*Computer Vision Winter Workshop 2007, Michael Grabner, Helmut Grabner (eds.)*
*St. Lambrecht, Austria, February 6–8*
*Graz Technical University*

# Solving polynomial equations for minimal problems in computer vision

Zuzana Kúkelová and Tomáš Pajdla

CMP, Department of Cybernetics, Czech Technical University in Prague, Czech Republic
kukelova@cmp.felk.cvut.cz, pajdla@cmp.felk.cvut.cz

**Abstract** *Many vision tasks require efficient solvers of systems of polynomial equations. Epipolar geometry and relative camera pose computation are tasks which can be formulated as minimal problems which lead to solving systems of algebraic equations. Often, these systems are not trivial and therefore special algorithms have to be designed to achieve numerical robustness and computational efficiency. In this work we suggest improvements of current techniques for solving systems of polynomial equations suitable for some vision problems. We introduce two tricks. The first trick helps to reduce the number of variables and degrees of the equations. The second trick can be used to replace computationally complex construction of Gröbner basis by a simpler procedure. We demonstrate benefits of our technique by providing a solution to the problem of estimating radial distortion and epipolar geometry from eight correspondences in two images. Unlike previous algorithms, which were able to solve the problem from nine correspondences only, we enforce the determinant of the fundamental matrix be zero. This leads to a system of eight quadratic and one cubic equation. We provide an efficient and robust solver of this problem. The quality of the solver is demonstrated on synthetic and real data.*

## 1 Introduction

Estimating camera models from image matches is an important problem. It is one of the oldest computer vision problems and even though much has already been solved some questions remain still open. For instance, a number of techniques for modeling and estimating projection models of wide angle lenses appeared recently [6, 17, 9, 26, 27]. Often in this case, the projection is modeled as the perspective projection followed by radial "distortion" in the image plane.

Many techniques for estimating radial distortion based on targets [28, 30], plumb lines [1, 4, 12, 25], and multiview constraints [20, 29, 11, 6, 17, 27, 19, 14] have been suggested. The particularly interesting formulation, based on the division model [6], has been introduced by Fitzgibbon. His formulation leads to solving a system of algebraic equations. It is especially nice because the algebraic constraints of the epipolar geometry, $\det(\mathtt{F}) = 0$ for an uncalibrated and $2\,\mathtt{E}\,\mathtt{E}^\top\mathtt{E} - \mathrm{trace}(\mathtt{E}\,\mathtt{E}^\top)\mathtt{E} = 0$ for a calibrated situation [10], can be "naturally" added to the constraints arising from correspondences to reduce the number of points needed for estimating the distortion and the fundamental matrix. A smaller number of the points considerably reduces the number of samples in RANSAC [5, 10]. However, the resulting systems of polynomial equations are more difficult than, e.g., the systems arising from similar problems for estimating epipolar geometry of perspective cameras [23, 22]. In this paper we will solve the problem arising from taking $\det(\mathtt{F}) = 0$ constraint into account.

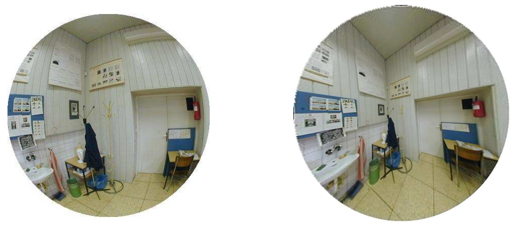
**Figure 1:** (Left) Image with radial distortion. (Right) Corrected image.

Fitzgibbon [6] did not use the algebraic constraints on the fundamental matrix. In fact, he did not explicitly pose his problem as finding a solution to a system of algebraic equations. Thanks to neglecting the constraints, he worked with a very special system of algebraic equations which can be solved numerically by using a quadratic eigenvalue solver. Micusik and Pajdla [17] also neglected the constraints when formulating the estimation of paracatadioptric camera model from image matches as a quartic eigenvalue problem. The work [19] extended Fitzbiggon's method for any number of views and any number of point correspondences using generalized quadratic eigenvalue problem for rectangular matrices, again without explicitly solving algebraic equations.

Li and Hartley [14] treated the original Fitzgibbon's problem as a system of algebraic equations and used the hidden variable technique [2] to solve them. No algebraic constraint on the fundamental matrix has been used. The resulting technique solves exactly the same problem as [6] but in a different way. Our experiments have shown that the quality of the result was comparable but the technique [14] was considerably slower than the original technique [6]. Work [14] mentioned the possibility of using the algebraic constraint $\det(\mathtt{F}) = 0$ to solve for a two parametric model from the same number of points but it did not use it to really solve the problem. Using this constraint makes the problem much harder because the degree of equations involved significantly increases.

We formulate the problem of estimating the radial distortion from image matches as a system of algebraic equations and by using the constraint $\det(\mathtt{F}) = 0$ we get a minimal solution to the autocalibration of radial distortion from eight correspondences in two views.

Our work adds a new minimal problem solution to the family of previously developed minimal problems, e.g. the perspective three point problem [5, 8], the five point relative pose problem [18, 22, 15], the six point focal length problem [23, 13], six point generalized camera problem [24].

We follow the general paradigm for solving minimal problems in which a problem is formulated as a set of algebraic equations which need to be solved. Our main contribution is in improving the technique for solving the set of algebraic equations and applying it to solve the minimal problem for the autocalibration of radial distortion. We use the algebraic constraint $\det(\mathtt{F}) = 0$ on the fundamental matrix to get an 8-point algorithm. It reduces the number of samples in RANSAC 1.15 (1.45, 2.53) times for 10% (30%, 60%) outliers and is more stable than previously known 9-point algorithms [6, 14].

## 2 Solving algebraic equations

In this section we will introduce the technique we use for solving systems of algebraic equations. We use the nomenclature from excellent monographs [3, 2], where basic concepts from polynomial algebra, algebraic geometry, and solving systems of polynomial equations are explained.

Our goal is to solve a system of algebraic equations $f_1(x) = ... = f_m(x) = 0$ which are given by a set of $m$ polynomials $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ in $n$ variables over the field $\mathbb{C}$ of complex numbers. We are only interested in systems which have a finite number, say $N$, solutions and thus $m \geq n$.

The ideal $I$ generated by polynomials $F$ can be written as

$$I = \left\{ \sum_{i=1}^{m} f_i \, p_i \, | \, p_i \in \mathbb{C}[x_1, ..., x_n] \right\}$$

with $f_1, ..., f_m$ being generators of $I$. The ideal contains all polynomials which can be generated as an algebraic combination of its generators. Therefore, all polynomials from the ideal are zero on the zero set $Z = \{x | f_1(x) = ... = f_m(x) = 0\}$. In general, an ideal can be generated by many different sets of generators which all share the same solutions. There is a special set of generators though, the reduced Gröbner basis $G = \{g_1, ..., g_l\}$ w.r.t. the lexicographic ordering, which generates the ideal $I$ but is easy (often trivial) to solve. Computing this basis and "reading off" the solutions from it is the standard method for solving systems of polynomial equations. Unfortunately, for most computer vision problems this "Gröbner basis method w.r.t. the lexicographic ordering" is not feasible because it has double exponential computational complexity in general.

To overcome this problem, a Gröbner basis $G$ under another ordering, e.g. the graded reverse lexicographical ordering, which is often easier to compute, is constructed. Then, the properties of the *quotient ring* $A = \mathbb{C}[x_1, ..., x_n]/I$, i.e. the set of equivalence classes represented by remainders modulo $I$, can be used to get the solutions. The linear basis of this quotient ring can be written as $B = \{\mathbf{x}^\alpha | \mathbf{x}^\alpha \notin \langle LM(I) \rangle\} = \left\{\mathbf{x}^\alpha | \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$, where $\mathbf{x}^\alpha$ is monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$, $\overline{\mathbf{x}^\alpha}^G$ is the reminder of $\mathbf{x}^\alpha$ on the division by $G$, and $\langle LM(I) \rangle$ is ideal generated by leading monomials of all polynomials form $I$. In many cases (when $I$ is radical [2]), the dimension of $A$ is equal to the number of solutions $N$. Then, the basis of $A$ consists of $N$ monomials, say $B = \left\{\mathbf{x}^{\alpha(1)}, ..., \mathbf{x}^{\alpha(N)}\right\}$. Denoting the basis as $b(\mathbf{x}) = \left[\mathbf{x}^{\alpha(1)} ... \mathbf{x}^{\alpha(N)}\right]^T$, every polynomial $q(\mathbf{x}) \in A$ can be expressed as $q(\mathbf{x}) = b(\mathbf{x})^T c$, where $c$ is a coefficient vector. The multiplication by a fixed polynomial $f(\mathbf{x})$ (a polynomial in variables $\mathbf{x} = (x_1, ..., x_n)$) in the quotient ring $A$ then corresponds to a linear operator $T_f : A \rightarrow A$ which can be described by a $N \times N$ action matrix $M_f$. The solutions to the set of equations can be read off directly from the eigenvalues and eigenvectors of the action matrices. We have

$$f(\mathbf{x}) \, q(\mathbf{x}) = f(\mathbf{x}) \left(b(\mathbf{x})^T c\right) = \left(f(\mathbf{x}) \, b(\mathbf{x})^T\right) c$$

Using properties of the action matrix $M_f$, we obtain

$$\left(f(\mathbf{x}) \, b(\mathbf{x})^T\right) c = b(\mathbf{x})^T M_f c,$$

Each polynomial $t \in \mathbb{C}[x_1, ..., x_n]$ can be written in the form $t = \sum_{i=1}^{l} h_i g_i + r$, where $g_i$ are basis vectors $g_i \in G = \{g_1, ..., g_l\}$, $h_i \in \mathbb{C}[x_1, ..., x_n]$ and $r$ is the reminder of $t$ on the division by $G$.

If $\mathbf{p} = (p_1, ..., p_n)$ is a solution to our system of equations, then we can write

$$f(\mathbf{p}) \, q(\mathbf{p}) = \left(f(\mathbf{p}) \, b(\mathbf{p})^T\right) c = \sum_{i=1}^{l} h_i(\mathbf{p}) \, g_i(\mathbf{p}) + r(\mathbf{p})$$

where $r(\mathbf{p})$ is the reminder of $f(\mathbf{p}) \, q(\mathbf{p})$ on the division by $G$. Because $g_i(\mathbf{p}) = 0$ for all $i = 1, ..., l$ we have $\sum_{i=1}^{l} h_i(\mathbf{p}) \, g_i(\mathbf{p}) + r(\mathbf{p}) = r(\mathbf{p})$ and therefore

$$\left(f(\mathbf{p}) \, b(\mathbf{p})^T\right) c = r(\mathbf{p}) = \overline{\left(f(\mathbf{p}) \, b(\mathbf{p})^T\right) c}^G.$$

Thus, for a solution $\mathbf{p}$, we have

$$\overline{\left(f(\mathbf{p}) \, b(\mathbf{p})^T\right) c}^G = \left(f(\mathbf{p}) \, b(\mathbf{p})^T\right) c = b(\mathbf{p})^T M_f c$$

for all $c$, and therefore

$$f(\mathbf{p}) \, b(\mathbf{p})^T = b(\mathbf{p})^T M_f.$$

Therefore, if $\mathbf{p} = (p_1, ..., p_n)$ is a solution to our system of equations and $f(\mathbf{x})$ is chosen such that the values $f(\mathbf{p})$ are distinct for all $\mathbf{p}$, the $N$ left eigenvectors of the action matrix $M_f$ are of the form

$$v = \beta b(\mathbf{p}) = \beta \left[\mathbf{p}^{\alpha(1)} ... \mathbf{p}^{\alpha(N)}\right]^T,$$

for some $\beta \in \mathbb{C}$, $\beta \neq 0$.

Thus action matrix $\mathtt{M}_f$ of the linear operator $T_f : A \rightarrow A$ of the multiplication by a suitably chosen polynomial $f$ w.r.t. the basis $B$ of $A$ can be constructed and then the solutions to the set of equations can be read off directly from the eigenvalues and eigenvectors of this action matrix [2].

### 2.1 Simplifying equations by lifting

The complexity of computing an action matrix depends on the complexity of polynomials (degree, number of variables, form, etc.). It is better to have the degrees as well as the number of variables low. Often, original generators $F$ may be transformed into new generators with lower degrees and fewer variables. Next we describe a particular transformation method—*lifting method*—which proved to be useful.

Assume $m$ polynomial equations in $l$ monomials. The main idea is to consider each monomial that appears in the system of polynomial equations as an unknown. In this way, the initial system of polynomial equations of arbitrary degree becomes linear in the new "monomial unknowns". Such system can by written in a matrix form as

$$\mathtt{M}X = 0$$

where $X$ is a vector of $l$ monomials and $\mathtt{M}$ is a $m \times l$ coefficient matrix.

If $m < l$, then a basis of $m - l$ dimensional null space of matrix $\mathtt{M}$ can be found and all monomial unknowns can be expressed as linear combinations of basic vectors of the null space. The coefficients of this linear combination of basic vectors become new unknowns of the new system which is formed by utilizing algebraic dependencies between monomials. In this way we obtain a system of polynomial equations in new variables. The new set of variables consists of unknown coefficients of linear combination of basic vectors and of old unknowns which we need for utilizing dependencies between monomials. The new system is equivalent to the original system of polynomial equations but may be simpler. This abstract description will be made more concrete in Section 3.1.

### 2.2 Constructing action matrix efficiently

The standard method for computing action matrices requires to construct a complete Gröbner basis and the linear basis $B$ of the algebra $A$ and to compute $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G$ for all $\mathbf{x}^{\alpha(i)} \in B = \left\{\mathbf{x}^{\alpha(1)}, ..., \mathbf{x}^{\alpha(N)}\right\}$ [2]. Note that $\mathbf{x}^{\alpha(i)} = x_1^{\alpha_1(i)} x_2^{\alpha_2(i)} ... x_n^{\alpha_n(i)}$. For some problems, however, it may be very expensive to find a complete Gröbner basis. Fortunately, to compute $\mathtt{M}_f$ we do not always need a complete Gröbner basis. Here we propose a method for constructing the action matrix assuming that the monomial basis $B$ of algebra $A$ is known or can be computed for a class of problems in advance.

Many minimal problems in computer vision have the convenient property that the monomials which appear in the set of initial generators $F$ are always same irrespectively from the concrete coefficients arising from non-degenerate image measurements. For instance, when computing the essential matrix from five points, we need to have five linear, linearly independent, equations in elements of $\mathtt{E}$ and ten higher order algebraic equations $2\,\mathtt{E}\,\mathtt{E}^\top\mathtt{E} - \mathrm{trace}(\mathtt{E}\,\mathtt{E}^\top)\,\mathtt{E} = 0$ and $\det(\mathtt{E}) = 0$ which do not depend on particular measurements. Therefore, the leading monomials of the corresponding Gröbner basis, and thus the monomials in the basis $B$ are always the same. They can be found once in advance. To do so, we use the approach originally suggested in [23, 21, 22] for computing Gröbner bases but we retrieve the basis $B$ and polynomials required for constructing the action matrix instead.

Having $B$, the action matrix can be computed as follows. If for some $\mathbf{x}^{\alpha(i)} \in B$ and chosen $f$, $f\mathbf{x}^{\alpha(i)} \in A$, then $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G = f\mathbf{x}^{\alpha(i)}$ and we are done. For all other $\mathbf{x}^{\alpha(i)} \in B$ for which $f\mathbf{x}^{\alpha(i)} \notin A$ consider polynomials $q_i = f\mathbf{x}^{\alpha(i)} + h_i$ from $I$ with $h_i \in A$. For these $\mathbf{x}^{\alpha(i)}$, $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G = \overline{q_i - h_i}^G = -h_i \in A$. Since polynomials $q_i$ are from the ideal $I$, we can generate them as algebraic combinations of the initial generators $F$. Write $h_i = \sum_{j=1}^N c_{ji}\mathbf{x}^{\alpha(j)}$ for some $c_{ji} \in \mathbb{C}$, $i = 1, ..., N$. Then the action matrix $M_f$ has the form

$$M_f = \begin{pmatrix} c_{11} & c_{12} & . & . & . & c_{1N} \\ c_{21} & . & & & & . \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & . \\ c_{N1} & c_{N2} & & & & c_{NN} \end{pmatrix}.$$

To get this action matrix $\mathtt{M}_f$, it suffice to generate polynomials $q_i = f\mathbf{x}^{\alpha(i)} + \sum_{j=1}^N c_{ji}\mathbf{x}^{\alpha(j)}$ from the initial generators $F$ for all these $\mathbf{x}^{\alpha(i)} \in B$. This in general seems to be as difficult as generating the Gröbner basis but we shall see that it is quite simple for the problem of calibrating radial distortion which we describe in the next section. It is possible to generate $q_i$'s by starting with $F$ and systematically generating new polynomials by multiplying them by individual variables and reducing them by the Gauss-Jordan elimination. This technique is a variation of the F4 algorithm for constructing Gröbner bases [7] and seems to be applicable to more vision problems. We are currently investigating it and will report more results elsewhere.

### 2.3 The solver

The algorithmic description of our solver of polynomial equations is as follows.

1. Assume a set $F = \{f_1, ..., f_m\}$ of polynomial equations.
2. Use the lifting to simplify the original set of polynomial equations if possible. Otherwise use the original set.
3. Fix a monomial ordering (The graded reverse lexicographical ordering is often good).
4. Use Macaulay 2 [21] to find the basis $B$ as the basis which repeatedly appears for many different choices of random coefficients. Do computations in a suitably chosen finite field to speed them up.
5. For suitably chosen polynomial $f$ construct the polynomials $q_i$ by systematically generating higher order polynomials from generators $F$. Stop when all $q_i$'s are found. Then construct the action matrix $\mathtt{M}_f$.
6. Solve the equations by finding eigenvectors of the action matrix. If the initial system of equations was transformed, extract the solutions to the original problem.

This method extends the Gröbner basis method proposed in [23, 21] (i) by using lifting to simplify the problem and (ii) by constructing the action matrix without constructing a complete Gröbner basis. This brings an important advantage for some problems. Next we will demonstrate it by showing how to solve the minimal problem for correcting radial distortion from eight point correspondences in two views.

## 3 A minimal solution for radial distortion

We want to correct radial lens distortion using the minimal number of image point correspondences in two views. We assume one-parameter division distortion model [6]. It is well known that for standard uncalibrated case without considering radial distortion, 7 point correspondences are sufficient and necessary to estimate the epipolar geometry. We have one more parameter, the radial distortion parameter $\lambda$. Therefore, we will need 8 point correspondences to estimate $\lambda$ and the epipolar geometry. To get this "8-point algorithm", we have to use the singularity of the fundamental matrix $\mathtt{F}$. We obtain 9 equations in 10 unknowns by taking equations from the epipolar constraint for 8 point correspondences

$$\mathbf{p}_{u_i}^{\top}(\lambda)\,\mathtt{F}\,\mathbf{p}'_{u_i}(\lambda) = 0, \quad i = 1,\dots,8$$

and the singularity of $\mathtt{F}$

$$\det(\mathtt{F}) = 0,$$

where $\mathbf{p}'_u(\lambda), \mathbf{p}_u(\lambda)$ represent homogeneous coordinates of a pair of undistorted image correspondences.

The one-parameter division model is given by the formula

$$\mathbf{p}_u \sim \mathbf{p}_d/(1+\lambda r_d^2)$$

where $\lambda$ is the distortion parameter, $\mathbf{p}_u = (x_u, y_u, 1)$, resp. $\mathbf{p}_d = (x_d, y_d, 1)$, are the corresponding undistorted, resp. distorted, image points, and $r_d$ is the radius of $\mathbf{p}_d$ w.r.t. the distortion center. We assume that the distortion center has been found, e.g., by [9]. We also assume square pixels, i.e. $r_d^2 = x_d^2 + y_d^2$. To use the standard notation, we write the division model as

$$\mathbf{x} + \lambda\mathbf{z} = \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} 0 \\ 0 \\ r_d^2 \end{pmatrix} \sim \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix}.$$

### 3.1 Reducing 9 to 7 unknowns by lifting

We simplify the original set of equations by lifting. The epipolar constraint gives 8 equations with 15 monomials (nine $1^{st}$order, five $2^{nd}$order, one $3^{rd}$ order)

$$(x_i + \lambda z_i)^T\,\mathtt{F}\,(x'_i + \lambda z'_i) = 0,\ i = 1,...,8$$

$$x_i^T \mathtt{F} x'_i + \lambda\left(x_i^T \mathtt{F} z'_i + z_i^T \mathtt{F} x'_i\right) + \lambda^2 z_i^T \mathtt{F} z'_i = 0,\ i = 1,...,8$$

$$\mathtt{F} = \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix}$$

We consider each monomial as an unknown and obtain 8 homogeneous equations linear in the new 15 monomial unknowns. These equation can be written in a matrix form

$$\mathtt{M}X = 0$$

where $X = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, \lambda f_3, \lambda f_6, \lambda f_7, \lambda f_8, \lambda f_9, \lambda^2 f_9,)^T$ and $\mathtt{M}$ is the coefficient matrix.

If we denote the $i$-th row of the matrix $\mathtt{M}$ as $m_i$ and write

$$x_i + \lambda z_i = \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} 0 \\ 0 \\ r_d^2 \end{pmatrix},$$

then $m_i = (x_d x'_d,\ x_d y'_d,\ x_d,\ y_d x'_d,\ y_d y'_d,\ y_d,\ x'_d,\ y'_d,\ 1,\ x_d r'^2_d,\ x_{yd} r'^2_d,\ r_d^2 x'_d,\ r_d^2 y'_d,\ r_d^2 + r'^2_d,\ r_d^2 r'^2_d)$.

We obtain 8 linear equations in 15 unknowns. So, in general we can find 7 dimensional null-space. We write

$$X = x_1N_1 + x_2N_2 + x_3N_3 + x_4N_4 + x_5N_5 + x_6N_6 + x_7N_7$$

where $N_1, ..., N_7 \in \mathbb{R}^{15\times 1}$ are basic vectors of the null space and $x_1, \dots, x_7$ are coefficients of the linear combination of the basic vectors. Assuming $x_7 \neq 0$, we can set $x_7 = 1$. Then we can write

$$X = \sum_{i=1}^{7} x_i N_i = \sum_{i=1}^{6} x_i N_i + N_7$$

$$X_j = \sum_{i=1}^{6} x_i N_{ij} + N_{7j}, \quad j = 1,..,15$$

Considering dependencies between monomials and $\det(\mathtt{F}) = 0$ we get 7 equations for 7 unknowns $x_1, x_2, x_3, x_4, x_5, x_6, \lambda$ :

$$X_{10} = \lambda.X_3 \Rightarrow \sum_{i=1}^{6} x_i N_{i,10} + N_{7,10} = \lambda \sum_{i=1}^{6} x_i N_{i,3} + N_{7,3}$$

$$X_{11} = \lambda.X_6 \Rightarrow \sum_{i=1}^{6} x_i N_{i,11} + N_{7,11} = \lambda \sum_{i=1}^{6} x_i N_{i,6} + N_{7,6}$$

$$X_{12} = \lambda.X_7 \Rightarrow \sum_{i=1}^{6} x_i N_{i,12} + N_{7,12} = \lambda \sum_{i=1}^{6} x_i N_{i,7} + N_{7,7}$$

$$X_{13} = \lambda.X_8 \Rightarrow \sum_{i=1}^{6} x_i N_{i,13} + N_{7,13} = \lambda \sum_{i=1}^{6} x_i N_{i,8} + N_{7,8}$$

$$X_{14} = \lambda.X_9 \Rightarrow \sum_{i=1}^{6} x_i N_{i,14} + N_{7,14} = \lambda \sum_{i=1}^{6} x_i N_{i,9} + N_{7,9}$$

$$X_{15} = \lambda.X_{14} \Rightarrow \sum_{i=1}^{6} x_i N_{i,15} + N_{7,15} = \lambda \sum_{i=1}^{6} x_i N_{i,14} + N_{7,14}$$

$$\det(\mathtt{F}) = 0 \Rightarrow \det \begin{pmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{pmatrix} = 0$$

This set of equations is equivalent to the initial system of polynomial equations but it is simpler because instead of eight quadratic and one cubic equation in 9 unknowns (assuming $f_9 = 1$) we have only 7 equations (six quadratic and one cubic) in 7 unknowns. We will use these 7 equations to create the action matrix for the polynomial $f = \lambda$.

### 3.2 Computing $B$ and the number of solutions

To compute $B$, we solve our problem in a random finite prime field $\mathbb{Z}_p$ $(\mathbb{Z}/\langle p\rangle)$ with $p >> 7$, where exact arithmetic can be used and numbers can be represented in a simple and efficient way. It speeds up computations and minimizes memory requirements.

We use algebraic geometric software Macaulay 2, which can compute in finite fields, to solve the polynomial equations for many random coefficients from $\mathbb{Z}_p$, to compute the number of solutions, the Gröbner basis, and the basis $B$. If the basis $B$ remains stable for many different random coefficients, it is generically equivalent to the basis of the original system of polynomial equations.

We can use the Gröbner basis and the basis $B$ computed for random coefficients from $\mathbb{Z}_p$ thanks to the fact that in our class of problems the way of computing the Gröbner basis is always the same and for particular data these Gröbner bases differ only in coefficients. This holds for $B$, which consists of the same monomials, as well. Also, the way of obtaining polynomials that are necessary to create the action matrix is always the same and for a general data the generated polynomials differ again only in their coefficients. This way we have found that our problem has 16 solutions. To create the action matrix, we use the graded reverse lexicographic ordering with $x_1 > x_2 > x_3 > x_4 > x_5 > \lambda > x_6$. With this ordering, we get the basis $B = (x_6^3, \lambda^2, x_1x_6, x_2x_6, x_3x_6, x_4x_6, x_5x_6, x_6^2, x_1, x_2, x_3, x_4, x_5, \lambda, x_6, 1)$ of the algebra $A = \mathbb{C}\,[x_1, x_2, x_3, x_4, x_5, \lambda, x_6]/I$ which, as we shall see later, is suitable for finding the action matrix $\mathtt{M}_\lambda$.

### 3.2.1 Computing the number of solutions and basis $B$

Here we show the program for computing the number of solutions and basis $B$ of our problem in Macaulay 2. Similar programs can be used to compute the number of solutions and basis of the algebra $A$ for other problems.

```
 // polynomial ring with coeffs from Z_30097
 R = ZZ/30097[x_1..x_9, MonomialOrder=>Lex];
// Formulate the problem over Z_p => the set
  of equations eq (known variables -> random
  numbers from Z_p )
F = matrix({{x_1,x_2,x_3},{x_4,x_5,x_6},
  {x_7,x_8,1_R}});
X1 = matrix{apply(8,i->(random(R^2,R^1)))}
  ||matrix({{1_R,1_R,1_R,1_R,1_R,1_R,1_R,1_R}});
Z1 = matrix({{0_R,0_R,0_R,0_R,0_R,0_R,0_R,0_R}})
  ||matrix({{0_R,0_R,0_R,0_R,0_R,0_R,0_R,0_R}})
  ||matrix{apply(8,i->X1_(0,i)^2+X1_(1,i)^2)};
X2 = matrix{apply(8,i->(random(R^2,R^1)))}
  ||matrix({{1_R,1_R,1_R,1_R,1_R,1_R,1_R,1_R}});
Z2 = matrix({{0_R,0_R,0_R,0_R,0_R,0_R,0_R,0_R}})
  ||matrix({{0_R,0_R,0_R,0_R,0_R,0_R,0_R,0_R}})
  ||matrix{apply(8,i->X2_(0,i)^2+X2_(1,i)^2)};
P1 = X1 + x_9*Z1;
P2 = X2 + x_9*Z2;
eq = apply(8, i->(transpose(P1_[i]))*F*(P2_[i]));
// Ideal generated by polynomials eq + the
  polynomial det(F)
I1 = ideal(eq) + ideal det F;
// Compute the number of solutions
gbTrace 100
dim I1  //the dimension of ideal I
  (zero-dimensional ideal <==> V(I) is a
  finite set
 degree I1 //the number of solutions (the
  number of points in V(I))
 transpose gens gb I1  // Groebner basis
 //the quotient ring A=ZZ/30097[x_1..x_9]/I1
  A = R/I1
 B = basis A //the basis of the quotient
  ring A
```

The above program in Macaulay 2 gives not only the number of solutions and the basis $B$ of the algebra $A$, but also the information like how difficult it is to compute the Gröbner basis, and how many and which S-polynomials have to be generated.

The level of verbosity is controlled with the command `gbTrace(n)`. For example for n=0 no extra information is produced and for n=100 we get which S-polynomials were computed, from which polynomials were these S-polynomials created, which S-polynomial did not reduce to 0, which were inserted into the basis and so on.

## 3.3 Constructing action matrix

Here we construct the action matrix $\mathtt{M}_\lambda$ for multiplication by polynomial $f = \lambda$. The method described in Section 2.2 calls for generating polynomials $q_i = \lambda \mathbf{x}^{\alpha(i)} + \sum_{j=1}^{N} c_{ji}\mathbf{x}^{\alpha(j)} \in I$.

In graded orderings, the leading monomials of $q_i$ are $\lambda \mathbf{x}^{\alpha(i)}$. Therefore, to find $q_i$, it is enough to generate at least one polynomial in the required form for each leading monomial $\lambda \mathbf{x}^{\alpha(i)}$. This can be, for instance, done by systematically generating polynomials of $I$ with ascending leading monomials and testing them. We stop when all necessary polynomials $q_i$ are obtained. Let $d$ be the degree of the highest degree polynomial from initial generators $F$. Then we can generate polynomials $q_i$ from $F$ in this way:

1. Generate all monomial multiples $\mathbf{x}^\alpha f_i$ of degree $\leq d$.
2. Write the polynomial equations in the form $\mathtt{M}X = 0$, where $\mathtt{M}$ is the coefficient matrix and $X$ is the vector of all monomials ordered by the used monomial ordering.
3. Simplify matrix $\mathtt{M}$ by the Gauss-Jordan (G-J) elimination.
4. If all necessary polynomials $q_i$ have been generated, stop.
5. If no new polynomials with degree $< d$ were generated by G-J elimination, set $d = d + 1$.
6. Go to 1.

In this way we can systematically generate all necessary polynomials. Unfortunately, we also generate many unnecessary polynomials. We use Macaulay 2 to identify the unnecessary polynomials and avoid generating them.

In the process of creating the action matrix $\mathtt{M}_\lambda$, we represent polynomials by rows of the matrix of their coefficients. Columns of this matrix are ordered according to the monomial ordering. The basic steps of generating the polynomials necessary for constructing the action matrix are as follows:

1. We begin with six 2nd degree polynomials $f_1^{(0)}, \ldots, f_6^{(0)}$ and one 3rd degree polynomial $f_7^{(0)} = \det(\mathtt{F}) = 0$. We perform G-J elimination of the matrix representing the six 2nd degree polynomials and reconstruct the six reduced polynomials $f_1^{(1)}, \ldots, f_6^{(1)}$.

2. We multiply $f_1^{(1)}, \dots, f_6^{(1)}$ by $1, x_1, x_2, x_3, x_4, x_5, \lambda, x_6$ and add $f_7^{(0)}$ to get 49 $2^{nd}$ and $3^{rd}$ degree polynomials $f_1^{(2)}, \dots, f_{49}^{(2)}$. They can be represented by 119 monomials and a $49 \times 119$ matrix with rank 49, which we simplify by one G-J elimination again.

3. We obtain 15 new $2^{nd}$ degree polynomials $(f_{29}^{(2)} \dots, f_{43}^{(2)})$, six old $2^{nd}$ degree polynomials (reduced polynomials $f_1^{(1)}, \dots, f_6^{(1)}$, now $f_{44}^{(2)} \dots, f_{49}^{(2)}$) and 28 polynomials of degree three. In order to avoid adding $4^{th}$ degree polynomials on this stage we add only $x_1, x_2, x_3, x_4, x_5, \lambda, x_6$ multiples of these 15 new $2^{nd}$ degree polynomials to the polynomials $f_1^{(2)}, \dots, f_{49}^{(2)}$. Thus obtaining 154 polynomials $f_1^{(3)}, \dots, f_{154}^{(3)}$ representable by a $154 \times 119$ matrix, which has rank 99. We simplify it by G-J elimination and obtain $f_1^{(4)}, \dots, f_{99}^{(4)}$.

4. The only $4^{th}$ degree polynomial that we need is a polynomial in the form $\lambda x_6^3 + h$, $h \in A$. To obtain this polynomial, we only need to add monomial multiples of one polynomial $g$ from $f_1^{(4)}, ..., f_{99}^{(4)}$ which has leading monomial $LM(g) = \lambda x_6^2$. This is possible thanks to our monomial ordering. All polynomials $f_1^{(4)}, ..., f_{99}^{(4)}$ and $x_1, x_2, x_3, x_4, x_5, \lambda, x_6$ multiples of the $3^{rd}$ degree polynomial $g$ with $LM(g) = \lambda x_6^2$ give 106 polynomials $f_1^{(5)}, ..., f_{106}^{(5)}$ which can be represented by a $106 \times 126$ matrix of rank 106. After another G-J elimination, we get 106 reduced polynomials $f_1^{(6)}, ..., f_{106}^{(6)}$. Because the polynomial $g$ with $LM(g) = \lambda x_6^2$ has already been between the polynomials $f_1^{(2)}, ..., f_{49}^{(2)}$, we can add its monomial multiples already in the $3^{rd}$ step. After one G-J elimination we get the same 106 polynomials. In this way, we obtain polynomial $q$ with $LM(q) = \lambda x_6^3$ as $q = \lambda x_6^3 + c_1 x_2 x_6^2 + c_2 x_3 x_6^2 + c_3 x_4 x_6^2 + c_4 x_5 x_6^2 + h'$, for some $c_1, ..., c_4 \in \mathbb{C}$ and $h' \in A$ instead of the desired $\lambda x_6^3 + h$, $h \in A$.

5. Among the polynomials $f_1^{(6)}, ..., f_{106}^{(6)}$, there are 12 out of the 14 polynomials that are required for constructing the action matrix. The first polynomial which is missing is the above mentioned polynomial $q_1 = \lambda x_6^3 + h_1$, $h_1 \in A$. To obtain this polynomial from $q$, we need to generate polynomials from the ideal with leading monomials $x_2 x_6^2$, $x_3 x_6^2$, $x_4 x_6^2$, and $x_5 x_6^2$. The second missing polynomial is $q_2 = \lambda^3 + h_2$, $h_2 \in A$. All these $3^{rd}$ degree polynomials from the ideal $I$ can be, unfortunately, obtained only by eliminating the $4^{th}$ degree polynomials. To get these $4^{th}$ degree polynomials, the polynomial with leading monomial $x_1 x_6^2$, resp. $x_2 x_6^2$, $x_3 x_6^2$, $x_4 x_6^2$, $x_5 x_6^2$ is multiplied by $\lambda$ and subtracted from the polynomial with leading monomial $\lambda x_6$ multiplied by $x_1 x_6$, resp. by $x_2 x_6$, $x_3 x_6$, $x_4 x_6$, $x_5 x_6$. After G-J elimination, a polynomial with the leading monomial $x_2 x_6^2$, resp. $x_3 x_6^2$, $x_4 x_6^2$, $x_5 x_6^2$, $\lambda^3$ is obtained.

6. All polynomials needed for constructing the action matrix are obtained. Action matrix $M_\lambda$ is constructed.

### 3.4 Solving equations using eigenvectors

The eigenvectors of $\mathtt{M}_\lambda$ give solutions for $x_1, x_2, x_3, x_4, x_5, \lambda, x_6$. Using a backsubstitution, we obtain solutions for $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, \lambda$. In this way we obtain 16 (complex) solutions. Generally less than 10 solutions are real.

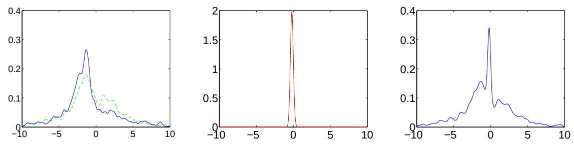

**Figure 2:** Distribution of real roots in $[-10, 10]$ using kernel voting for 500 noiseless point matches, 200 estimations and $\lambda_{true} = -0.2$. (Left) Parasitic roots (green) vs. roots for mismatches (blue). (Center) Genuine roots. (Right) All roots, 100% of inliers.

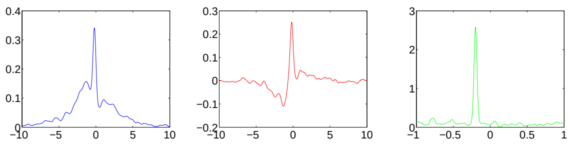

**Figure 3:** Distribution of real roots using kernel voting for 500 noiseless point matches, 100% inliers, 200 groups and $\lambda_{true} = -0.2$. (Left) Distribution of all roots in $[-10, 10]$. (Center) Distribution of all roots minus the distribution of roots from mismatches in $[-10, 10]$. (Right) Distribution of all roots in $[-1, 1]$.

## 4 Experiments

We test our algorithm on both synthetic (with various levels of noise, outliers and radial distortions) and real images and compare it to the existing 9-point algorithms for correcting radial distortion [6, 14]. We can get up to 16 complex roots. In general, more than one and less than 10 roots are real. If there is more than one real root, we need to select the best root, the root which is consistent with most measurements. To do so, we treat the real roots of the 16 (in general complex) roots obtained by solving the equations for one input as real roots from different inputs and use RANSAC [5, 10] or kernel voting [14] for several (many) inputs to select the best root among all generated roots. The kernel voting is done by a Gaussian kernel with fixed variance and the estimate of $\lambda$ is found as the position of the largest peak. See [14] for more on kernel voting for this problem. To evaluate the performance of our algorithm, we distinguish three sets of roots. "All roots" is the set of all real roots obtained by solving the equations for $K$ (different) inputs. "Genuine roots" denote the subset of all roots obtained by selecting the real root closest to the true $\lambda$ for each input containing only correct matches. The set of genuine roots can be identified only in simulated experiments. "Parasitic roots" is the subset of all roots obtained by removing the genuine roots from all roots when everything is evaluated on inputs containing only correct matches. The results of our experiments for the kernel voting are shown in Figure 2. Figure 2 (Left) shows that the distribution of all real roots for mismatches is similar to the distribution of the parasitic roots. This allows to treat parasitic roots in the same way as the roots for mismatches. Figures 2 (Left and Center) show that the distribution of genuine roots is very sharp compared to the distribution of parasitic roots and roots for mismatches. Therefore, it is

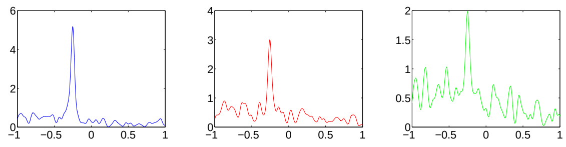

**Figure 4:** Kernel voting results, for $\lambda_{true} = -0.25$, noise level $\sigma = 0.2$ (1 pixel), image size $768 \times 576$ and (Left) 100% inliers, (Center) 90% inliers (Right) 80% inliers. Estimated radial distortion parameters were (Left) $\lambda = -0.2510$ (Center) $\lambda = -0.2546$ (Right) $\lambda = -0.2495$.

possible to estimate the true $\lambda$ as the position of the largest peak, Figure 2 (Right). These experiments show that it is suitable to use kernel voting and that it make sense to select the best root by casting votes from all computed roots. It is clear from results shown in Figure 3 that it is meaningful to vote for $\lambda$'s either (i) within the range where the most of the computed roots fall (in our case [-10,10]), Figure 3 (Left), or (ii) within the smallest range in which we are sure that the ground truth lie (in our case [-1,1]), Figure 3 (Right). For large number of input data, it might also makes sense to subtract the apriory computed distribution of all real roots for mismatches from the distribution of all roots.

### 4.1 Tests on synthetic images

We initially studied our algorithm using synthetic datasets. Our testing procedure was as follows:

1. Generate a 3D scene consisting of $N$ $(= 500)$ random points distributed uniformly within a cuboid. Project $M\%$ of the points on image planes of the two displaced cameras. These are matches. In both image planes, generate $(100 - M)\%$ random points distributed uniformly in the image. These are mismatches. Altogether, they become undistorted correspondences.
2. Apply the radial distortion to the undistorted correspondences to generate noiseless distorted points.
3. Add Gaussian noise of standard deviation $\sigma$ to the distorted points.
4. Repeat $K$ times (We use $K = 100$ here, but in many cases $K$ from 30 to 50 is sufficient).
   (a) Randomly choose 8 point correspondences from given $N$ correspondences.
   (b) Normalize image point coordinates to $[-1, 1]$
   (c) Find up to 16 roots of the minimal solution to the autocalibration of radial distortion.
   (d) Select the real roots in the feasible interval, e.g., $-1 < \lambda < 1$ and the corresponding $\mathtt{F}$'s.
5. Use kernel voting to select the best root.

The resulting density functions for different outlier contaminations and for the noise level 1 pixel are shown in Figure 4. Here, $K = 100$, image size was $768 \times 576$ and $\lambda_{true} = -0.25$. In all cases, a good estimate, very close to the true $\lambda$, was found as the position of the maximum of the root density function. We conclude, that the method is robust to mismatches and noise.

In the next experiment we study the robustness of our algorithm to increasing levels of Gaussian noise added to the distorted points. We compare our results to the results of two existing 9-point algorithms [6, 14]. The ground truth radial distortion $\lambda_{true}$ was $-0.5$ and the level of noise varied from $\sigma = 0$ to $\sigma = 1$, i.e. from 0 to 5 pixels. Noise level 5 pixels is relatively large but we get good results even for this noise level and 20% of outliers.

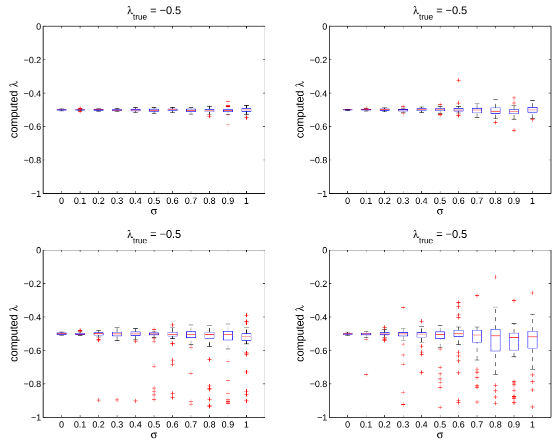

**Figure 5:** Estimated $\lambda$ as a function of noise $\sigma$, ground truth $\lambda_{true} = -0.5$ and (Top) $inliers = 90\%$, (Bottom) $inliers = 80\%$. Blue boxes contain values from 25% to 75% quantile. (Left) 8-point algorithm. (Right) 9-point algorithm.

Figure 5 (Left) shows $\lambda$ computed by our 8-point algorithm as a function of noise level $\sigma$. Fifty lambdas were estimated from fifty 8-tuples of correspondences randomly drawn for each noise level and (Top) 90% and (Bottom) 80% of inliers. The results are presented by the Matlab function *boxplot* which shows values 25% to 75% quantile as a blue box with red horizontal line at median. The red crosses show data beyond 1.5 times the interquartile range. The results for 9-point algorithms [6, 14], which gave exactly identical results, are shown for the same input, Figure 5 (Right).

The median values (from -0.50 to -0.523) for 8-point as well as 9-point algorithms are very close to the ground truth value $\lambda_{true} = -0.5$ for all noise levels. The variances of the 9-point algorithms, Figure 5 (Right), are considerably larger, especially for higher noise levels, than the variances of the 8-point algorithm Figure 5 (Left). The 8-point algorithm thus produces higher number of good estimates for the fixed number of samples. This is good both for RANSAC as well as for kernel voting.

### 4.2 Tests on real images

The input images with relatively large distortion, Figures 1 (Left) and 6 (Left), were obtained as cutouts from 180° angle of view fish-eye images. Tentative point correspondences were found by the wide base-line matching algorithm [16]. They contained correct as well as incorrect matches. Distortion parameter $\lambda$ was estimated by our 8-point algorithm and the kernel voting method. The input (Left) and corrected (Right) images are presented in Fig-

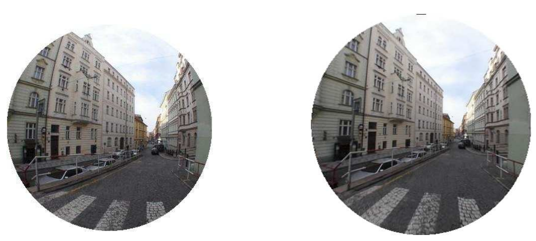

**Figure 6:** Real data. (Left) Input image with significant radial distortion. (Right) Corrected image.

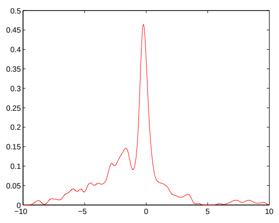

**Figure 7:** Distribution of real roots obtained by kernel voting for image in Figure 6. Estimated $\lambda = -0.22$.

ures 1 and 6. Figure 7 shows the distribution of real roots, for image from figure 6 (Left), from which $\lambda = -0.22$ was estimated as the argument of the maximum.

## 5 Conclusion

In this work we suggest improvements of current techniques for solving systems of polynomial equations and apply them to the minimal problem for the autocalibration of radial distortion. Our algorithm reduces the number of samples in RANSAC and is more stable than previously known 9-point algorithms. Our current MATLAB implementation of this algorithm runs about 0.05s on a P4/2.8GHz CPU. Most of this time is spent in the Gauss-Jordan elimination. However this time can be still reduced by further optimization. For comparison our MATLAB implementation of Fitzgibbon's algorithm runs about 0.004s and the original implementation of Hongdong Li's algorithm [14] based on MATLAB Symbolic-Math Toolbox runs about 0.86s.

## Acknowledgement

This work was supported by MSM6840770038 and the EC project FP6-IST-027787 DIRAC. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

## References

[1] C. Bräuer-Burchardt and K. Voss. A new algorithm to correct fish-eye and strong wide-angle-lens-distortion from single images. *ICIP 2001*, pp. 225–228.

[2] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, 2005.

[3] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, 1992.

[4] D. Devernay and O. Faugeras. Straight lines have to be straight. *MVA*, 13(1):14–24, 2001.

[5] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[6] A. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. *CVPR 2001*, pp. 125–132.

[7] J.-C. Faugere. A new efficient algorithm for computing gröbner bases ($f_4$). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[8] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE PAMI*, 25(8):930–943, 2003.

[9] R. Hartley and S. Kang. Parameter-free radial distortion correction with centre of distortion estimation. *ICCV 2005*, pp. 1834–1841.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[11] S. Kang. Catadioptric self-calibration. *CVPR 2000*,

[12] S. Kang. Radial distortion snakes. *IAPR MVA Workshop 2000*, pp. 603–606, Tokyo.

[13] H. Li. A simple solution to the six-point two-view focal-length problem. *ECCV 2006*, pp. 200–213.

[14] H. Li and R. Hartley. A non-iterative method for correcting lens distortion from nine-point correspondences. *OMNIVIS 2005*.

[15] H. Li and R. Hartley. Five-point motion estimation made easy. *ICPR 2006*, pp. 630–633.

[16] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[17] B. Micusik and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. *CVPR 2003*,

[18] D. Nister. An efficient solution to the five-point relative pose. *IEEE PAMI*, 26(6):756–770, 2004.

[19] R. Steele and C. Jaynes. Overconstrained linear estimation of radial distortion and multi-view geometry. *ECCV 2006*,

[20] G. Stein. Lens distortion calibration using point correspondences. *CVPR 1997*, pp. 600:602.

[21] H. Stewenius. *Gröbner basis methods for minimal problems in computer vision*. PhD thesis, Lund University, 2005.

[22] H. Stewenius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS J. of Photogrammetry and Remote Sensing*, 60:284–294, 2006.

[23] H. Stewenius, D. Nister, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *CVPR 2005*, pp. 789–794.

[24] H. Stewenius, D. Nister, M. Oskarsson, and K. Astrom. Solutions to minimal generalized relative pose problems. *OMNIVIS 2005*.

[25] R. Strand and E. Hayman. Correcting radial distortion by circle fitting. *BMVC 2005*.

[26] S. Thirthala and M. Pollefeys. Multi-view geometry of 1d radial cameras and its application to omnidirectional camera calibration. *ICCV 2005*, pp. 1539–1546.

[27] S. Thirthala and M. Pollefeys. The radial trifocal tensor: A tool for calibrating the radial distortion of wide-angle cameras. *CVPR 2005* pp. 321–328.

[28] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. of Robotics and Automation*, 3(4):323344, 1987.

[29] Z. Zhang. On the epipolar geometry between two images with lens distortion. In *ICPR 1996*.

[30] Z. Zhang. A flexible new technique for camera calibration. *IEEE PAMI*, 22(11):1330–1334, 2000.