordinates we need to know the camera intrinsic parameters. Since we assume that camera intrinsic parameters are unknowns we have formulated the lens distortion correction in terms of pixel coordinates.

To calibrate accurately a camera, usually a linear technique is applied to get an initial estimation of camera parameters and then a bundle adjustment is used to improve the accuracy of the parameter estimation. The bundle adjustment is based on a nonlinear minimization where the mean square error between the observed and predicted image points is minimized. Usually the distortion model is included in the bundle adjustment parameter minimization (see for instance [12] for more details). In that sense, the method we propose can be used to get a fast initial estimation of the distortion model in the bundle adjustment procedure.

The paper is organized as follows: In Sect. 2 we introduce the measure of the distortion error we propose in this paper. In Sect. 3 we present the algebraic analysis of the proposed measure of the distortion error. In Sect. 4, we analyze the numerical aspects of the implementation of our algorithm for estimating the distortion parameters. Section 5 is devoted to the performed numerical experiments. Finally, in Sect. 6 we present some conclusions.

2 Measure of the Distortion Error

Let $\{(x_{l,i}, y_{l,i})\}$ with $l = 1, \ldots, N$ and $i = 1, \ldots, N_l$ be the projection of N sets of $3D$ aligned points in the $2D$ image, let $\{(\hat{x}_{l,i}, \hat{y}_{l,i})\}$ be the corrected (undistorted) points using the distortion model (1) and $\mathbf{k} = (k_0, k_1, \ldots, k_{N_k})^T$ the distortion parameters. For each line l, and for each point i, we note by $\overline{\hat{y}_{l,i}}$ and $\overline{\hat{x}_{l,i}}$ the average of the respective variables taken over i. We also consider the covariance matrix given by

$$\hat{S}^l(\mathbf{k}) = \begin{pmatrix} \hat{S}^l_{xx} & \hat{S}^l_{xy} \\ \hat{S}^l_{xy} & \hat{S}^l_{yy} \end{pmatrix}$$
$$\equiv \frac{1}{N_l} \begin{pmatrix} \sum_{i=1}^{N_l} (\hat{x}_{l,i} - \overline{\hat{x}_{l,i}})^2 & \sum_{i=1}^{N_l} (\hat{y}_{l,i} - \overline{\hat{y}_{l,i}})(\hat{x}_{l,i} - \overline{\hat{x}_{l,i}}) \\ \sum_{i=1}^{N_l} (\hat{y}_{l,i} - \overline{\hat{y}_{l,i}})(\hat{x}_{l,i} - \overline{\hat{x}_{l,i}}) & \sum_{i=1}^{N_l} (\hat{y}_{l,i} - \overline{\hat{y}_{l,i}})^2 \end{pmatrix}.$$

The lens distortion measure we propose in this paper is based in the following lemma:

Lemma 1 *Let*

$$\hat{E}(\mathbf{k}) = \frac{1}{N} \sum_{l=1}^{N} \left(\hat{S}^l_{xx} \hat{S}^l_{yy} - \left(\hat{S}^l_{xy} \right)^2 \right) \tag{2}$$

then $\hat{E}(\mathbf{k}) \geq \mathbf{0}$ *and* $\hat{E}(\mathbf{k}) = \mathbf{0}$ *if and only if for each line* l, *the points* $\{(\hat{x}_{l,i}, \hat{y}_{l,i})\}_{i=1,\ldots,N_l}$ *are aligned.*

Proof Using the Cauchy-Schwarz inequality (see for instance [13]) we obtain that for any line l

$$\left(\hat{S}^l_{xy} \right)^2 \leq \hat{S}^l_{xx} \hat{S}^l_{yy}$$

and the equality holds only when the variables are proportional, that is, when there exists a_l, b_l such that for any $i \in \{1, N_l\}$

$$a_l(\hat{x}_{l,i} - \overline{\hat{x}_{l,i}}) + b_l(\hat{y}_{l,i} - \overline{\hat{y}_{l,i}}) = 0$$

so in particular the points $\{(\hat{x}_{l,i}, \hat{y}_{l,i})\}_{i=1,\dots,N_l}$ are aligned, which concludes the proof of the lemma. $\square$

According with this lemma, we propose as lens distortion measure $\hat{E}(\mathbf{k})$ and the distortion parameters $\mathbf{k}$ will have to minimize the functional $\hat{E}(\mathbf{k})$.

Next, we will show that $\hat{E}(\mathbf{k})$ is a 4-degree polynomial in the coefficient of $\mathbf{k}$. Indeed, using the distortion model (1) we obtain:

$$\hat{S}^l_{xx} = \frac{1}{N_l}\sum_{i=1}^{N_l}\left(\sum_{j=0}^{N_k} k_j\left((r_{l,i})^j x_{l,i} - \overline{(r_{l,i})^j x_{l,i}}\right)\right)^2 = \mathbf{k}^T A^l \mathbf{k},$$

$$\hat{S}^l_{yy} = \frac{1}{N_l}\sum_{i=1}^{N_l}\left(\sum_{j=0}^{N_k} k_j\left((r_{l,i})^j y_{l,i} - \overline{(r_{l,i})^j y_{l,i}}\right)\right)^2 = \mathbf{k}^T B^l \mathbf{k},$$

$$\hat{S}_{xy} = \frac{1}{N_l}\sum_{i=1}^{N_l}\left(\sum_{j=0}^{N_k} k_j\left((r_{l,i})^j x_{l,i} - \overline{(r_{l,i})^j x_{l,i}}\right)\right) \times \left(\sum_{j=0}^{N_k} k_j\left((r_{l,i})^j y_{l,i} - \overline{(r_{l,i})^j y_{l,i}}\right)\right) = \mathbf{k}^T C^l \mathbf{k},$$

where $r_{l,i}=\sqrt{(x_{l,i}-x_c)^2+(y_{l,i}-y_c)^2}$, $\overline{(r_{l,i})^j x_{l,i}}$ and $\overline{(r_{l,i})^j y_{l,i}}$ are the average of the respective variables taken over i, and A^l, B^l, C^l are $(N_k+1)\times(N_k+1)$ matrix given by

$$\begin{aligned} A^l_{m,n} &= \frac{1}{N_l}\sum_{i=1}^{N_l}((r_{l,i})^m x_{l,i} - \overline{(r_{l,i})^m x_{l,i}}) \times ((r_{l,i})^n x_{l,i} - \overline{(r_{l,i})^n x_{l,i}}),\\ B^l_{m,n} &= \frac{1}{N_l}\sum_{i=1}^{N_l}((r_{l,i})^m y_{l,i} - \overline{(r_{l,i})^m y_{l,i}}) \times ((r_{l,i})^n y_{l,i} - \overline{(r_{l,i})^n y_{l,i}}),\\ C^l_{m,n} &= \frac{1}{N_l}\sum_{i=1}^{N_l}((r_{l,i})^m x_{l,i} - \overline{(r_{l,i})^m x_{l,i}}) \times ((r_{l,i})^n y_{l,i} - \overline{(r_{l,i})^n y_{l,i}}). \end{aligned} \tag{3}$$

Therefore, the distortion error measure $\hat{E}(\mathbf{k})$ can be expressed as

$$\hat{E}(\mathbf{k}) = \frac{1}{N}\sum_{l=1}^{N} \mathbf{k}^T A^l \mathbf{k}\mathbf{k}^T B^l \mathbf{k} - \mathbf{k}^T C^l \mathbf{k}\mathbf{k}^T C^l \mathbf{k} \tag{4}$$

which is a 4-degree homogeneous polynomial in the variable $\mathbf{k}$.

Of course, the global minimum of $\hat{E}(\mathbf{k})$ corresponds to the trivial solution $\mathbf{k} \equiv (0,0,\dots,0)^T$. To avoid this problem, usually k_0 is set to one ($k_0 = 1$). As it is explained in Sect. 4, in this paper we use another approach: we fit k_0 using a zoom factor by minimizing the sum of the square distance between the distorted and undistorted points.

3 Algebraic Analysis of the Distortion Error Measure

In this section, we show how to approach the problem by means of computer algebra techniques. For simplicity in the exposition, we present the results for polynomials with real coefficients, but it must be said that they are valid over more general polynomials rings; for further details on this topic we refer the reader to [14] or [15].

As mentioned in Sect. 2, one needs to minimize the distortion error measure function $\hat{E}(\mathbf{k})$, which is a real polynomial in the variable $\mathbf{k}$. Minimizing a polynomial in several variables can be reduced to compute the solutions of an algebraic system of equations, namely the one generated by its gradient. In our case:

$$\mathcal{S} := \left\{\frac{\partial \hat{E}(\mathbf{k})}{\partial k_i} = 0\right\}_{i=1,\dots,N_k}.$$

As it will be explained in Sect. 4, parameter k_0 is estimated in a different way as a zoom factor, therefore the above algebraic system does not include the derivative of $\hat{E}(\mathbf{k})$ with respect to k_0.

When the polynomial is univariate, say k_p is the variable, one just has to approximate the real roots on the univariate polynomial

$$\frac{\partial \hat{E}(k_p)}{\partial k_p}.$$

However, when more than one variable appears, the problem is not so trivial. In order to approach this new situation, one can apply computer algebra techniques to prepare symbolically the algebraic system $\mathcal{S}$ before numerical methods are executed. The two-variable case can be treated by means of symbolic linear algebra techniques while the case of more than two variables requires, in general, abstract algebra techniques. In both cases, the underlining theory comes from algebraic geometry and commutative algebra. To be more precise, we first describe in detail how to approach the problem when two variable are considered, and afterward we give a brief description on how to proceed in the general case.

So, let us assume that we are working with two variables, say k_p, k_q. Observe that this is the case when working with two distortion parameters, and that the system $\mathcal{S}$ turns to be

$$\mathcal{S} := \left\{\frac{\partial \hat{E}(k_p,k_q)}{\partial k_p} = 0,\ \frac{\partial \hat{E}(k_p,k_q)}{\partial k_q} = 0\right\}.$$

In order to compute the solutions of $\mathcal{S}$ we apply the so called resultant-based method. Let us describe this method. For this purpose, let $G_1(k_p,k_q)$ and $G_2(k_p,k_q)$ be two bivariate polynomials with real coefficients. Choosing one variable, say k_q, as a main variable, we can write G_1 and G_2 as

$$G_1(k_p,k_q)=a_n(k_p)k_q^n+\cdots+a_1(k_p)k_q+a_0(k_p),$$

$$G_2(k_p,k_q)=b_m(k_p)k_q^m+\cdots+b_1(k_p)k_q+b_0(k_p),$$

where $a_i(k_p)$ and $b_i(k_p)$ are univariate polynomials with real coefficients, and $a_n(k_p)$, $b_m(k_p)$ are not identically zero, with $n>0$ and $m>0$. In this situation, the *resultant of* G_1 *and* G_2 *with respect to the variable* k_q (we denote it by $\mathrm{Res}_{k_q}(G_1,G_2)$) is defined as the determinant of the $(n+m)\times(n+m)$ Sylvester matrix

$$\left(\begin{array}{ccccccc}
a_n(k_p) & a_{n-1}(k_p) & \cdots & a_0(k_p) & 0 & \cdots & 0\\
0 & a_n(k_p) & a_{n-1}(k_p) & \cdots & a_0(k_p) & \cdots & 0\\
\vdots & & \ddots & & & \ddots & \vdots\\
0 & 0 & \cdots & a_n(k_p) & a_{n-1}(k_p) & \cdots & a_0(k_p)\\
b_m(k_p) & b_{m-1}(k_p) & \cdots & b_0(k_p) & 0 & \cdots & 0\\
0 & b_m(k_p) & b_{m-1}(k_p) & \cdots & b_0(k_p) & \cdots & 0\\
\vdots & & \ddots & & & \ddots & \vdots\\
0 & 0 & \cdots & b_m(k_p) & b_{m-1}(k_p) & \cdots & b_0(k_p)
\end{array}\right)\begin{array}{l}\left.\vphantom{\begin{array}{c}a\\a\\ \vdots\\a\end{array}}\right\}m\\ \left.\vphantom{\begin{array}{c}a\\a\\ \vdots\\a\end{array}}\right\}n\end{array}.$$

Observe that $\mathrm{Res}_{k_q}(G_1,G_2)$ is a real univariate polynomial in the variable k_p. Therefore, the variable k_q has been eliminated. For our purposes, the main applicable properties on resultants are the following.

Theorem 1 *Let* $G_1(k_p,k_q), G_2(k_p,k_q)$ *as above, and let* $G(k_p)=\mathrm{Res}_{k_q}(G_1,G_2)$. *Then, it holds that*

1. $G(k_p)$ *is identically zero if and only if* G_1 *and* G_2 *have a common non-constant factor.*
2. *If* $(\lambda,\mu)\in\mathbb{C}^2$ *is a common root* G_1 *and* G_2 *then* $G(\lambda)=0$.
3. *If* $G(\lambda)=0$ *then one of the following statements holds*
   - 3.1. $a_n(\lambda)=b_m(\lambda)=0$,
   - 3.2. $\exists\mu\in\mathbb{C}$ *such that* (λ,μ) *is a common root of* G_1 *and* G_2.

Proof See Theorem 4.3.3, p. 98, in [14]. □

The geometrical meaning of Theorem 1 is as follows (see Sect. 2.3 in [16] for further details). Let G_1, G_2 and G be as above. Then we can see G_1 and G_2 as curves in the k_pk_q-coordinate plane $\mathbb{C}^2$. In this situation, the roots of G are the k_p-coordinates of the intersection points of the two curves (see Fig. 1). Moreover, the real roots of G contain the k_p-coordinate of the real intersection points of the two curves.

In order to apply Theorem 1, first, note that in the construction of $\mathrm{Res}_{k_q}(G_1,G_2)$, we have required that $\deg_{k_q}(G_1)>0$ and $\deg_{k_q}(G_2)>0$. Let us see that this assumption is not a loss of generality for our purposes. Indeed, if $\deg_{k_q}(G_1)=0$ (similarly if $\deg_{k_q}(G_2)=0$), then G_1 only depends on k_p. Then, if $\deg_{k_q}(G_2)=0$, G_2 is also univariate and the real solutions of $\{G_1(k_p)=G_2(k_p)=0\}$ are the real roots of the greatest common divisor of both polynomials. On the other hand, if $\deg_{k_q}(G_2)>0$, for each real root α of the univariate polynomial $G_1(k_p)$, one has to determine the real roots of the univariate polynomial $G_2(\alpha,k_q)$. That is, if $\deg_{k_q}(G_1)=0$, $\deg_{k_q}(G_2)>0$, the real solutions of the

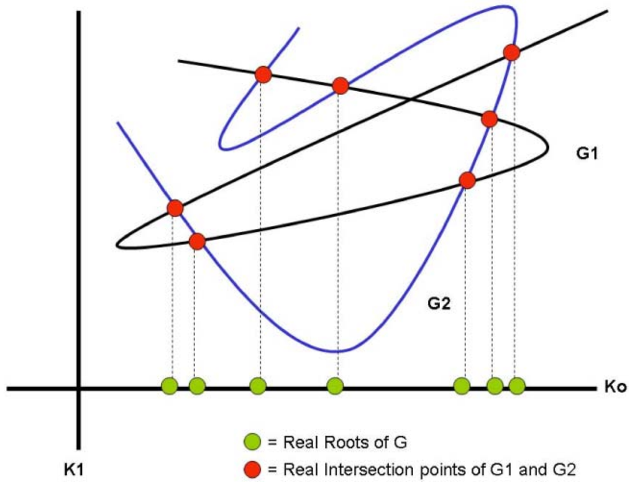


Fig. 1 Geometric interpretation of the resultant $G=\mathrm{Res}_{k_q}(G_1,G_2)$

system $\{G_1(k_p) = G_2(k_p, k_q) = 0\}$ are

$\{(\alpha, \beta_\alpha) \in \mathbb{R}^2 \mid G_1(\alpha) = 0, G_2(\alpha, \beta_\alpha) = 0\}$.

Moreover, if the following conditions are satisfied:

(i) the conditions on the degree are fulfilled (i.e. $\deg_{k_q}(G_1) > 0$ and $\deg_{k_q}(G_2) > 0$),
(ii) $\gcd(G_1, G_2) = 1$ (i.e. the greatest common divisor of both polynomials is 1),
(iii) and either $a_n(k_p)$ or $b_m(k_p)$ is a constant polynomial (note that a_n and b_m are, by definition, not identically zero),

then Theorem 1 implies that all the solutions (in the particular the real ones) of the system $\{G_1(k_p, k_q) = 0, G_2(k_p, k_q) = 0\}$ can be obtained from the roots of $G(k_p)$; this process is known as the lifting process.

We have already seen that hypothesis (i) (see above) can be assumed w.l.o.g. Let us see how to proceed in general with hypotheses (ii) and (iii).

- *Hypothesis* (ii). If $\gcd(G_1, G_2) = D \neq 1$, dividing G_1, G_2 by D one gets two new polynomials, say G_1^* and G_2^*, fulfilling the *gcd* condition, and the solutions of $\{G_1 = 0, G_2 = 0\}$ are the solutions of $D = 0$ union the finitely many solutions of $\{G_1^* = 0, G_2^* = 0\}$. Moreover, note that since in our case the polynomials come from empirical data the most expectable situation is that the polynomials are coprime, i.e. its gcd is 1.
- *Hypothesis* (iii). If none of the polynomials $a_n(k_p)$, $b_m(k_p)$ is constant, one can check whether taking k_p as a main variable the property holds. If for none of the variables k_p and k_q the requirement holds, then one can always apply a linear change of coordinates such that the new polynomials verify the property; note that applying the inverse of the linear change of coordinates to the solutions of the new system one gets the solutions of the initial one. In order to deterministically choose this linear change of coordinate, we proceed as follows: we express one of the polynomials, say G_1, as a sum of homogenous polynomials (recall that a bivariate polynomial $H(k_p, k_q)$ is homogeneous of degree r is $H(tk_p, tk_q) = t^r H(k_p, k_q)$ where t is a new variable):

$$G_1(k_p, k_q) = H_r(k_p, k_q) + \cdots + H_1(k_p, k_q) \\ + H_0(k_p, k_q),$$

where H_i is homogeneous of degree i. So, H_i collects all terms in G_1 of total degree i; or equivalently H_i is the i-degree part of the Taylor expansion of G_1 around $(0, 0)$. In this situation, if $(1, b) \in \mathbb{R}^2$ is such that $H_r(b, 1) \neq 0$ then

$$G_1(k_p + bk_q, k_q) = H_r(b, 1)k_q^r + \text{terms of lower degree},$$

and therefore the requirement is achieved.

The next proposition shows that, in our case, hypothesis (iii) holds except for some non realistic point distribution configuration.

Proposition 2 *If for some line l, the points $\{((r_{l,i})^p x_{l,i}, (r_{l,i})^p y_{l,i})\}_{i=1,\ldots,N_l}$ are not aligned, then hypothesis* (iii) *holds.*

Proof From (4), in the particular case of the distortion model, one has that

$$\frac{\partial \hat{E}(k_p, k_q)}{\partial k_q} = b_3(k_p)k_q^3 + b_2(k_p)k_q^2 + b_1(k_p)k_q + b_0(k_q)$$

where

$$b_3(k_p) = 4\frac{1}{N}\sum_{l=1}^{N}(A_{pp}^l B_{pp}^l - (C_{pp}^l)^2).$$

Therefore $b_3(k_p)$ is constant, and since $A_{pp}^l B_{pp}^l - (C_{pp}^l)^2 \geq 0$, then $b_3(k_p) = 0$ if and only if for every line l, the points $((r_{l,i})^p x_{l,i} - \overline{(r_{l,i})^p x_{l,i}}, (r_{l,i})^p y_{l,i} - \overline{(r_{l,i})^p y_{l,i}})$ lie on a line. In particular for each line l, there exist a_l, b_l such that, for each i,

$$a_l((r_{l,i})^p x_{l,i} - \overline{(r_{l,i})^p x_{l,i}}) + b_l((r_{l,i})^p y_{l,i} - \overline{(r_{l,i})^p y_{l,i}}) = 0$$

and therefore points $((r_{l,i})^p x_{l,i}, (r_{l,i})^p y_{l,i})$ lie on a line. So we conclude the statement of the proposition and hypothesis (iii) holds. □

Remark We observe that if for all lines l, the points $\{((r_{l,i})^p x_{l,i}, (r_{l,i})^p y_{l,i})\}_{i=1,\ldots,N_l}$ are aligned then the lens distortion of all lines is completely corrected using the transformation

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = r^p \begin{pmatrix} x \\ y \end{pmatrix} = \left((x - x_c)^2 + (y - y_c)^2\right)^{\frac{p}{2}} \begin{pmatrix} x \\ y \end{pmatrix}. \quad (5)$$

However this transformation does not fit model (1) except in the case $(x_c, y_c) = (0, 0)$ because in (5) the distance r is computed with respect to (x_c, y_c) but the points (x, y) and $(\hat{x}, \hat{y})$ are not translated with respect to (x_c, y_c) as in (1). The case $(x_c, y_c) = (0, 0)$ is an unrealistic configuration because, in practice, in real images, the distortion center is located near the pixel image center (far beyond (0, 0)). On the other hand, we observe that we can switch the roles of p and q in the analysis and if for one line l, the points $\{((r_{l,i})^p x_{l,i}, (r_{l,i})^p y_{l,i})\}_{i=1,\ldots,N_l}$ are aligned and $\{((r_{l,i})^q x_{l,i}, (r_{l,i})^q y_{l,i})\}_{i=1,\ldots,N_l}$ are also aligned (with $p \neq q$) then we easily conclude that the line has a trivial configuration, that is the line is composed of just 1 or 2 points or the points $\{(x_{l,i}, y_{l,i})\}_{i=1,\ldots,N_l}$ lie in a line passing by $(0, 0)$.

Summarizing, one can derive the following algorithm to compute the real solutions of

$$\mathcal{S} := \left\{ \frac{\partial \hat{E}(k_p, k_q)}{\partial k_p} = 0,\ \frac{\partial \hat{E}(k_p, k_q)}{\partial k_q} = 0 \right\},$$

where we assume w.l.o.g. that hypotheses (i), (ii), and (iii) hold. Note that, once these solutions are known, minimizing the distortion error measure function $\hat{E}(k_p, k_q)$, in the compact set of analysis, is trivial.

1. Determine $G_1 := \frac{\partial \hat{E}}{\partial k_p}$ and $G_2 := \frac{\partial \hat{E}}{\partial k_q}$.
2. Determine $G(k_p) := \mathrm{Res}_{k_q}(G_1, G_2)$ and approximate the real roots of $G(k_p)$. Let $\mathcal{R} = \{\alpha_1, \ldots, \alpha_s\}$ be the set of real roots of G.
3. For each $\alpha \in \mathcal{R}$ approximate the common real roots of the univariate polynomials $G_1(\alpha, k_q)$ and $G_2(\alpha, k_2)$. Let $\mathcal{R}_\alpha$ be the set of these real common roots.
4. The real solutions of $\mathcal{S}$ are $\{(\alpha, \beta_\alpha) \mid \alpha \in \mathcal{R} \text{ and } \beta_\alpha \in \mathcal{R}_\alpha\}$.

In the general case, i.e. when working with $s > 2$ variables, say $k_{p_1}, \ldots, k_{p_s}$, the problem cannot be approached so directly by means of resultants. Nevertheless, one can apply *Gröbner basis* techniques or multivariate-resultants (see [17] and [14] for further information). Of course, Gröbner basis techniques can also be applied to the case of two variables but, in that case, we find more suitable the resultant-based method. The basic idea of Gröbner basis, as a tool for solving algebraic systems, is to provide a new algebraic system of equations equivalent to $\mathcal{S}$ (i.e. with the same solutions) but much simpler, and such that it has a suitable structure ("triangular") to compute the solutions. Roughly speaking, Gröbner basis can be seen as a generalization of the Gaussian elimination when the equations are not linear. We leave, as future research work, the applications of the Gröbner basis method to the current problem.

4 The Algorithm

Using the technique presented above, we can estimate any pair k_p, k_q $(p, q \geq 1)$ of the distortion parameters. In fact we can update any previous distortion parameter estimation by optimizing any pair of distortion parameters. Indeed, given an estimation $\mathbf{k}$ of the distortion parameters, we can write

$$\begin{pmatrix} \hat{x}_{l,i} - x_c \\ \hat{y}_{l,i} - y_c \end{pmatrix} = \left(\sum_{j=0}^{N_k} k_j (r_i)^j + \epsilon_p (r_i)^p + \epsilon_q (r_i)^q \right) \begin{pmatrix} x_{l,i} - x_c \\ y_{l,i} - y_c \end{pmatrix},$$

where ϵ_p, ϵ_q are additive update of k_p, k_q. We can minimize the lens distortion measure $\hat{E}(\mathbf{k})$ with respect to ϵ_p, ϵ_q using the algebraic approach presented in the above section. Once ϵ_p, ϵ_q are estimated we can update k_p and k_q just by adding ϵ_p, ϵ_q.

The distortion parameters $\mathbf{k}$ are computed setting $k_0 = 1$. In order to yield undistorted points as close as possible to the distorted ones we estimate a zoom factor to minimize the sum of the square distance between the distorted and the corrected (undistorted) points. Let s be such zoom factor, then we have:

$$\begin{pmatrix} \hat{x} - x_c \\ \hat{y} - y_c \end{pmatrix} = sL(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix}$$

we minimize:

$$H(s) = \sum_{l=1}^{N} \sum_{i=1}^{N_l} \left(\hat{x}_{l,i} - x_{l,i}\right)^2 + \left(\hat{y}_{l,i} - y_{l,i}\right)^2$$

an straightforward computation leads to

$$H(s) = \sum_{l=1}^{N} \sum_{i=1}^{N_l} \left(s \sum_{j=0}^{N_k} k_j^n \left(r_{l,i}\right)^{j+1} - r_{l,i} \right)^2$$

the minimum of the above function is attained in

$$s_{\min} = \frac{\sum_{l=1}^{N} \sum_{i=1}^{N_l} \sum_{j=0}^{N_k} k_j^n (r_{l,i})^{j+2}}{\sum_{l=1}^{N} \sum_{i=1}^{N_l} (\sum_{j=0}^{N_k} k_j^n (r_{l,i})^{j+1})^2}$$

and finally we update the polynomial $L(r)$ (i.e. $\mathbf{k}$) by multiplying all the coefficients by $s_{\min}$ (that is $k_j^n = s k_j^n\ \forall j$).

An interesting advantage of this approach is that the resolution of the undistorted image is similar to the resolution of the original (distorted) image. This is a very useful property if we need to generate the undistorted image from the original distorted one.

Therefore the derived algorithm for performing the numerical experiments can be structured in the following steps:

1. We compute the edges of the image using an edge detection algorithm with subpixel precision.
2. We select some collections of edge points corresponding to different 3D straight segments, that will be used to fit the distortion parameters.
3. We initialize $\mathbf{k} = (1, 0, \ldots, 0)^T$.
4. We choose any pair $p, q \in \mathbb{Z}$ $(1 \leq p, q \leq N_k)$ and we optimize k_p, k_q using the proposed algebraic technique.
5. We update $\mathbf{k}$ using a zoom factor such that distorted and undistorted points are as close as possible.

In order to compute the image edges we can use any standard edge detector algorithm (see for instance [18, 19]).

Remark (Point coordinates normalization) It is well known that when we deal with algebraic methods (see for instance

[5]) it is usually better to normalize the point coordinates before computing the algebraic solution of the problem. Following this strategy, as a first step, we normalize the edge points (x_i, y_i) using the transformation

$$x'_{l,i} = \frac{(x_{l,i} - x_c)}{A}, \qquad y'_{l,i} = \frac{(y_{l,i} - y_c)}{A},$$

where A is given by

$$A = \sqrt{\frac{\sum_{l=1}^{N} \sum_{i=1}^{N_l} (x_{l,i} - x_c)^2 + (y_{l,i} - y_c)^2}{2(N_1 + N_2 + \cdots + N_N)}}$$

and we compute the distortion parameters k'_i for the normalized edge points $\{(x'_{l,i}, y'_{l,i})\}_{i=1}^{N}$. Finally, in order to recover the distortion parameters k_i for the original edge points we have just to take into account that following the above expressions and (1) we have that

$$k_j = \frac{k'_j}{(A)^j}.$$

4.1 Inversion of the Radial Distortion Model

For some applications we need to invert the radial distortion model. For instance, to build the undistorted version of the image it is usually better to use the inverted of the radial distortion model. So we look for a radial function $G(\hat{r})$ such that

$$\begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} = G(\hat{r}) \begin{pmatrix} \hat{x} - x_c \\ \hat{y} - y_c \end{pmatrix},$$

where

$$\hat{r} = \sqrt{(\hat{x} - x_c)^2 + (\hat{y} - y_c)^2}.$$

From the above expression we obtain that

$$r = G(\hat{r})\hat{r}.$$

On the other hand we have

$$\begin{pmatrix} \hat{x} - x_c \\ \hat{y} - y_c \end{pmatrix} = L(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix}$$

and therefore

$$\hat{r} = L\left(G(\hat{r})\hat{r}\right) G(\hat{r})\hat{r}.$$

So we conclude that $G(\hat{r})$ is a root of the polynomial

$$P(\lambda) = 1 - L(\lambda\hat{r})\lambda = 1 - \sum_{j=0}^{N_k} k_j \hat{r}^j \lambda^{j+1}.$$

In order to minimize the distance between the distorted and undistorted points, we choose, among all possible real roots of $P(\lambda)$, the one nearest to 1.

4.2 Implementation Details and Computational Complexity

To implement the minimization algorithm we have presented in the previous section we use standard C language. We observe that we need just an algorithm to compute the determinant of an sparse 6×6 polynomial matrix of degree bounded by 3 which provides always a polynomial of degree at most 9 and an algorithm to compute real polynomial roots, so the numerical implementation is quite simple and we do not need to use sophisticated symbolic tools which suffers of numerical accuracy problems when we deal with general floating point arithmetic.

In order to compute the polynomial determinant, one only needs to take into account that it is an addition of products of the matrix entries. More precisely, we have proceeded as follows: we have implemented the basic polynomial arithmetic, i.e. very simple functions for executing polynomial multiplication and addition/subtraction, as well as the usual recursive algorithm for matrix determinant estimation. We observe that the only difference with the scalar matrix case is that the matrix elements are polynomials so when we have to multiply (or add) matrix elements we use the predefined polynomial operations.

To compute the polynomial roots we use the standard Jenkins-Traub real polynomial root algorithm.

From a computational complexity point of view, we observe that the computational cost of the algorithm depends on the number of edge points used for the different lines (given by $M = \sum_{l=1}^{N} N_l$). In our algorithm implementation, these edge points are used only in the computation of the matrix A^l, B^l and C^l defined in (3). Once these matrix are computed, we built the coefficients of the polynomial (4). The structure of this polynomial is independent of the number of edge points, so we can conclude that the computational complexity of the algorithm is linear in the number of line edge points M. We also observe that, in the case we want to compute higher polynomial degree lens distortion model using in an iterative way the proposed technique by updating different polynomial coefficients in each iteration, the computational complexity of the model is $\mathcal{O}(M + I)$ where I is the number of iterations. We observe that in the usual standard iterative techniques (as gradient descent), the computational complexity is $\mathcal{O}(M \cdot I)$.

5 Numerical Experiments

In order to validate the accuracy of the solution provided by the proposed algebraic method, we will compare such solution with the one obtained by minimizing the sum of the squares of the distances from the undistorted points to the lines. This distortion measure have been used in [10] to estimate the distortion parameters **k** and can be written as

$$D(\mathbf{k}) = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{N_l}\sum_{i=1}^{N_l}\frac{(a_l\hat{x}_{l,i}+b_l\hat{y}_{l,i}+c_l)^2}{a_l^2+b_l^2} \tag{6}$$

where $(\hat{x}_{l,i}, \hat{y}_{l,i})$ are the undistorted points using the distortion model provided by **k**, and where $a_l\hat{x}+b_l\hat{y}+c_l$ is the line that minimizes $D(\mathbf{k})$ for a given choice of points $\{(\hat{x}_{l,i}, \hat{y}_{l,i})\}_{i=1,\dots,N_l}$.

We used an standard gradient descent method [20] to minimize $D(\mathbf{k})$ which has been applied to some standard unconstrained optimization test problems to check its efficiency. Main details of the gradient implementation are explained along this section.

The advantages of the proposed algebraic method are: it is a direct method, it does not require iterations or distortion parameters initialization and it can not be trapped in local minima for the two-variable case, as it has been previously shown mathematically. On the other hand, as it will be confirmed from the numerical experiments, the proposed method is much faster than the iterative one when we deal with a large number of line edge points.

In the numerical experiments we will assume that the distortion center (x_c, y_c) is the center of the image. In Fig. 2 we show the images used to illustrate the performance of the algebraic method. In the case of the calibration patterns we use, we have printed them and taken photos of the printed images with a wide-angle lens camera.

The advantage of these calibration patterns is that we can easily identify the rectangles presented in the image, and automatically select the edge segments and points we will use for the estimation of the distortion model parameters. In Fig. 3 the achieved results, for the planar lens distortion calibration pattern, are shown.

The achieved quantitative results for the first calibration pattern are presented in Table 1. We applied the method to the even set of distortion parameters (k_2 and k_4) which is coherent with the fact that the even distortion parameters are more relevant than the odd distortion parameters (see for instance [4]). The odd distortion parameters are fixed to zero ($k_1 = 0$ and $k_3 = 0$). The algebraic (distortion measure value $\hat{E}(\mathbf{k})$) and the numerical (distance function value $D(\mathbf{k})$) are presented for the 2-degree polynomial (k_0 and k_2) and for the 4-degree polynomial (k_0, k_2 and k_4) for both methods starting from the trivial setup ($\mathbf{k} = (1, 0, \dots, 0)^T$). So in the case of 2-degree polynomial we assume $\mathbf{k} = (k_0, 0, k_2)^T$ and in the case of 4-degree polynomial we assume $\mathbf{k} = (k_0, 0, k_2, 0, k_4)^T$. $\hat{E}(\mathbf{k})$ and $D(\mathbf{k})$ (initial value and solutions) are normalized to their setup values. The number of iterations, the number of function evaluations and the CPU time are also included. This CPU time (execution time) is for the programs running on an one-core 2.4 GHz (2 GB RAM) Intel PC machine. The gradient algorithm has been coded in standard C language using double precision. It is important to indicate that this CPU time does not include image preprocessing (loading the image and capturing the points to process) and postprocessing operations (normalization and inverse radial transformation to represent the undistorted image), hence, it is a real indication on the time needed for obtaining the distortion parameters.

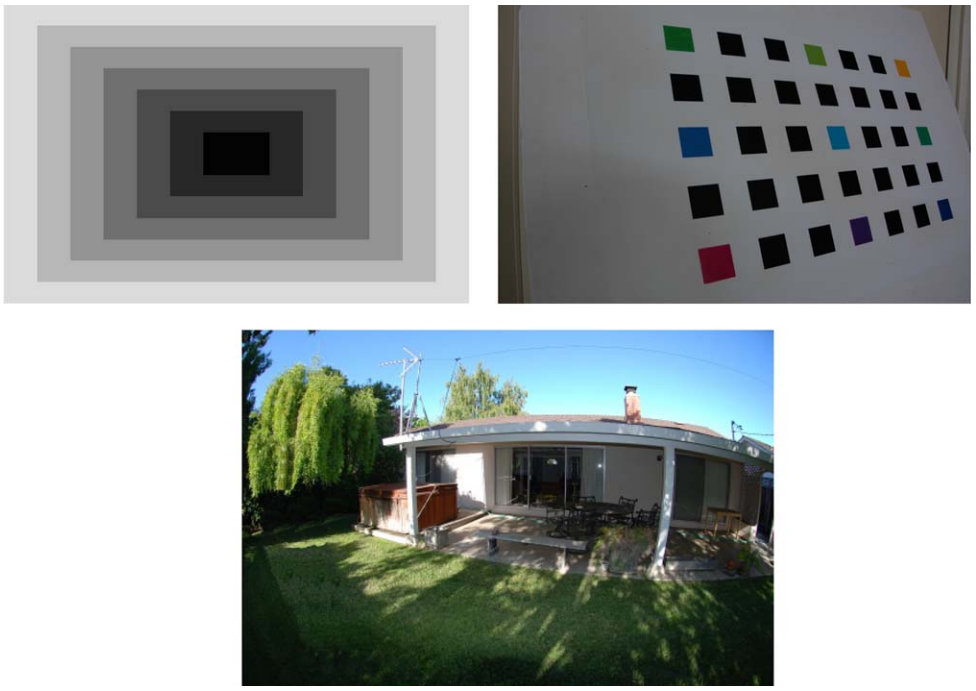


Fig. 2 Test images used in the numerical experiments

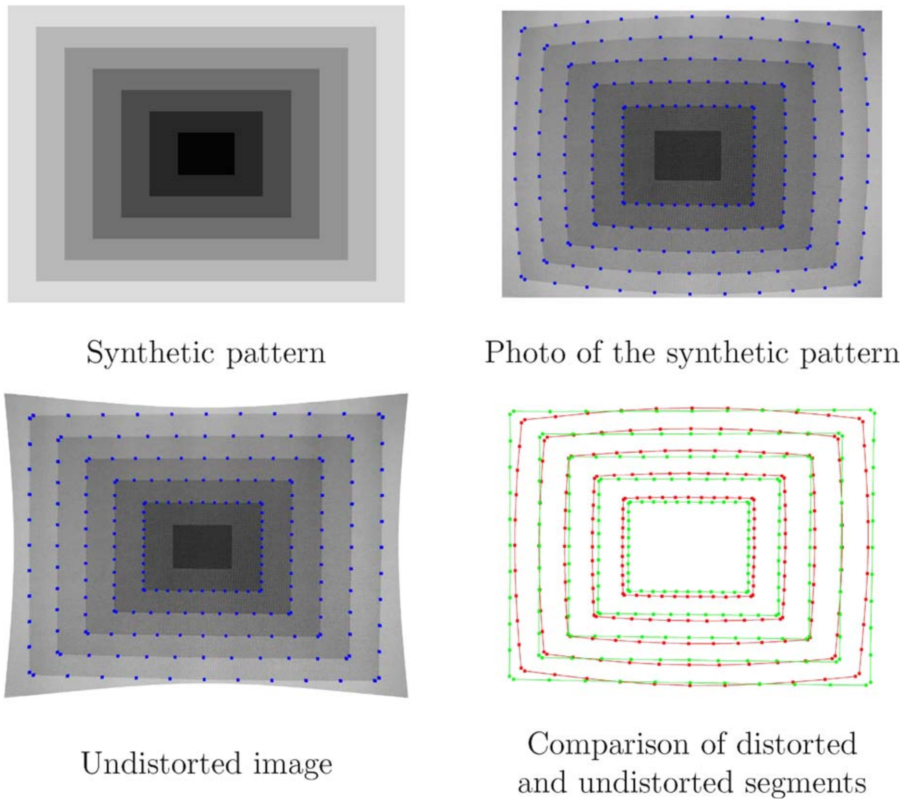

Synthetic pattern
Photo of the synthetic pattern
Undistorted image
Comparison of distorted and undistorted segments

Fig. 3 Illustration of the results obtained on the synthetic pattern using the proposed technique

Table 1 Comparison between algebraic method and numerical method for the geometric pattern starting from the trivial solution (Number of lines: 20 and number of points: 11 each line). The stop criteria for the numerical method has been established to a tolerance of 1e−4. $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ values are normalized with respect to the trivial setup

$N = 4$	Initial Value	Solution $N = 2$		Solution $N = 4$	
		Algebraic	Steepest descent	Algebraic	Steepest descent
k_0	1	9.3374e−01	9.3313e−01	9.1696e−01	9.1799e−01
k_2	0	1.1617e−07	1.1722e−07	1.6575e−07	1.6317e−07
k_4	0	–	–	−2.5597e−14	−2.4600e−14
$D(\mathbf{k})$	1	0.013826	0.013728	0.003391	0.003357
$\hat{E}(\mathbf{k})$	1	0.012033	0.012092	0.002805	0.002866
Number of iterations		1	2	1	6
Function evaluations		–	25	–	145
CPU Time (ms)		≈ 0	≈ 16	≈ 15	≈ 78

As it has been mentioned before, no iterations are needed to get the solution using the algebraic method. If we look at the CPU time, we can see that the algebraic method is around five times faster than the steepest descent for the 4-degree polynomial case. However, for the 2-degree polynomial case, the gradient descent iterative approach solves the problem in two gradient iterations, which was also the expected if we take into consideration the geometric profiles of the distance function (see Figs. 4 and 6), but it requires 25 function evaluations and a measurable CPU time (the algebraic method is practically instantaneous).

We use an standard gradient method (steepest descent method) to minimize $D(\mathbf{k})$ function after examining the geometric profiles of the distance function, which is clearly a smooth non convex function. However, we are aware about the zigzagging behaviour close to the solution that a simple gradient method exhibits (increasing the number of function evaluations), which can be avoided using a conjugate method or a second order method (Newton or quasi-Newton). An exact quadratic fit line search (TPP, three point pattern condition) has been implemented for the inner local search (see [20]). This local search approach provides global convergence under appropriate assumptions such as pseudoconvexity, which makes it useful for the problem we are dealing with. This kind of search is normally used but, as a main drawback, it requires a number of function eval-

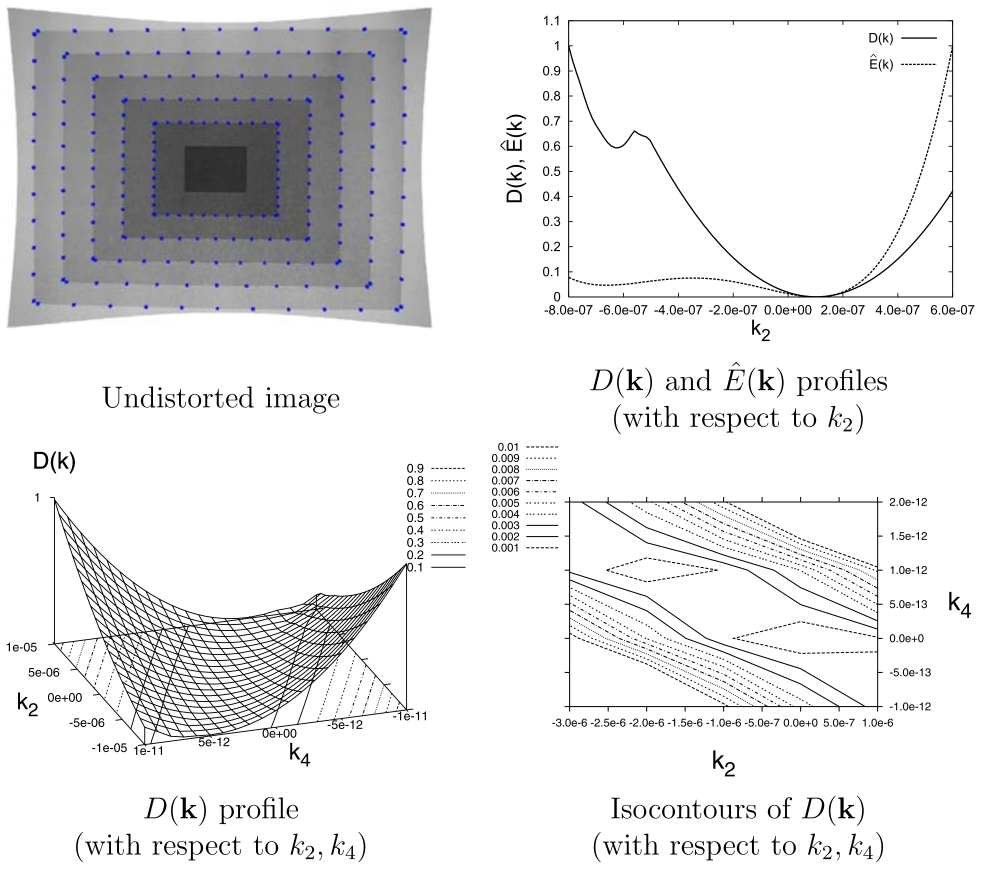


Fig. 4 Illustration of the profiles of $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ with respect to k_2 and k_4

uations higher than, for instance, a Newton search. When applying the steepest descent algorithm, a first-order finite difference method is used to estimate the numerical derivatives. Results show that this simple derivative estimation behaves well for all analyzed cases. When the gradients are found using finite differences, a gradient will cost as many function evaluations as there are nonzeros in the gradient. Therefore, it can be said that all the function evaluations come mainly from the inner search (searching for the TPP points which only need function evaluations). The stopping criteria used is as follows: we run iterations until $(D(\mathbf{k}) - D(\mathbf{k}^*)/\max(1, D(\mathbf{k}^*))$ is less than the specified accuracy, with $D(\mathbf{k}^*)$ being the previous solution and $D(\mathbf{k})$ the actual one. This is a normally used stopping criteria for gradient-like methods (see for instance, [21]). We remark that, as a main difference from the algebraic method presented here, and inspired in the lens distortion measure $\hat{E}(\mathbf{k})$, to numerically minimize the function $D(\mathbf{k})$, no simplification is needed.

As it can be deduced from the above results, if we consider $D(\mathbf{k})$ as the solution to compare with, one observes that our algebraic approach suits well for all the analyzed cases, with a relative error around 1% for the final $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ values. Besides, the distortion error measure, $\hat{E}(\mathbf{k})$ is reduced up to a factor of 1000 for the 4-degree polynomial compared to the trivial solution. Therefore, through comparing the solution precision attained between both methods, the algebraic approach can be regarded as a valid outstanding approach for the lens distortion problem.

To illustrate the visual effect of the obtained undistorted image, we present in Fig. 3 (left bottom corner) the undistorted image using the lens distortion model with the lower $D(\mathbf{k})$ value given by $k_0 = 9.1696\text{e}{-01}$, $k_2 = 1.6575\text{e}{-07}$ and $k_4 = -2.5597\text{e}{-14}$ (Table 1, last column). The corrected image (undistorted) and the location of edges in the distorted and undistorted image are also represented in the same figure.

The corrected image is again shown in Fig. 4 (left top corner). We can also see the $D(\mathbf{k})$ and the $\hat{E}(\mathbf{k})$ profiles for the 2-degree polynomial case (right top corner) which corresponds to the distortion parameters of Table 1, third column ($k_0 = 9.3374\text{e}{-01}$, $k_1 = 0$ and $k_2 = 1.1617\text{e}{-07}$). To draw both functions in an unique plot, $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ have been normalized respect to their highest value. In that figure, one notices the location of the $\hat{E}(\mathbf{k})$ optimal solution, which lies close to $D(\mathbf{k})$ global minimum, and hence validates our proposal. It is included in the same figure (left bottom and right bottom corner) a two dimensional representation and the isocontours map of $D(\mathbf{k})$ with respect to k_2 and k_4 for the distortion parameters of the last column of Table 1. Note

Fig. 5 Illustration of the proposed technique on a real image

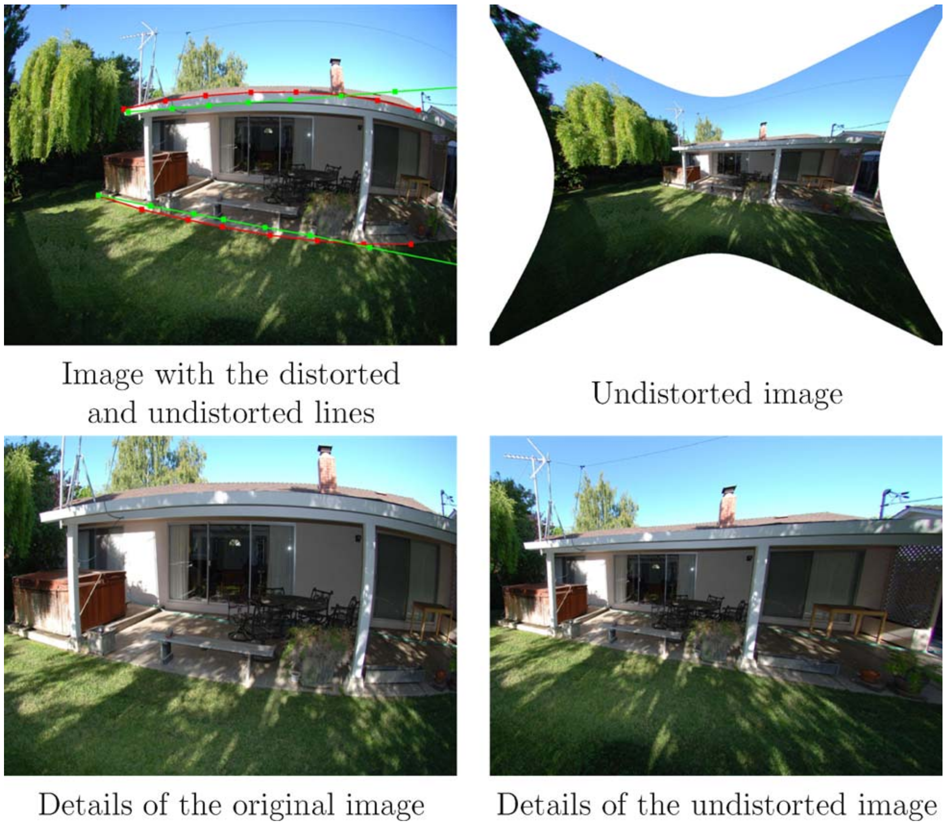


Image with the distorted and undistorted lines
Undistorted image
Details of the original image
Details of the undistorted image

Fig. 6 Illustration of the profiles of $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ with respect to k_2 and k_4

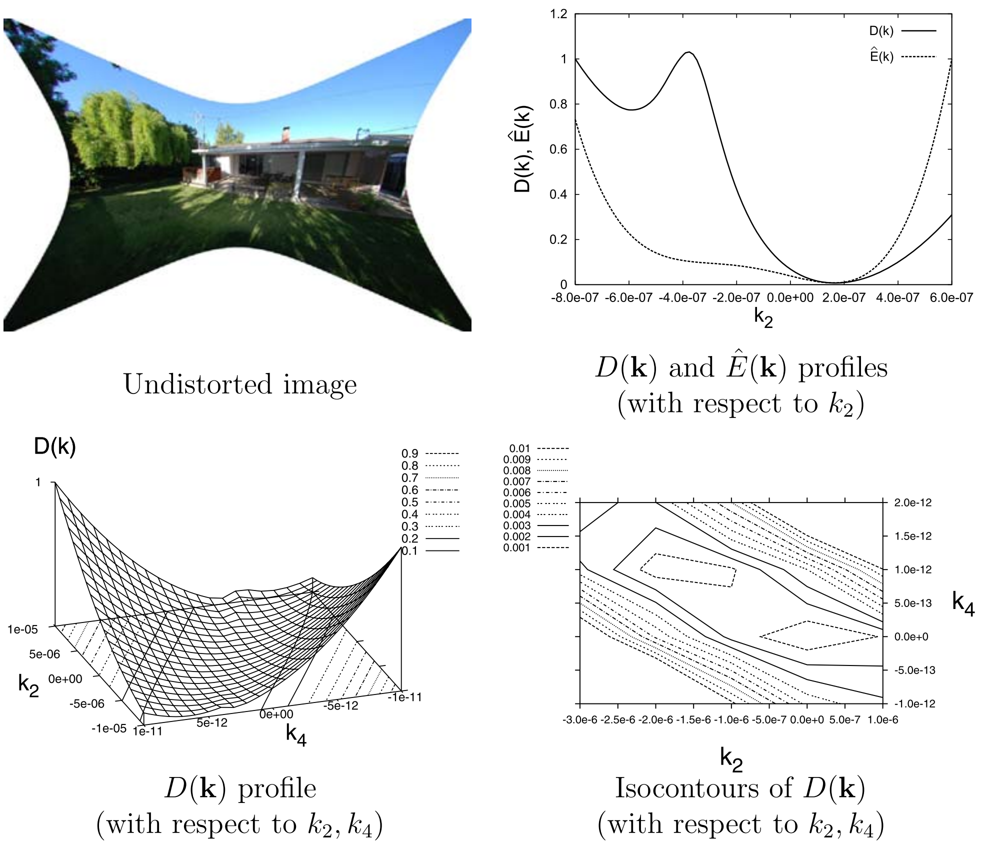


Table 2 Comparison between algebraic method and numerical method for the 2 lines photo starting from the trivial solution. The stop criteria for the numerical method has been established to a tolerance of 1e−4. $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ values are normalized with respect to the trivial setup

$N=4$	Initial Value	Solution $N=2$		Solution $N=4$	
		Algebraic	Steepest descent	Algebraic	Steepest descent
k_0	1	8.2045e−01	8.2045e−01	8.4924e−01	8.4924e−01
k_2	0	1.6117e−07	1.6803e−07	4.4743e−08	4.4456e−08
k_4	0	–	–	5.1485e−14	5.1687e−14
$D(\mathbf{k})$	1	0.071948	0.071077	0.002534	0.002534
$\hat{E}(\mathbf{k})$	1	0.088919	0.090003	0.003357	0.003360
Number of iterations		1	2	1	5
Function evaluations		–	37	–	140
CPU Time (ms)		≈ 0	≈ 0	≈ 0	≈ 0

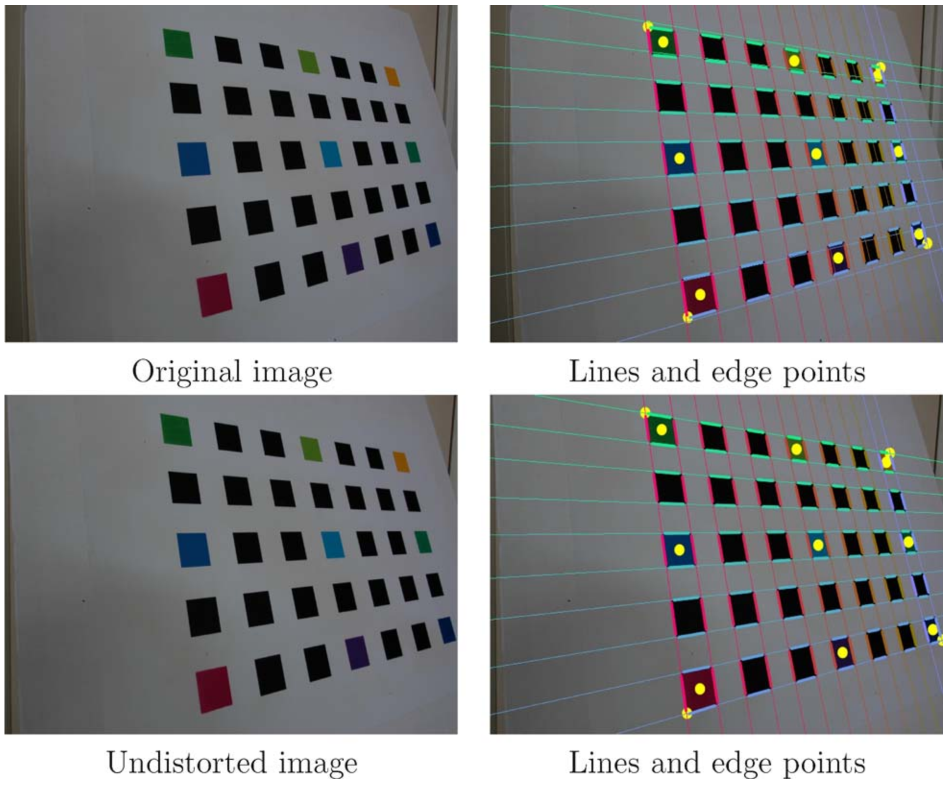


Fig. 7 Illustration of the proposed technique on an standard calibration pattern

that both functions are non convex which assures that a numerical optimization algorithm (such a descent gradient) can be trapped into the local minima.

Next results are for a photograph having an important barrel distortion. The original image (width = 3872 pixels, height = 2592 pixels) is shown in Fig. 2 (center) and repeated in Fig. 5 (left top corner) as well as the corrected image. To remark the robustness of our implementation, we tried to correct the distortion from a reduced number of distorted sample points (eighteen), to be aligned along two straight lines. We used a 2-degree and a 4-degree polynomial in the distortion parameters k_i to compare solutions. The achieved quantitative results are presented in Table 2. Results are similar to the above presented for the geometric pattern, therefore the algebraic approach remains valid for all the cases. Once again, the quality of the solution improves with the polynomial degree and $\hat{E}(\mathbf{k})$ is reduced up to a factor of 1000 from the trivial solution. The number of gradient iterations, number of function evaluations and the CPU time have been dramatically reduced due to the number of image data points to deal with are quite low (eighteen).

To finish we present the next result to show that the proposed method works well for all kind of lines orientation. It consists of a more complex geometric pattern (see Fig. 7) including a set of diagonal lines (in the previous results, there were only horizontal and vertical lines) and a considerable number of data points (number of lines is 24 and the number of points is around 400 each line). Results after applying the

Table 3 Comparison between algebraic method and numerical method for the second calibration pattern starting from the trivial solution (Number of lines: 24 and number of points: around 400 each line). The stop criteria for the numerical method has been established to a tolerance of 1e−4. $D(\mathbf{k})$ and $\hat{E}(\mathbf{k})$ values are normalized with respect to the trivial setup

$N=4$	Initial Value	Solution $N=2$		Solution $N=4$	
		Algebraic	Steepest descent	Algebraic	Steepest descent
k_0	1	1.0000e+00	1.0000e+00	1.0000e+00	1.0000e+00
k_2	0	1.3650e−07	1.3826e−07	1.7735e−07	1.7647e−07
k_4	0	–	–	−7.9658e−14	−7.0131e−14
$D(\mathbf{k})$	1	0.043591	0.043442	0.031417	0.030210
$\hat{E}(\mathbf{k})$	1	0.053891	0.053946	0.038441	0.038472
Number of iterations		1	2	1	14
Function evaluations		–	26	–	322
CPU Time (s)		≈ 0	≈ 0.4690	≈ 0.0310	≈ 5.6250

Table 4 Estimated distortion parameter and objective function value after applying a steepest descent optimization algorithm from the algebraic method solution for the second geometric pattern. The stop criteria for the numerical method has been established to a tolerance of 1e−4. $D(\mathbf{k})$ values are normalized with respect to the trivial setup

k_0	k_2	k_4	$D(\mathbf{k})$	Iterations	Function evaluations	CPU Time (s)
1.0000e+00	0	–	0.043442	2	26	≈ 0.4690
1.0000e+00	1.3650e-07	–	0.043442	2	26	≈ 0.4690
1.0000e+00	0	0	0.030210	14	322	≈ 5.6250
1.0000e+00	1.7735e−07	−7.9658e−14	0.030181	2	28	≈ 0.5000

Table 5 Dependency of CPU time respect to stop criteria for the numerical method applied to minimize $D(\mathbf{k})$ from the trivial solution for the second geometric pattern. $D(\mathbf{k})$ values are normalized with respect to the trivial setup

$N=4$	$D(\mathbf{k})$	Iterations	Function evaluations	CPU Time (s)
Algebraic	3.141701491392861e−02	1	–	≈ 0.0310
$\lvert tol\rvert = 1.0\text{e}{-2}$	3.031493383334863e−02	7	155	≈ 2.7030
$\lvert tol\rvert = 1.0\text{e}{-4}$	3.021124372302807e−02	14	322	≈ 5.6250
$\lvert tol\rvert = 1.0\text{e}{-6}$	3.019263303802254e−02	29	670	≈ 11.5620
$\lvert tol\rvert = 1.0\text{e}{-8}$	3.019263303802254e−02	29	670	≈ 11.5940

algebraic and the numerical optimization method are listed in Table 3. Similar conclusions as above can be considered about the quality of the solutions, however, CPU time comparison between both methods reveals that for complex images, the algebraic method is around 180 times faster than the gradient method which we consider as a another relevant contribution of this work.

Due to the non convexity properties of the functions $\hat{E}(\mathbf{k})$ and $D(\mathbf{k})$, it seems reasonably to use the algebraic solution as a starting point for a fast local minimizer algorithm to improve the result. This is also another use of the proposed method. The results after executing the descent algorithm from the distortion parameters of Table 3 are listed in Table 4 for the 2-degree and 4-degree polynomial cases. As it was expected, there is no need to improve the solution for the 2-degree case because is a global solution. For the 4-degree polynomial case, to get a final solution at a low CPU time, it is advisable to first execute the algebraic method and then to run a numerical method using the algebraic solution as a starting point. Note that, to get the solution (0.030), the steepest descent method requires 322 function evaluations and 5.6250 seconds from the trivial solution $\mathbf{k}=0$. If the gradient method starts from the algebraic solution ($k_0 = 1.0000\text{e}{+00}$, $k_2 = 1.7735\text{e}{-07}$, $k_4 = -7.9658\text{e}{-14}$), it is only necessary 28 function evaluations (it means a reduction of a factor of 10) and 0.5310 seconds $(0.0310 + 0.5000)$, that is, a CPU reduction of a factor of 10.

A question of significance is how to choose the final tolerance when solving a optimization problem. If the tolerance is too small, obtaining such a result can be very expensive in computing time. If the tolerance is too large, the solution will be poorly defined. All the results provided by the gradient algorithm were obtained with a tolerance of 1e−4. This value has been established because it gives a reasonably compromise between solution precision and CPU time as it is shown in Table 5. The gradient zigzagging behaviour can be observed and, increasing the tolerance beyond 1e−4 implies more CPU time and little reduction of the objective function value.

6 Conclusions

In this paper we present an algebraic approach to radial lens distortion parameter estimation based on edge line rectification. We propose a polynomial distortion error measure based on the Cauchy-Schwarz inequality. We present an algebraic analysis of the distortion error measure. From a mathematical point of view the method is well founded, elegant and it provides a direct solution to the problem.

We have implemented the proposed method for 1 or 2 distortion parameter models. The distortion parameters are obtained using the algebraic resultant-based method which estimates the global minimum of the functional without iterations. The parameter model is also updated using a zoom factor which minimizes the square distance between the distorted and undistorted edge points.

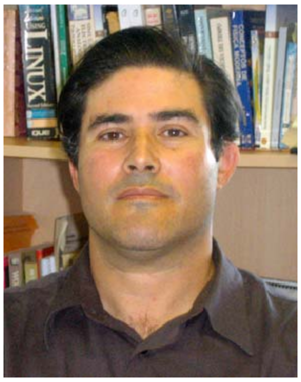
The numerical experiments we have presented are very promising. The lens distortion of the edge lines is strongly reduced in the undistorted image and the barrel distortion is properly removed. We show that the obtained algebraic minimum is very close to the solution obtained by minimizing the square distance of the edge points to a straight line. We have also shown that in the case we deal with a large number of edge points, the proposed algebraic method is much faster than the usual gradient descent iterative method. Another important advantage of our method is that it does not require initialization for the distortion parameter k_i. In particular it can be used as initialization of the distortion parameters in bundle adjustment calibration techniques.

Acknowledgement We thanks the paper reviewers for their valuable comments and suggestions.

References

1. Faugeras, O.: Three-Dimensional Computer Vision. MIT Press, Cambridge (1993)
2. Faugeras, O., Luong, Q.-T., Papadopoulo, T.: The Geometry of Multiple Images. MIT Press, Cambridge (2001)
3. Tsai, R.Y.: A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. IEEE J. Robot. Autom. **3**(4), 323–344 (1987)
4. Brown, D.C.: Close-range camera calibration. Photogramm. Eng. **37**, 855–866 (1971)
5. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2004)
6. McGlone, C. (ed): Manual of Photogrammetry, 4th edn. Am. Soc. of Photogrammetry, Falls Church (1980)
7. Brown, D.C.: Decentering distortion of lenses. Photogramm. Eng. **32**(3), 444–462 (1966)
8. Kang, S.B.: Semiautomatic methods for recovering radial distortion parameters from a single image. Digital CRL. Technical Report CRL 97/3 (1997)
9. Swaminathan, R., Nayar, S.K.: Non-metric calibration of wide-angle lenses and polycameras. IEEE Trans. Pattern Anal. Mach. Intell. **22**(10), 1172–1178 (2000)
10. Devernay, F., Faugeras, O.: Straight lines have to be straight. Mach. Vis. Appl. **13**(1), 14–24 (2001)
11. El-Melegy, T.M., Farag, A.A.: Nonmetric lens distortion calibration: Closed-form solutions, robust estimation and model selection. In: ICCV 03: Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 1(1), pp. 554–559 (2003)
12. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)
13. Steele, J.M.: The Cauchy Schwarz Master Class. Cambridge University Press, Cambridge (2004)
14. Winkler, F.: Polynomial Algorithms in Computer Algebra. Springer, New York (1996)
15. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms, 2nd edn. Springer, New York (1997)
16. Sendra, J.R., Winkler, F., Perez-Daz, S.: Rational Algebraic Curves: A Computer Algebra Approach. Springer, Berlin (2007)
17. Cox, D., Little, J., O'Shea, D.: Using Algebraic Geometry. Springer, New York (1998)
18. Deriche, R.: Fast algorithms for low-level vision. IEEE Trans. Pattern Anal. Mach. Intell. **1**(12), 78–88 (1990)
19. Aubert, G., Kornprobst, P.: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, 2nd edn. Applied Mathematical Sciences, vol. 147. Springer, Berlin (2006)
20. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. Wiley, New York (1993)
21. Jonasson, K.: A projected conjugate gradient method for sparse minimax problems. Numer. Algorithms **5**, 309–323 (1993)

Luis Alvarez has received a M.Sc. in applied mathematics in 1985 and a Ph.D. in mathematics in 1988, both from Complutense University (Madrid, Spain). Between 1991 and 1992 he worked as post-doctoral researcher at CEREMADE laboratory in the computer vision research group directed by Prof. Jean-Michel Morel. Since 2000 he is full professor at the University of Las Palmas of Gran Canaria (ULPGC). He has created the research group Análisis Matemático de Imágenes (AMI) at the ULPGC. He is an expert in computer vision and applied mathematics. His main research interest areas are the applications of mathematical analysis to computer vision including problems like multiscale analysis, mathematical morphology, optic flow estimation, stereo vision, shape representation, medical imaging, synthetic image generation, camera calibration, etc.



Luis Gómez has received a M.Sc. in Physics in 1988 (UNED, Madrid, SPAIN) and a Ph.D. in Telecommunication engineering in 1992 (University of Las Palmas de Gran Canaria (ULPGC, SPAIN). Since 1994 he is an assistant professor at the University of Las Palmas of Gran Canaria (ULPGC). He is an expert in nonlinear optimization and applied mathematics to computer vision. His main research interest areas are the applications of optimization to engineering problems, such as ultrasound imaging, camera calibration, shape complementarity, etc. He is working at the AMI (Análisis Matemático de Imágenes, ULPGC) group directed by professor Luis Álvarez.

J. Rafael Sendra is Full Professor of Applied Mathematics at Alcalá de Henares University (Madrid). His main research interest includes symbolic computation and its applications, approximate algebraic methods, effective algorithms in real and complex algebraic geometry, theory and algorithms for curves and surfaces, and applications to computer aided geometric design. He received his Ph.D. degree in Mathematics from Alcalá de Henares University in 1990.