# Globally Convergent Autocalibration using Interval Analysis

A. Fusiello,
Dipartimento di Informatica
Università degli studi di Verona
I-37134 Verona, Italy
fusiello@sci.univr.it
A. Benedetti
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125
arrigo@vision.caltech.edu
M. Farenzena, and A. Busti
Facoltà di Scienze MM. FF. NN.
Università degli studi di Verona
I-37134 Verona, Italy

10 Luglio 2003
Rapporto di Ricerca RR 09/2003
Dipartimento di Informatica
Università degli studi di Verona

**Abstract**

We address the problem of autocalibration of a moving camera with unknown constant intrinsic parameters. Existing autocalibration techniques use numerical optimization algorithms whose convergence to the correct result cannot be guaranteed, in general. To address this problem, we have developed a method where an interval branch-and-bound method is employed for numerical minimization. Thanks to the properties of Interval Analysis this method converges to the global solution with mathematical certainty and arbitrary accuracy, and the only input information it requires from the user are a set of point correspondences and a search box. The cost function is based on the Huang-Faugeras constraint of the fundamental matrix, and a closed form expression for its Jacobian and Hessian matrices is derived through matrix differential calculus. A recently proposed interval extension based on Bernstein polynomial forms has been investigated to speed up the search for the solution. Finally, experimental results on synthetic and real images are presented.

Currently affiliated with KLA-Tencor, One Technology Dr., Milpitas, CA 95035. Email: Arrigo.Benedetti@kla-tencor.com.

# I. INTRODUCTION

The goal of Computer Vision is to compute properties (mainly geometric) of the three-dimensional world from images. One of the challenging problems of Computer Vision is to *reconstruct* a three-dimensional model of the scene from a moving camera. Most of the earlier studies in the field assume that the intrinsic parameters of the camera (focal length, image center and aspect ratio) are known. Computing camera motion in this case is a well known problem for which several methods are available (see [1] for a review). Given all the parameters of the camera, reconstruction is straightforward.

However, there are situations where the intrinsic parameters are unknown and the camera is not accessible (e.g. when using stock footage). In these cases the only information one can exploit are contained in the video sequence itself.

The classical approach to *autocalibration* (or *self-calibration*), in the case of a single moving camera with constant but unknown intrinsic parameters, is based on the Kruppa equations [2], which have been found to be very sensitive to noise [3], possibly due to the instability in the computation of the epipole [4]. Indeed, formulations which avoid the epipole seems to be more stable [5], [4].

Other methods [6], [7], [8], based on the *stratification* approach, upgrade a projective reconstruction to an Euclidean one without solving explicitly for the intrinsic parameters (see [9] for a review). The constant intrinsic parameters constraint has been released in [10], [11], by assuming that some other parameters are known.

Recently, Mendonça and Cipolla [12] presented an algorithm which directly recovers the intrinsic parameters from fundamental matrices, like the Kruppa equations, but it is simpler and copes with varying parameters.

Under the assumption that only the (varying) focal length is unknown, closed form and linear solutions can be obtained [13], [14], [5], [15]. In all the other cases the parameters come from the solution of a system of polynomial equations or from the minimization of a non-linear function. In principle, continuation (homotopy) techniques could be applied to the former case, though—in practice—iterative minimization techniques must be used [3], as homotopy algorithms are applicable only in the case of few displacements, and can give rise to bifurcation phenomena. When minimizing a non-linear function by gradient descent methods, convergence to the global minimum is not guaranteed: it depends on the initialization—for deterministic algorithms,—or it is guaranteed only in probability—for stochastic algorithms [16]. Quasi-linear approaches reduce the sensitivity to the initial guess [17], [8], [18], but they do not solve the problem. The solutions of a simpler problem (only focal is unknown) have been used to initialize the minimization in [11], [19]. In [20], a stratified approach has been proposed, based on the direct evaluation of a dense sampling of the search space. Albeit some of these techniques are effective, none of the existing methods is provably convergent.

In this paper we introduce a method for autocalibration that is *guaranteed* to converge to the global minimum, regardless of the starting point. In the same spirit of [12], [4], [16], we compute directly the intrinsic parameters from fundamental matrices. We assume constant intrinsic parameters, but the technique is flexible and can be adapted to varying parameters as well.

The minimization algorithm is based on Interval Analysis (IA) [21], a branch of numerical analysis that has received increasing attention during the last decade and has been strangely overlooked by the computer vision community.

Classical numerical optimization methods for the multidimensional case start from some approximate trial points and sample the objective function at only a finite number of points. There is no way to guarantee that the function does not have some unexpectedly small values between these trial points, without making specific assumptions. On the contrary, IA optimization algorithms [22] evaluate the objective function over a continuum of points, including those points that are not finitely representable on the computer. They solve the optimization problem with *automatic result verification*, i.e. with the guarantee that the global minimizers have been found.

The rest of the paper is structured as follows. The next section introduces notation and some background notions of Computer Vision. The autocalibration problem that we address is formulated in Sec. III. In

Sec. IV the reader is first introduced to Interval Analysis, and the specific optimization algorithm is described. Results on synthetic and real data are reported in Sec. V, and conclusions are drawn in Sec. VI. Appendix I is devoted to the derivation of the expressions for Jacobian and Hessian of the cost function, whereas in Appendix II we give the details on the algorithm used for recovering scene structure.

## II. Background

Throughout this paper we will use the general projective camera model [23]. Let $w = [x, y, z, 1]^{\mathsf{T}}$ be the homogeneous coordinates of a 3D point in the world reference frame. The homogeneous coordinates of the projected image point are given by[1]

$$m \backsim P\, w, \tag{1}$$

where $P \triangleq A\,[R|t]$ is the camera matrix, whose position and orientation are represented, respectively, by the translation vector $t$ and the $3 \times 3$ rotation matrix $R$. The matrix $A$ contains the *intrinsic parameters*, and has the following form:

$$A = \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where $\alpha_u$, $\alpha_v$ are the *focal lengths* in horizontal and vertical pixels, respectively, $(u_0, v_0)$ are the coordinates of the *principal point*, given by the intersection of the optical axis with the retinal plane, and $\gamma$ is the *skew* factor, that models non-rectangular pixels.

Two conjugate points $m$ and $m'$ are related by the *fundamental matrix* $F$:

$$m'^{\mathsf{T}} F m = 0 \tag{3}$$

The rank of $F$ is in general two and, being defined up to a scale factor, it depends upon seven parameters. Its computation requires a minimum of eight conjugate points to obtain a unique solution [24]. $F$ depends on the intrinsic and extrinsic parameters according[2] to

$$F \backsim A'^{-\mathsf{T}}([t]_\times R)A^{-1}. \tag{4}$$

When conjugate points are in normalized coordinates ($A^{-1}m$), i.e., intrinsic parameters are known, one obtains the *essential matrix*:

$$E \backsim [t]_\times R. \tag{5}$$

The essential matrix encodes the rigid transformation between the two cameras, and it depends upon five independent parameters: three for the rotation and two for the translation up to a scale factor.

## III. Problem Formulation

In many practical cases, the intrinsic parameters are unknown and point correspondences are the only information that can be extracted from a sequence of images. *Autocalibration* consists in computing the intrinsic parameters, or—in general—recovering the Euclidean *stratum*, starting from point correspondences. In this section we will see which constraints are available for autocalibration.

As we saw in Sec. II, the epipolar geometry of two views is described by the fundamental matrix, which depends on seven parameters. Since the five parameters of the essential matrix are needed to describe the rigid displacement, two independent constraints are available for the computation of the intrinsic parameters from the fundamental matrix. Indeed, the essential matrix is characterized by the following Theorem [25], [13]:

[1] $\backsim$ denotes equality up to a scale factor.

[2] $[t]_\times$ is the skew-symmetric matrix associated with the cross-product.

*Theorem 1:* A real $3 \times 3$ matrix $E$ can be factored as the product of a nonzero skew-symmetric matrix and a rotation matrix if and only if $E$ has two identical singular values and one zero singular value.

By exploiting this constraint, Hartley [13] derived two quadratic equations in the two values of the focal length. He also pointed out that no more information could be extracted from the fundamental matrix without making additional assumptions (e.g. constant intrinsic parameters).

It can be shown (see Sec. III-A) that the conditions on the singular values are equivalent to:

$$\det(E) = 0 \quad \wedge \quad 2\,\mathrm{tr}(EE^{\mathsf{T}})^2 - \mathrm{tr}^2(EE^{\mathsf{T}}) = 0, \tag{6}$$

which in turn is equivalent to the Kruppa equations [3]. The second clause of (6) can be decomposed [3] in two independent polynomial constraints.

All these constraints are algebraic interpretations of the so-called *rigidity constraint*, namely the fact that for any fundamental matrix $F$ there exist two intrinsic parameters matrix $A$ and $A'$ and a rigid motion represented by $t$ and $R$ such that (4) is satisfied.

The autocalibration method by Mendonça and Cipolla is based on Theorem 1. They designed a cost function which takes the intrinsic parameters as arguments, and the fundamental matrices as parameters, and returns a positive value proportional to the difference between the two non-zero singular value of the essential matrix. Let $F_{ij}$ be the fundamental matrix relating views $i$ and $j$ (computed from point correspondences), and let $A_i$ and $A_j$ be the respective (unknown) intrinsic parameter matrices. The cost function is

$$\chi(A_i,\ i \triangleq 1 \ldots n) = \sum_{i=1}^{n} \sum_{j>n}^{n} w_{ij} \frac{{}^1\sigma_{ij} - {}^2\sigma_{ij}}{{}^1\sigma_{ij} + {}^2\sigma_{ij}}, \tag{7}$$

where ${}^1\sigma_{ij} > {}^2\sigma_{ij}$ are the non zero singular values of

$$E_{ij} = A_i^{\mathsf{T}} F_{ij} A_j, \tag{8}$$

and $w_{ij}$ are normalized weight factors. In the general case of $n$ views, the $n(n-1)/2$ fundamental matrices are not independent, neither are the $n(n-1)/2$ constraints that can be derived from them. It can be shown [11] that, if $n_k$ parameters are known and $n_c$ parameters are constant, the unknown intrinsic parameters can be computed provided that

$$n(n_k + n_c) \geq 8 + n_c. \tag{9}$$

For example, if the intrinsic parameters are constant, three views are sufficient to recover them. If the skew is zero and the other parameters are varying, at least eight views are needed.

### *A. The Huang-Faugeras cost function*

The use of (7) as an optimization criterion has been considered, however it has posed several problems. First, its Hessian matrix is singular at the solution (${}^1\sigma_{ij} = {}^2\sigma_{ij}$), which can lead to higher run times of the optimization procedure [22]. Secondly, bounding the ranges of ${}^1\sigma_{ij}$ of an interval essential matrix with wide entries is not trivial, since it requires the solution of a min-max optimization problem. For these reasons we seek to minimize a cost function based on the equivalent constraint given by (6).

In the same spirit of the Mendonça-Cipolla algorithm, we minimize

$$\chi(A_i) \triangleq \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} \frac{2\,\mathrm{tr}(E_{ij}E_{ij}{}^{\mathsf{T}})^2 - \mathrm{tr}^2(E_{ij}E_{ij}{}^{\mathsf{T}})}{\mathrm{tr}^2(E_{ij}E_{ij}{}^{\mathsf{T}})}. \tag{10}$$

By using the property that the trace of a square matrix $X$ is equal to the sum of its eigenvalues and the property that the eigenvalues of $XX^{\mathsf{T}}$ are equal to the squares of the singular values of $X$, we can write:

$$\mathrm{tr}(EE^{\mathsf{T}})^2 = \sum_{k=1}^{3} \sigma_k^4(E). \tag{11}$$

Hence, the left hand side of (6) can be rewritten as

$$\begin{aligned} 2\,\mathrm{tr}(EE^{\mathsf{T}})^2 - \mathrm{tr}^2(EE^{\mathsf{T}}) = \\ 2(\sigma_1^4 + \sigma_2^4 + \sigma_3^4) - (\sigma_1^2 + \sigma_2^2 + \sigma_3^2)^2 = \\ (\sigma_1^2 - \sigma_2^2)^2 + \sigma_3^2(\sigma_3^2 - 2(\sigma_1^2 + \sigma_2^2)). \end{aligned} \tag{12}$$

Therefore, provided that $\sigma_3 = 0$, the cost function expressed by (10) is the square of the Mendonça-Cipolla function (7). The essential matrix $E$ is derived from the fundamental matrix via (8); if $F$ is computed with an algorithm that enforces its rank to be two, then $\sigma_3 = 0$. Since the left hand side of (6) is always positive, we do not need to take its square, as it would be required in a generic least squares problem. This is a very desirable property, since it reduces the order of the numerator and the denominator of the cost function from sixteen to eight.

In the following we assume that the intrinsic parameters of the camera are constant for the $n$ views, i.e.

$$\chi(A_i) \triangleq \chi(A) \triangleq \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} \left( \frac{2\,\mathrm{tr}(E_{ij} {E_{ij}}^{\mathsf{T}})^2}{\mathrm{tr}^2(E_{ij} {E_{ij}}^{\mathsf{T}})} - 1 \right). \tag{13}$$

## IV. Interval Analysis

Interval Arithmetic [26] is an arithmetic defined on intervals, rather than on real numbers. In the beginning, Interval Arithmetic was mainly employed for bounding the measurement errors of physical quantities for which no statistical distribution was known. Later on it was leveraged to a broad new field of applied mathematics, aptly named Interval Analysis, where rigorous proofs are the consequence of numerical computations.

### *A. Notation and useful results*

In the sequel of this section we shall follow the notation used in [27], where intervals are denoted by boldface, scalar quantities are denoted by lower case letters and vectors and matrices are denoted by upper case. Brackets "$[\cdot]$" will delimit intervals, while parentheses "$(\cdot)$" will delimit vectors and matrices. Underscores and overscores will represent respectively lower and upper bounds of intervals. An interval $\boldsymbol{x}$ is called *degenerate* when $\underline{\boldsymbol{x}} = \overline{\boldsymbol{x}} = x$. $\mathbb{IR}$ and $\mathbb{IR}^n$ stand respectively for the set of real intervals and the set of real interval vectors of dimension $n$. The midpoint of an interval $\boldsymbol{x}$ is denoted by $\mathrm{m}(\boldsymbol{x})$, and the vector whose entries are midpoints of the entries of $\boldsymbol{X} \in \mathbb{IR}^n$ is denoted by $\mathrm{m}(\boldsymbol{X})$. The *width* of $\boldsymbol{x}$ is defined as $\mathrm{w}(\boldsymbol{x}) = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}$. If $\boldsymbol{X} \in \mathbb{IR}^n$ then $\mathrm{w}(\boldsymbol{X}) = \max\{\mathrm{w}(\boldsymbol{x}_i),\ i = 1, \ldots, n\}$. If $f(x)$ is a function defined over an interval $\boldsymbol{x}$ then $\boldsymbol{f}^u(\boldsymbol{x})$ denotes the range of $f(x)$ over $\boldsymbol{x}$. Similarly, the range of $F : \mathbb{R}^n \to \mathbb{R}$ over $\boldsymbol{X}$ is denoted by $\mathbf{F}^u(\boldsymbol{X})$.

Interval arithmetic is an arithmetic defined on sets of intervals. If $\boldsymbol{x} = [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$ and $\boldsymbol{y} = \left[\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}\right]$, a binary operation in the *ideal interval arithmetic* between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as:

$$\begin{gathered} \boldsymbol{x}\,\mathrm{op}\,\boldsymbol{y} \triangleq \{x\,\mathrm{op}\,y \mid x \in \boldsymbol{y} \text{ and } y \in \boldsymbol{y}\}, \\ \text{for } \mathrm{op} \in \{+, -, \times, \div\}. \end{gathered}$$

Thus, the ranges of the four elementary interval operations are exactly the ranges of the corresponding

real operations. The operational definitions for the four elementary interval arithmetic operations are

$$\begin{aligned}
\boldsymbol{x}+\boldsymbol{y} &\triangleq \left[\underline{\boldsymbol{x}}+\underline{\boldsymbol{y}}, \overline{\boldsymbol{x}}+\overline{\boldsymbol{y}}\right], \\
\boldsymbol{x}-\boldsymbol{y} &\triangleq \left[\underline{\boldsymbol{x}}-\overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}-\underline{\boldsymbol{y}}\right], \\
\boldsymbol{x}\times\boldsymbol{y} &\triangleq \left[\min\left\{\underline{\boldsymbol{x}}\ \underline{\boldsymbol{y}}, \underline{\boldsymbol{x}}\ \overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}\ \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}}\ \overline{\boldsymbol{y}}\right\},\right. \\
&\quad \left.\max\left\{\underline{\boldsymbol{x}}\ \underline{\boldsymbol{y}}, \underline{\boldsymbol{x}}\ \overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}\ \underline{\boldsymbol{y}}, \overline{\boldsymbol{x}}\ \overline{\boldsymbol{y}}\right\}\right], \\
\frac{1}{\boldsymbol{x}} &\triangleq \begin{cases} [1/\overline{\boldsymbol{x}}, 1/\underline{\boldsymbol{x}}] & \text{if } \underline{\boldsymbol{x}} > 0 \\ [1/\underline{\boldsymbol{x}}, 1/\overline{\boldsymbol{x}}] & \text{if } \overline{\boldsymbol{x}} < 0 \end{cases} \quad (0 \notin [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]), \\
\boldsymbol{x} \div \boldsymbol{y} &\triangleq \boldsymbol{x} \times 1/\boldsymbol{y}.
\end{aligned}$$

The above definitions imply the ability to perform the four elementary operations with arbitrary precision. When implemented on a digital computer, however, truncation errors occur that may cause the resulting interval not to contain the result that would be obtained with ideal interval arithmetic. In order to avoid this effect, the lower endpoint of the interval must be rounded down to the nearest machine number less than the mathematically correct result, and the upper endpoint must be rounded up to the nearest machine number greater than the mathematically correct result. This mode of operation, called *direct rounding*, is available on any machine supporting the IEEE floating point standard.

Our use of IA is motivated by the need to obtain bounds on the range of mathematical functions.

*Definition 1 (Interval extension):* A function $\boldsymbol{F} : \mathbb{IR}^n \to \mathbb{IR}$ is said to be an *interval extension* of $F : \mathbb{R}^n \to \mathbb{R}$ provided

$$\mathbf{F}^u(\boldsymbol{X}) \subseteq \mathbf{F}(\boldsymbol{X})$$

for all intervals $\boldsymbol{X} \subset \mathbb{IR}^n$ within the domain of $\boldsymbol{F}$ [27].

The *natural* interval extension of a function is obtained by replacing variables with intervals and executing all operations according to the rule above. For instance, $\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{x}^2 - \boldsymbol{x}$, $\boldsymbol{f}_2(\boldsymbol{x}) = \boldsymbol{x}(\boldsymbol{x}-1)$, and $\boldsymbol{f}_3(\boldsymbol{x}) = (\boldsymbol{x}-1/2)^2 - 1/4$ are all interval extensions of $f(x) = x^2 - x = x(x-1) = (x-1/2)^2 - 1/4$. By setting $\boldsymbol{x} = [0,1]$ we have

$$\boldsymbol{f}_2(\boldsymbol{x}) = [0,1]\,([0,1]-1) = [0,1]\,[-1,0] = [-1,0]\,,$$

which necessarily includes the exact range $\boldsymbol{f}^u([0,1]) = [-1/4, 0]$.

However, the bounds provided by natural interval extensions are usually too wide or pessimistic to be of value. The following definition characterizes how sharply interval extensions enclose the range of a function.

*Definition 2 (Order $\alpha$ inclusion function):* Let $\boldsymbol{F}(\boldsymbol{X})$ be an interval extension of $F : \mathbb{R}^n \to \mathbb{R}$ evaluated over a box $\boldsymbol{X}$. We say that $\boldsymbol{F}$ is an *order $\alpha$ inclusion function* for $F$ if there is a constant $K$, independent of the box $\boldsymbol{X}$, such that

$$\mathrm{w}(\boldsymbol{F}(\boldsymbol{X})) - \mathrm{w}(\mathbf{F}^u(\boldsymbol{X})) \leq K\mathrm{w}(\boldsymbol{X})^\alpha \tag{14}$$

for all boxes $\boldsymbol{X}$ with $w(\boldsymbol{X})$ sufficiently small.

It can be shown [27] that natural interval extensions are first order. Higher-order inclusion functions are key to the design of efficient global optimization algorithms, as we shall see in the next section.

*Definition 3 (Interval Newton method):* Let $f : \boldsymbol{x} \subset \mathbb{R} \to \mathbb{R}$ be a function with continuous first derivative on $\boldsymbol{x}$ and let $x \in \boldsymbol{x}$. If $\boldsymbol{f}'(\boldsymbol{x})$ is any interval extension of the derivative of $f$ over $\boldsymbol{x}$, then the operator

$$\boldsymbol{N}(f; \boldsymbol{x}, x) \triangleq x - f(x)/\boldsymbol{f}'(\boldsymbol{x}) \tag{15}$$

is called the univariate interval Newton method.

It can be shown that if $\boldsymbol{N}(f; \boldsymbol{x}, x) \subset \boldsymbol{x}$, then there exist a unique solution of $f(x) = 0$ in $\boldsymbol{x}$. As any solution within $\boldsymbol{x}$ must also be within $\boldsymbol{N}(f; \boldsymbol{x}, x)$, the interval Newton method provides the following (quadratically convergent) iteration

$$\boldsymbol{x} \leftarrow \boldsymbol{N}(f; \boldsymbol{x}, x) \cap \boldsymbol{x}. \tag{16}$$

If, at a certain iteration, the intersection is empty, then the starting interval contains no solutions. If $0 \in \boldsymbol{f}'(\boldsymbol{x})$ the quotient in (15) is computed using the rule of extended interval division defined in [22]. The outcome will be, in general, the union of disjoint intervals. Multivariate interval Newton methods can be defined as well [27]. The method can be also generalized to handle continuous only (i.e, non-differentiable) functions [27].

### *B. IA based Global Optimization*

The ability of Interval Analysis to compute bounds to the range of functions has been most successful in global optimization. IA algorithms are usually based on branch-and-bound schemes [22], referred to as Moore-Skelboe or Hansen algorithm. The overall structure is:

1) store in a list $\mathcal{L}$ the initial box $\boldsymbol{X}_0 \in \mathbb{IR}^n$ containing the sought minima;
2) pick a box $\boldsymbol{X}$ from $\mathcal{L}$;
3) if $\boldsymbol{X}$ is guaranteed not to contain a global minimizer, then discard it, otherwise subdivide $\boldsymbol{X}$ and store the sub-boxes in $\mathcal{L}$;
4) repeat from step 2) until the width of the intervals in $\mathcal{L}$ are below the desired accuracy.

The criteria used to delete boxes are based on rigorous bounds, therefore the box containing the global minimizer is never deleted even in the presence of rounding errors.

We employed an algorithm inspired by a recently proposed global optimization method [28], based on the Moore-Skelboe-Hansen branch-and-bound algorithm and Bernstein polynomials for bounding the range of the objective function. A combination of several test have been used in our implementation.

The *cut-off* uses an upper bound $\hat{F}$ of the global minimum of the objective function $F$ to discards an interval $\boldsymbol{X}$ from $\mathcal{L}$ if $\boldsymbol{F}(\boldsymbol{X}) > \hat{F}$. Any value taken by $F$ is an upper bound for its global minimum, but the tighter is the bound, the more effective is the cut-off test. In Section IV-B.1 we describe the method that we used to determine and updated $\hat{F}$.

The *monotonicity* test determines whether the function $F$ is strictly monotone in an entire sub-box $\boldsymbol{X}$. Denote the interval extension of the gradient evaluated in $\boldsymbol{X}$ by $\boldsymbol{G}(\boldsymbol{X})$. If $0 \notin \boldsymbol{G}(\boldsymbol{X})$ then $\boldsymbol{X}$ can be deleted.

The *concavity* test examines the concavity of $F$, using its Hessian matrix $H$. Let $\boldsymbol{H}_{i,i}(\boldsymbol{X})$ denote the interval extension of the $i-$th diagonal entry of Hessian evaluated in $\boldsymbol{X}$. A box can be deleted if $\overline{\boldsymbol{H}}_{i,i}(\boldsymbol{X}) < 0$ for some $i$.

The *Interval Newton step* applies one step of the interval Newton method (16) to the non-linear system $\nabla F(X) = 0$, $X \in \boldsymbol{X}$. As a consequence we may validate that $\boldsymbol{X}$ contains no stationary points, in which case we discard $\boldsymbol{X}$, otherwise we may contract or subdivide $\boldsymbol{X}$. The complete optimization scheme can be summarized as the following pseudocode:

```
GLOBAL-OPTIMIZATION ALGORITHM
  U ← ∅
  L ← {X_0} list of boxes sorted in order of increasing F_(X)
  while L ≠ ∅ do
        remove the first box X from L
        if w(X) < EPS then U ← U ∪ {X}
        else if (cut-off test: F(X) > F̂   or
                 monotonicity test: 0 ∉ G(X)   or
                 concavity test: H̄_{i,i}(X) < 0 for some i)   then Y ← ∅
             else  interval Newton step: Y ← X ∩ N(∇F, X, m(X))
        bisect Y and insert the resulting boxes in L
        update F̂
  end
  return U
```

A problem of global optimization algorithms based on IA is the so called *cluster effect*: as observed in [29], sub-boxes containing no solutions cannot be easily eliminated if there is a local minimum nearby. As a consequence of over-estimation in range bounding, many small boxes are created by repeated splitting, whose processing may dominate the total work spent on global search. This phenomenon occurs when the order of the inclusion function is less than three [29], hence we shall look for sharper inclusion functions.

*1) Taylor-Bernstein forms:* An interesting extension of IA that reduces the over-estimation is based on Taylor polynomials.

*Definition 4 (Taylor Model):* Let $F : \boldsymbol{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ be a function that is $(m+1)$ times continuously partially differentiable. Let $X_0$ be a point in $\boldsymbol{X}$ and $P_{m,F}$ the the $m$-th order Taylor polynomial of $F$ around $X_0$. Let $\boldsymbol{I}_{m,F}$ be an interval such that

$$F(X) \in P_{m,F}(X - X_0) + \boldsymbol{I}_{m,F} \quad \forall X \in \boldsymbol{X}. \tag{17}$$

We call the pair $(P_{m,F}, \boldsymbol{I}_{m,F})$ an $m$-th order *Taylor model* of $F$ [30] .
Hence $P_{m,F} + \boldsymbol{I}_{m,F}$ encloses $F$ between two hypersurfaces on $\boldsymbol{X}$ (Fig. 1).

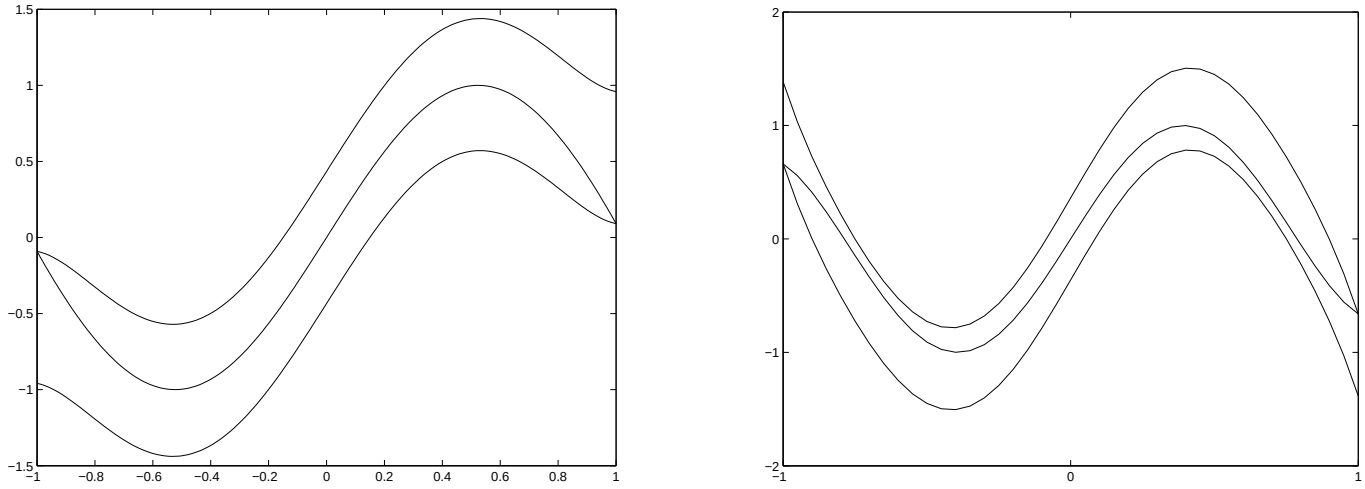


Fig. 1. Example of bounding a 7th order polynomial with a 3rd order Taylor model

Taylor models of any computable function can be obtained recursively using the *Taylor Model Arithmetic* described in [30]. In order to bound the range of a function $F$ over a domain $\boldsymbol{X}$, it is sufficient to compute an interval extension $\boldsymbol{P}_{m,F}(\boldsymbol{X})$ for the polynomial $P_{m,F}$, since from Definition 4 it follows that

$$\mathbf{F}^u(\boldsymbol{X}) \subseteq \boldsymbol{P}_{m,F}(\boldsymbol{X}) + \boldsymbol{I}_{m,F}.$$

The sharpness of the bounds depends on the method used to obtain the inclusion function for $P_{m,F}$. More precisely, if $\boldsymbol{P}^u_{m,F}(\boldsymbol{X})$ is the exact range of $P_{m,F}$, then $\boldsymbol{P}^u_{m,F}(\boldsymbol{X}) + \boldsymbol{I}_{m,F}$ is an $m+1$ order inclusion function for $F$ over $\boldsymbol{X}$, where $m$ is the degree of the Taylor polynomial [28].

A *Taylor-Bernstein form* is a Taylor model where the polynomial is expressed in the Bernstein basis rather than in the canonical power basis. The advantage is that the Taylor-Bernstein form allows to compute the exact range of the polynomial part. Hence, with $n \geq 2$, the cluster effect is avoided. A Bernstein polynomial has the form (in one dimension):

$$p(x) = \sum_{i=1}^{n} a_i \begin{pmatrix} n \\ i \end{pmatrix} x^i (1-x)^{n-i}. \tag{18}$$

An important property of these polynomials is that $p(x)$ on $\boldsymbol{x}$ is a convex combination of $a_i$'s, so that the coefficients of the Bernstein form provide lower and upper bounds to the range:

$$\boldsymbol{p}^u(\boldsymbol{x}) \subseteq [\min\{a_i\}, \max\{a_i\}].$$

If the polynomial is monotone over a domain $\boldsymbol{x}$ then the Bernstein form gives the exact range since the minimum and maximum occurs respectively at $a_1$ and $a_n$, $a_1 = p(\underline{\boldsymbol{x}})$ and $a_n = p(\overline{\boldsymbol{x}})$. This suggests that the exact range of a polynomial $p$ on $\boldsymbol{x}$ can be obtained by transforming the polynomial into Bernstein form and then repeatedly subdividing it the until the bounds of all sub-boxes are exact. The subdivision can be easily done with De Casteljau algorithm, well known in Computer Graphics [31]. Bernstein polynomials can be easily extended to the multivariate case, where analogous properties hold.

The knowledge of the exact range of $P_{m,F}$ helps to make the cut-off test more effective. Indeed, if $\boldsymbol{P}^u_{m,F}(\boldsymbol{X})$ is the exact range, then $\underline{\boldsymbol{P}}^u_{m,F}(\boldsymbol{X}) = \min\{P_{m,F}\}$ and the minimum of $F$ over $\boldsymbol{X}$ is contained in $\underline{\boldsymbol{P}}^u_{m,F}(\boldsymbol{X}) + \boldsymbol{I}_{m,F}$. Then $\underline{\boldsymbol{P}}^u_{m,F}(\boldsymbol{X}) + \overline{\boldsymbol{I}}_{m,F}$ is an upper bound of the minimum of $F$ over $\boldsymbol{X}$. The cut-off value $\hat{F}$ is the smallest upper bound for all the boxes in the list.

The advantages and limits of Taylor models are widely discussed in [32], where the author also points out that the Taylor-Bernstein form is well suited to low dimension problems.

As seen above, the minimization algorithm makes use of the Jacobian and Hessian matrix of the cost function, that we derive in closed form in Appendix I.

## V. Experimental results

The algorithm was tested on both synthetic and real data.

### A. Synthetic data

Synthetic data consisted of 50 points randomly scattered in a sphere of unit radius, centered at the origin. Views were generated by placing cameras at random positions, at a mean distance from the center of 2.5 units with a standard deviation of 0.25. The orientations of the cameras were chosen randomly with the constraint that the optical axis should point toward the center. The intrinsic parameters were given a known value: $\alpha_u = \alpha_v = 800, u_0 = v_0 = 256$. As customary it was assumed $\gamma = 0$. Image points were (roughly) contained in a $512 \times 512$ image. Fundamental matrices were computed using the linear 8-point algorithm with data normalization as described by Hartley in [33]. The weight $w_{ij}$ has been defined as the residual of the estimation of $F_{ij}$ [12]. We used Taylor models of degree four. The required accuracy was $10^{-10}$; using this value we typically get a box width of 2.5 pixels.

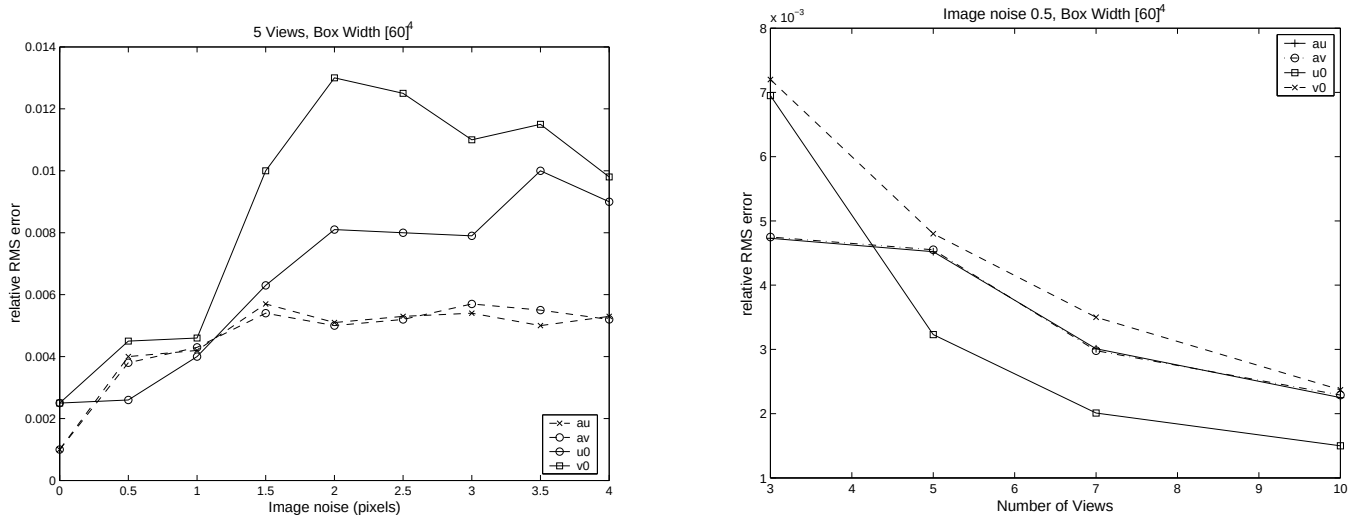


Fig. 2. Relative RMS error on intrinsic parameters versus image noise standard deviation (left) and number of views (right).

In order to assess the accuracy of the method, Gaussian noise with variable standard deviation was added to image points and the number of views was varied as well (the minimum number of views required to achieve autocalibration is three, according to (9)). The algorithm was started with the box

$[0, 60]^4$ centered on the true solution. Since the fundamental matrices are affected by image noise, the minimizer of the cost function does not coincide with the actual intrinsic parameters. The relative RMS error is reported in Fig. 2. Each point is the average of 50 independent trials.

| Number of views | 3 | 5 | 7 | 10 |
|---|---|---|---|---|
| Time [min] | 1.2 | 6.6 | 8.0 | 17.1 |

TABLE I

COMPUTATION TIMES VERSUS NUMBER OF VIEWS. THE INITIAL BOX WAS $[0, 60]^4$.

Computation times were recorded for varying number of views, initial box width and number of unknowns. Table I reports computation times versus number of views. Table II reports computation times versus box width (5 views) starting from a reference box of $[300, 1700] \times [300, 1700] \times [156, 356] \times [156, 356]$. In the first column all the four parameters were considered unknown, in the second one only focal lengths were unknown, whereas the principal point was set at (256, 256). These figures refers to our implementation in MATLAB and C++, on a Pentium III 900 MHz processor.

| Box Width | Time [min] (4 unknowns) | Time [min] (2 unknowns) |
|---|---|---|
| Ref. | 23.2 | 9.1 |
| -10% | 16.6 | 8.8 |
| -20% | 15.3 | 7.2 |
| +10% | 28.3 | 11.9 |
| +20% | 32.0 | 14.8 |

TABLE II

COMPUTATION TIMES VERSUS INITIAL BOX WIDTH. IN THE RIGHTMOST COLUMN THE PRINCIPAL POINT WAS KNOWN.

In order to compare our minimization with a standard gradient method, we used the quasi-Newton method implemented by the `fminunc` function in the MATLAB Optimization Toolbox. The algorithm was initialized by randomly choosing a point in the domain $[300, 1700] \times [300, 1700] \times [156, 356] \times [156, 356]$. After performing 100 trials we recorded how many times the algorithm converged to the correct solution, which was assumed to be the one to which it converged when initialized with the true intrinsic parameters (within a 10% tolerance). The quasi-Newton method converged in the 86% of cases, with 5 views and 1.0 pixel noise. Average running time was 0.9 sec.

### *B. Real data*

We tested our algorithm on the same real sequences used in other works on autocalibration [16], [34], [4], [35]. The starting box was chosen as follows: the midpoint for $(u_0, v_0)$ is the image center and the width is 20% of the image size; the interval for the focal lengths is always $[300 \times 1700]$. Point correspondences were obtained manually. Table III compares our results with those previously published, when available. Please note that the values reported by other articles are the result of different autocalibration algorithms, and must not be taken as ground truth.

In order to obtain a more meaningful evaluation we compared the results of our algorithm with those obtained by a standard calibration technique [36]. Table IV reports the result for three sequences taken in Verona with the same photocamera.

In all the real experiments the final interval width was around one pixel. Computation times for the real sequences are shown in Table V. All the sequences consists of five frames.

### *C. 3-D Reconstruction*

Using the midpoint of intrinsic parameters computed by autocalibration, and the fundamental matrices, structure was recovered by first factorizing out the motion from the essential matrices [13], then recovering

| | Our algorithm | | | | Previous results | | | |
|---|---|---|---|---|---|---|---|---|
| Sequence | $\alpha_u$ | $\alpha_v$ | $u_0$ | $v_0$ | $\alpha_u$ | $\alpha_v$ | $u_0$ | $v_0$ |
| Valbonne [34] | 619 | 699 | 234 | 372 | 681 | 679 | 259 | 383 |
| ETL [35] | 800 | 831 | 405 | 352 | 837 | 837 | (378) | (252) |
| Nekt [4] | 720 | 600 | 410 | 191 | 713 | 605 | 378 | 314 |

TABLE III

MIDPOINTS OF INTRINSIC PARAMETERS COMPUTED WITH OUR ALGORITHM VERSUS PREVIOUS RESULTS. VALUES IN BRACKETS WERE GUESSED, NOT COMPUTED.

| Sequence | $\alpha_u$ | $\alpha_v$ | $u_0$ | $v_0$ |
|---|---|---|---|---|
| *Calibration* | *1463* | *1459* | *636* | *498* |
| Castel Vecchio | 1486 | 1446 | 699 | 432 |
| Scala Ragione | 1439 | 1482 | 605 | 477 |
| Piazza Dante | 1436 | 1407 | 612 | 561 |

TABLE IV

MIDPOINT OF INTRINSIC PARAMETERS COMPUTED WITH OUR ALGORITHM VERSUS CALIBRATION.

the projection matrices [34] and finally computing 3-D structure by triangulation [37]. As customary, results are refined by bundle adjustment, in order to obtain a maximum likelihood solution with respect to the underlying measures. More details can be found in Appendix II.

As shown in Fig. 3 and 4, the projection of the reconstructed points coincides with the original image points.

In order to assess quantitatively the metric accuracy of the reconstruction, as the absolute dimensions of the objects are unknown, we computed the angles between 3D segments that are known to be parallel or orthogonal in the real scene (Table VI). This segments are a subset of those shown in Fig. 3 and 4.

Test sequences and more results can be found on the World Wide Web at
`http://www.sci.univr.it/~fusiello/demo/autocal`.

## VI. CONCLUSIONS AND FUTURE WORK

Global optimization based on Interval Analysis has been applied to the autocalibration problem, obtaining a technique that is guaranteed to converge to the global solution with mathematical certainty and arbitrary accuracy. The result shows that our implementation is correct and achieves the global minimum in a reasonable time. The choice of the initial box is not critical for the successful termination of the algorithm – provided that it contains the global minimizer – because it only influences the computation time.

The accuracy of the method is in agreement with the figures reported in [12], [16], as we use basically the same cost function.

Future work will aim at reducing computation time by testing several variations to the present model.

| Sequences | Time [min] |
|---|---|
| Valbonne | 77 |
| ETL | 97 |
| Nekt | 65 |
| Scala Ragione | 50 |
| Castel Vecchio | 27 |
| Piazza Dante | 46 |

TABLE V

COMPUTATION TIMES FOR REAL SEQUENCES.

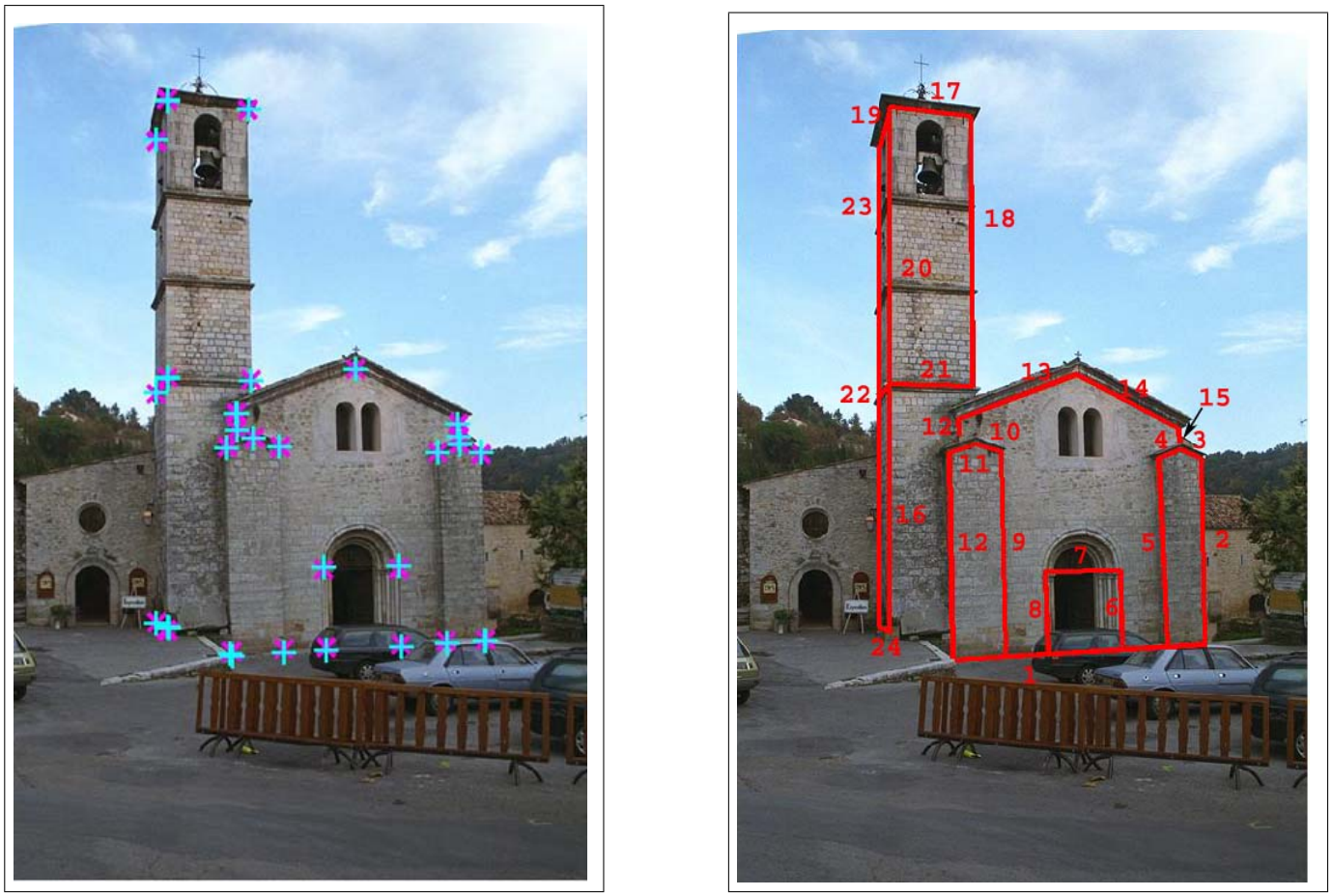


Fig. 3. Valbonne church. Left:the projection of reconstructed points (cyan $\times$ mark) are shown superimposed onto the original feature points (magenta $+$ mark). Right: segments joining selected features.

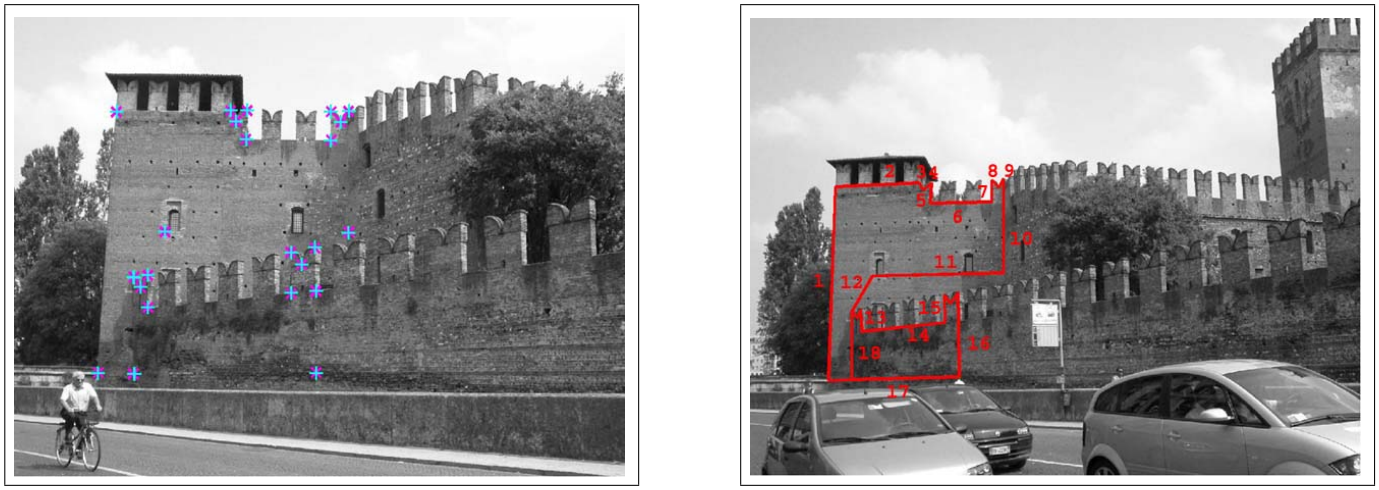

Fig. 4. Castel Vecchio. Left: the projection of reconstructed points (cyan $\times$ mark) are shown superimposed onto the original feature points (magenta $+$ mark). Right: segments joining selected features.

| Segments | Computed | True |
|---|---|---|
| 21-22 | $90.73^\circ$ | $90^\circ$ |
| 20-22 | $1.52^\circ$ | $0^\circ$ |
| 20-18 | $0.82^\circ$ | $0^\circ$ |
| 21-17 | $2.97^\circ$ | $0^\circ$ |
| 19-22 | $0.79^\circ$ | $0^\circ$ |
| 17-19 | $88.59^\circ$ | $90^\circ$ |
| 20-22 | $89^\circ$ | $90^\circ$ |
| 21-18 | $88.46^\circ$ | $90^\circ$ |

| Segments | Computed | True |
|---|---|---|
| 17-1 | $87.74^\circ$ | $90^\circ$ |
| 1-2 | $89.61^\circ$ | $90^\circ$ |
| 5-6 | $89.16^\circ$ | $90^\circ$ |
| 6-7 | $88.45^\circ$ | $90^\circ$ |
| 7-10 | $1.7^\circ$ | $0^\circ$ |
| 15-16 | $2.01^\circ$ | $0^\circ$ |
| 14-15 | $89.52^\circ$ | $90^\circ$ |
| 14-17 | $87.59^\circ$ | $90^\circ$ |
| 13-14 | $87.32^\circ$ | $90^\circ$ |

TABLE VI

ANGLES BETWEEN SELECTED SEGMENTS OF VALBONNE CHURCH (LEFT) AND CASTEL VECCHIO (RIGHT).

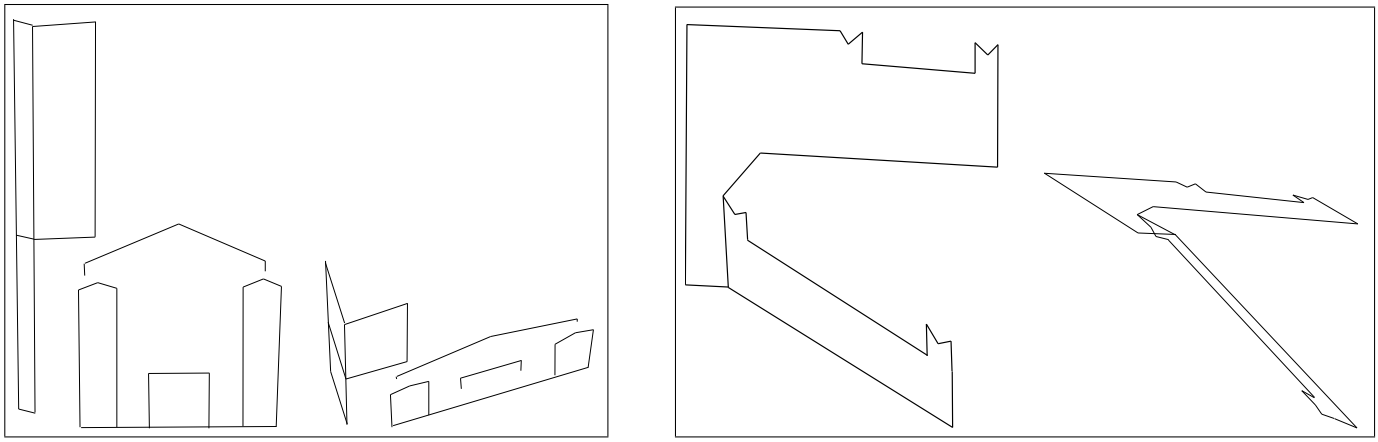

Fig. 5. Views of the 3D reconstruction of Valbonne church (left) and Castel Vecchio (right). To improve readability, only selected points joined by segments are shown.

We also plan to explore the use IA tools to automatically detect degenerate configurations, which are known to afflict autocalibration [38], [17].

## APPENDIX I
## DERIVATIVES OF THE COST FUNCTION

Instead of reverting to subscript notation for computing the derivatives of the cost function, we perform the entire operation using matrix algebra, and in particular the elegant *matrix differential calculus* introduced by Magnus and Neudecker [39].

### *A. Notation and useful results*

We will now introduce a notation and some related results on matrix differential calculus, which are used to compute the Jacobian and the Hessian matrix of (13).

*Definition 5:* Let $F$ be a differentiable $p \times q$ real matrix function of a $n \times m$ matrix of real variables $X$. The Jacobian matrix of $F$ at $X$ is the $pq \times nm$ matrix

$$\mathcal{D}\, F(X) \triangleq \frac{\partial \operatorname{vec} F(X)}{\partial (\operatorname{vec} X)^{\mathsf{T}}}. \tag{19}$$

where $\operatorname{vec}(A)$ is the $nm \times 1$ vector obtained by stacking the columns of $A$. The $\operatorname{vec}$ operator has some interesting properties in connection with the *Kronecker product* $\otimes$. It is possible to prove that:

$$\operatorname{vec} ABC = (C^{\mathsf{T}} \otimes A) \operatorname{vec} B. \tag{20}$$

*Theorem 2:* Let $F$ be a $p \times q$ matrix function of an $n \times m$ matrix $X$, and $G(Y)$ be a $r \times s$ matrix function of an $p \times q$ matrix $Y$. If the composite function $H(X) \triangleq G(F(X))$ is differentiable at $X_0$, its Jacobian matrix is

$$\mathcal{D}\, H(X_0) = (\mathcal{D}\, G(Y_0))(\mathcal{D}\, F(X_0)). \tag{21}$$

where $Y_0 \triangleq F(X_0)$.

*Theorem 3:* The identification theorem of Jacobian matrices states that: $\mathrm{d}\,\mathrm{vec}\, F(X) = A(X)\,\mathrm{d}\,\mathrm{vec}\, X$ is equivalent to $\mathcal{D}\, F(X) = A(X)$.

This theorem is extremely useful in the applications of matrix differential calculus since it transforms the problem of finding the Jacobian matrix of a matrix function into the problem of finding its differential, which is generally easier.

*Definition 6:* The Hessian matrix of a $p \times q$ matrix function $F$ of a $n \times m$ matrix variable $X$ is a block-partitioned matrix formed by $pq$ symmetric matrices of order $nm$ stacked vertically, defined by

$$\mathcal{H}\, F(X) \triangleq \mathcal{D}(\mathcal{D}\, F(X))^{\mathsf{T}}. \tag{22}$$

*Theorem 4:* Let $F$ be a $p \times q$ matrix function of an $n \times m$ matrix $X$, and $G(Y)$ be a $r \times s$ matrix function of an $p \times q$ matrix $Y$. If the composite function $H(X) \triangleq G(F(X))$ is twice differentiable at $X_0$, its Hessian matrix is

$$\mathcal{H}\, H(X_0) = (I_{r\times s} \otimes \mathcal{D}\, F(X_0))^{\mathsf{T}}\, \mathcal{H}\, G(Y_0)\, \mathcal{D}\, F(X_0) + (\mathcal{D}\, G(Y_0) \otimes I_{n\times m})\, \mathcal{H}\, F(X_0), \tag{23}$$

where $Y_0 \triangleq F(X_0)$.

*1) Miscellaneous properties:*

- Given a $n \times m$ matrix $A$, the commutation matrix $K_{nm}$ is the permutation matrix that transforms $\mathrm{vec}\, A$ into $\mathrm{vec}\, A^{\mathsf{T}}$:

$$\mathrm{vec}\, A^{\mathsf{T}} = K_{n,m}\, \mathrm{vec}\, A. \tag{24}$$

For square matrices we use the notation $K_n$ instead of $K_{n,n}$.

- The Jacobian matrix of the function

$$F(X) = XX^{\mathsf{T}},$$

where $X$ is a $n \times n$ matrix, is

$$\mathcal{D}\, F(X) = 2N_n(X \otimes I_n), \tag{25}$$

where $N_n \triangleq \frac{1}{2}(I_{n^2} + K_n)$. The Hessian matrix is

$$\mathcal{H}\, F(X) = (K_{n^2,n} \otimes I_n)(I_n \otimes (2N_n \otimes I_n)(I_n \otimes \mathrm{vec}\, I_n)) \triangleq Q_n. \tag{26}$$

- The Jacobian matrix of the scalar function

$$\phi(X) = \mathrm{tr}(XX^{\mathsf{T}})$$

is

$$\mathcal{D}\, \phi(X) = \mathrm{vec}^{\mathsf{T}}(2X). \tag{27}$$

and its Hessian matrix is

$$\mathcal{H}\, \phi(X) = 2I_{n^2}. \tag{28}$$

### B. Jacobian matrix of the cost function

Let us rewrite the cost function (13) as the composite function

$$\chi(A) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} \psi(S(E_{ij}(A))), \tag{29}$$

where

$$\psi : \mathcal{M}^{2\times 1} \to \mathbb{R},\ \psi(C) = \left(\tfrac{c_2}{c_1} - 1\right),\ C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{30}$$

$$S : \mathcal{M}^{3\times 3} \to \mathcal{M}^{2\times 1},\ S(B) = \begin{bmatrix} \operatorname{tr}^2\left(BB^{\mathsf{T}}\right) \\ 2\operatorname{tr}((BB^{\mathsf{T}})^2) \end{bmatrix} \tag{31}$$

$$E_{ij} : \mathcal{M}^{3\times 3} \to \mathcal{M}^{3\times 3},\ E_{ij}(A) = A^{\mathsf{T}} F_{ij} A \tag{32}$$

and $\mathcal{M}^{n\times m}$ denotes the set of real $n \times m$ matrices. Since the derivative is a linear operator, we will concentrate on the expression of the Jacobian matrix of a single term of the cost function. ¿From now on, the $ij$ superscripts will be omitted for simplicity.

By applying twice Theorem 2 to the composite function ($\psi \circ S \circ E$), we obtain

$$\mathcal{D}\,\psi(A) = \mathcal{D}\,\psi(C)\,\mathcal{D}\,S(B)\,\mathcal{D}\,E(A), \tag{33}$$

where $B \triangleq E(A)$ and $C \triangleq S(B)$.

Let us concentrate on $\mathcal{D}\,E(A)$. Using (8),

$$\mathrm{d}\,E(A) = \mathrm{d}\,A^{\mathsf{T}}(FA) + A^{\mathsf{T}} F\,\mathrm{d}\,A. \tag{34}$$

For the linearity of the $\mathrm{vec}$ operator,

$$\mathrm{d}\,\mathrm{vec}\,E(A) = \mathrm{vec}(\mathrm{d}\,A^{\mathsf{T}}(FA)) + \mathrm{vec}(A^{\mathsf{T}} F\,\mathrm{d}\,A). \tag{35}$$

Using (20), (24) and Theorem 3, we obtain

$$\mathcal{D}\,E(A) = ((A^{\mathsf{T}}F^{\mathsf{T}}) \otimes I_3)K_3 + (I_3 \otimes (A^{\mathsf{T}}F)). \tag{36}$$

We now proceed to compute $\mathcal{D}\,S(B)$. Using Theorem 2 and (25),(27) we can write

$$\mathcal{D}\,S(B) = \begin{bmatrix} 4\operatorname{tr}\left(BB^{\mathsf{T}}\right)\mathrm{vec}^{\mathsf{T}}(B) \\ 8\,\mathrm{vec}^{\mathsf{T}}(BB^{\mathsf{T}})N_n(B \otimes I_n). \end{bmatrix} \tag{37}$$

Finally,

$$\mathcal{D}\,\psi(C) = \begin{bmatrix} -\frac{c_2}{c_1^2} & \frac{1}{c_1} \end{bmatrix}. \tag{38}$$

### C. Hessian matrix of the cost function

By applying twice Theorem 4 to the composite function ($\psi \circ S \circ E$) we get,

$$\begin{aligned} \mathcal{H}\,\psi(A) = {} & (\mathcal{D}\,S(B)\,\mathcal{D}\,E(A))^{\mathsf{T}}\,\mathcal{H}\,\psi(C)\,\mathcal{D}\,S(B)\,\mathcal{D}\,E(A) \\ & + (\mathcal{D}\,\psi(C) \otimes I_9)(I_2 \otimes \mathcal{D}\,E(A))^{\mathsf{T}}\,\mathcal{H}\,S(B)\,\mathcal{D}\,E(A) + (\mathcal{D}\,S(B) \otimes I_9)\,\mathcal{H}\,E(A). \end{aligned} \tag{39}$$

All the Jacobian matrices occurring in (39) have been already computed in Sec. I-B. In the rest of this section we will find expressions for the three Hessian matrices $\mathcal{H}\,\psi(C)$, $\mathcal{H}\,S(B)$ and $\mathcal{H}\,E(A)$.

$$\mathcal{H}\,E(A) = \mathcal{D}(\mathcal{D}\,E(A))^{\mathsf{T}}. \tag{40}$$

After some rewriting, using Theorem 3 we get

$$\mathcal{H}\,E(A) = (I_9 \otimes K_3^{\mathsf{T}})(I_3 \otimes K_3 \otimes I_3)(I_9 \otimes \operatorname{vec} I_3)(I_3 \otimes F) + (I_3 \otimes K_3 \otimes I_3)(\operatorname{vec} I_3 \otimes I_9)(I_3 \otimes F^{\mathsf{T}}). \quad (41)$$

We now proceed to find an expression for $\mathcal{H}\,S(B)$:

$$\mathcal{H}\,S(B) = \begin{bmatrix} 8 \operatorname{vec} E \operatorname{vec}^{\mathsf{T}} E + 4 \operatorname{tr}(EE^{\mathsf{T}}) I_9 \\ 16(E^{\mathsf{T}} \otimes I_3) N_3 (E \otimes I_3) + 4(\operatorname{vec}^{\mathsf{T}}(EE^{\mathsf{T}}) \otimes I_9) Q_3 \end{bmatrix} \quad (42)$$

where $N_n$ and $Q_n$ are defined in Sec. I-A.1. Finally, we can write:

$$\mathcal{H}\,\psi(C) = \begin{bmatrix} 2\frac{c_2}{c_1^3} & -\frac{1}{c_1^2} \\ -\frac{1}{c_1^2} & 0 \end{bmatrix}. \quad (43)$$

## APPENDIX II
## COMPUTING CAMERA MOTION AND STRUCTURE

In this section we will briefly describe the algorithms used in the experiments to recover the motion of the camera and the scene structure, after autocalibration. They are quite standard algorithms, therefore we shall omit many details and demonstrations and go straight to the final formulae.

### *A. Camera motion*

If the essential matrix can be computed (which entails the knowledge of the intrinsic parameters), a theorem due to Hartley [13] allow to factorize it like in (5), therefore to recover camera motion.

*Theorem 5:* Let $E$ be $3 \times 3$ matrix satisfying the hypotheses of Theorem 1, and let $E = UDV^{\mathsf{T}}$ be its Singular Value Decomposition, where $D = \operatorname{diag}(\sigma, \sigma, 0)$. Then, up to a scale factor, the factorization of Theorem 1 is one of the following four:

$$S = U(\pm Z_1)U^{\mathsf{T}} \quad (44)$$

$$R = \det(UV^{\mathsf{T}})UZ_2V^{\mathsf{T}} \text{ or} \quad (45)$$

$$R = \det(UV^{\mathsf{T}})UZ_2^{\mathsf{T}}V^{\mathsf{T}}, \quad (46)$$

where

$$Z_1 \triangleq \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z_2 \triangleq \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (47)$$

The choice between the four displacements is determined by the requirement that the visible points be in front of *both* cameras, i.e., the depth of each point, with respect to both cameras must be positive.

### *B. Reconstruction*

In the autocalibration scenario no metric information about the scene is available. Therefore, we can compute the displacement between two cameras only up to an unknown scale factor. The projection matrices for the first two views are (the choice of the first camera fixes the world reference frame):

$$P^1 = A[I \mid 0] \qquad P^2 = A[R_{12} \mid t_{12}]. \quad (48)$$

where $R_{12}$ is the rotation and $t_{12}$ is the direction of translation, recovered from the factorization of $E_{12}$. While in the case of two views it is appropriate to use the direction of translation, since this results in a global scene scale factor, if we want to perform reconstruction from three views (or more), we have to obtain three coherent projection matrices. Using the direction of translation in computing $P^3$ (like we

did for $P^2$) would yield an incorrect result, in which the epipolar constraint between view 1 and 3 is not satisfied. As pointed out in [34], the correct projection matrix $P^i$ for any view $i > 2$ is

$$P^i = A[R_{1i} \mid \mu_{1i} t_{1i}]. \tag{49}$$

with

$$\mu_{1i} \triangleq \frac{||(R_{2i} t_{12} \times t_{2i})||^2}{(t_{1i} \times t_{2i})^{\mathsf{T}} (R_{2i} t_{12} \times t_{2i})}. \tag{50}$$

In this way, there is only one global scale factor left, corresponding to the unknown norm of the translation 1-2.

Three-dimensional (3D) structure is recovered from projection matrices and point correspondences by *triangulation*. Each image point defines a ray in space. The intersection of rays of corresponding points gives the position of the 3D point. In the presence of noise these rays cannot be guaranteed to intersect, and a number of solutions have been proposed (see [40] for a review). A commonly suggested method [37] is to compute the 3D point which minimizes the sum of the square distances of the 3D point to each ray. In the case of two views, this is the mid-point of the common perpendicular to the two rays.

Consider a 3D point $w$ which projects into $m_i$ in view $i = 1 \dots n$; its 3D position is given by [37]

$$w = \left( \sum_{i=1}^{n} (I - d_i d_i^{\mathsf{T}}) \right)^{-1} \left( \sum_{i=1}^{n} (c_i - (c_i^{\mathsf{T}} d_i) d_i \right), \tag{51}$$

where $c_i$ is the optical center of camera $i$ and $d_i = (Q_i^{-1} m_i)/||Q_i^{-1} m_i||$. This is called the ***mid-point*** method.

Finally, the projection matrices and the structure computed in this way are refined using a technique similar to the *bundle adjustment*, as suggested by [34].

Consider a set of three-dimensional points viewed by N cameras with matrices $\{P^i\}_{i=1 \dots N}$. Let $m_j^i \backsim P^i w_j$ be the (homogeneous) coordinates of the projection of the $j$-th point onto the $i$-th camera. Given the set of pixel coordinates $\{m_j^i\}$, find the set of camera matrices $\{P^i\}$ and the scene structure $\{w_j\}$ such that

$$m_j^i \backsim P^i w_j. \tag{52}$$

This is done by minimizing the image reprojection error:

$$\sum_{j=1}^{n} \sum_{i \in I_j} \left( \left( u_j^i - \frac{q_1^i w_j}{q_3^i w_j} \right)^2 + \left( v_j^i - \frac{q_1^i w_j}{q_3^i w_j} \right)^2 \right) \tag{53}$$

where $I_j$ is the set of camera indices where the point $w_j$ is visible,

$$P^i \triangleq \begin{bmatrix} q_1^i \\ q_2^i \\ q_3^i \end{bmatrix} \quad \text{and} \quad m_i^j \triangleq \begin{bmatrix} u_i^j \\ v_i^j \\ 1 \end{bmatrix}.$$

The cost function (53) is minimized over the set of projection matrices $\{P^i\}$, using the MATLAB implementation of the Levenberg-Marquardt algorithm. At each evaluation the point coordinates $\{w_j\}$ are updated by triangulation. The unknowns are the five intrinsic parameters, a rotation and a translation for each camera. Rotations are represented by unit quaternions, normalized at each evaluation.

## ACKNOWLEDGMENTS

The authors are grateful to R. P. Mendonça for introducing them to matrix differential calculus. A special thanks goes to Arnold Neumaier for many helpful discussion concerning the rigorous implementation of Taylor models. G. Roth kindly provided the image sequences he used in his paper [16]. This work is partially supported by the NSF Engineering Research Center for Neuromorphic Systems Engineering (CNSE) at Caltech (EEC-9402726).

## REFERENCES


[1] T. S. Huang and A. N. Netravali, "Motion and structure from feature correspondences: A review," *Proceedings of IEEE*, vol. 82, no. 2, pp. 252–267, 1994.

[2] S. J. Maybank and O. Faugeras, "A theory of self-calibration of a moving camera," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 123–151, 1992.

[3] Q.-T. Luong and O. Faugeras, "Self-calibration of a moving camera from point correspondences and fundamental matrices," *International Journal of Computer Vision*, vol. 22, no. 3, pp. 261–289, 1997.

[4] M. I. Lourakis and R. Deriche, "Camera self-calibration using the singular value decomposition of the fundamental matrix," in *Proc. of the 4th Asian Conference on Computer Vision*, vol. I, January 2000, pp. 403–408.

[5] R. I. Hartley, "Kruppa's equations derived from the fundamental matrix," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 133–135, February 1997.

[6] M. Pollefeys and L. Van Gool, "A stratified approach to metric self-calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997, pp. 407–412.

[7] A. Heyden and K. Åström, "Euclidean reconstruction from constant intrinsic parameters," in *Proceedings of the International Conference on Pattern Recognition*, Vienna, 1996, pp. 339–343.

[8] B. Triggs, "Autocalibration and the absolute quadric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997, pp. 609–614.

[9] A. Fusiello, "Uncalibrated Euclidean reconstruction: A review," *Image and Vision Computing*, vol. 18, no. 6-7, pp. 555–563, May 2000.

[10] A. Heyden and K. Åström, "Euclidean reconstruction from image sequences with varying and unknown focal length and principal point," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997, pp. 438–443.

[11] M. Pollefeys, R. Koch, and L. Van Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, 1998, pp. 90–95.

[12] P. Mendonça and R. Cipolla, "A simple techinique for self-calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. I:500–505.

[13] R. I. Hartley, "Estimation of relative camera position for uncalibrated cameras," in *Proceedings of the European Conference on Computer Vision*, Santa Margherita L., 1992, pp. 579–587.

[14] P. Sturm, "On focal length calibration fro two views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. II, Kauai, USA, 2001, pp. 145–150.

[15] S. Bougnoux, "From projective to Euclidean space under any practical situation, a criticism of self-calibration," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, 1998, pp. 790–796.

[16] G. Roth and A. Whitehead, "Some improvements on two autcalibration algorithms based on the fundamental matrix," in *Proceedings of the International Conference on Pattern Recognition*, vol. 2, Quebec City, Alberta, 2002, pp. 312–315.

[17] J. Ponce, "On computing metric upgrades of projective reconstructions under the rectangular pixel assumption," in *Proc. of the SMILE 2000 Workshop on 3D Structure from Multiple Images of Large-Scale Environments,*, ser. LNCS, M. Pollefeys, L. V. Gool, A. Zisserman, and A. Fitzgibbon, Eds., no. 2018. Dublin, Ireland: Springer-Verlag, 2000, pp. 52–67.

[18] J. Oliensis, "Fast and accurate self-calibration," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[19] A. Heyden and K. Åström, "Minimal conditions on intrinsic parameters for Euclidean reconstruction," in *Proceedings of the Asian Conference on Computer Vision*, Hong Kong, 1998.

[20] R. Hartley, E. Hayman, L. de Agapito, and I. Reid, "Camera calibration and the search for infinity," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[21] A. Neumaier, *Introduction to Numerical Analysis*. Cambridge: Cambridge Univ. Press, 2001.

[22] E. R. Hansen, *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[23] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[24] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, pp. 43–75, 1996.

[25] T. Huang and O. Faugeras, "Some properties of the E matrix in two-view motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 12, pp. 1310–1312, December 1989.

[26] R. E. Moore, *Interval Analysis*. Prentice-Hall, 1966.

[27] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*. Kluwer, 1996.

[28] P.S.V.Nataraj and K.Kotecha, "An algorithm for global optimization using the taylor-bernstein form as an inclusion function," *International Journal of Global Optimization*, vol. 24, pp. 417–436, 2002.

[29] B. Kearfott and Du, "The cluster problem in multivariate global optimization," *Journal of Global Optimization*, vol. 5, pp. 253–365, 1994.

[30] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *International Journal of Pure and Applied Mathematics*, vol. 4, no. 4, pp. 379–456, 2003.

[31] D. Rogers and J.A.Adams, *Mathematical Elements for Computer Graphics*, 2nd ed. McGraw-Hill, 1990.

[32] A. Neumaier, "Taylor forms - use and limits," *Reliable Computing*, vol. 9, pp. 43 – 79, 2002.

[33] R. I. Hartley, "In defence of the 8-point algorithm," in *Proceedings of the IEEE International Conference on Computer Vision*, 1995.

[34] C. Zeller and O. Faugeras, "Camera self-calibration from video sequences: the Kruppa equations revisited," INRIA, Research Report 2793, February 1996.

[35] T. Ueshiba and F. Tomita, "A factorization method for projective and Euclidean reconstruction from multiple perspective views via iterative depth estimation," in *Proceedings of the European Conference on Computer Vision*, University of Freiburg, Germany, 1998, pp. 296–310.

[36] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[37] P. Beardsley, A. Zisserman, and D. Murray, "Sequential update of projective and affine structure from motion," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 235–259, 1997.

[38] P. Sturm, "A case against Kruppa's equations for camera self-calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1199–1204, Sep 2000. [Online]. Available: http://www.inrialpes.fr/movi/publi/Publications/2000/Stu00c

[39] J. R. Magnus and H. Neudecker, *"Matrix Differential Calculus with Applications in Statistics and Econometrics"*. John Wiley & Sons, 1999.

[40] C. Rothwell, O. Faugeras, and G. Csurka, "A comparison of projective reconstruction methods for pairs of views," *Computer Vision and Image Understanding*, vol. 68, no. 1, pp. 37–58, October 1997.