A minimal solution to the autocalibration of radial distortion

Zuzana Kukelova Tomas Pajdla
Center for Machine Perception, Dept. of Cybernetics, Faculty of Elec. Eng.
Czech Technical University in Prague, Karlovo nám. 13, 121 35 Prague, Czech Rep.
{kukelova,pajdla}@cmp.felk.cvut.cz

Abstract

Epipolar geometry and relative camera pose computation are examples of tasks which can be formulated as minimal problems and solved from a minimal number of image points. Finding the solution leads 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 paper we provide 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 in nine variables. We simplify this system by eliminating six of these variables. Then, we solve the system by finding eigenvectors of an action matrix of a suitably chosen polynomial. We show how to construct the action matrix without computing complete Gröbner basis, which provides an efficient and robust solver. The quality of the solver is demonstrated on synthetic and real data.

1. Introduction

Estimating[1] 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 [29, 31], plumb lines [1, 4, 12, 25], and multiview constraints [20, 30, 11, 6, 17, 27, 19, 14] have been suggested. The particularly interesting formulation, based on the division model, has been introduced by Fitzgibbon [6]. 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. In this paper we will solve the problem arising from taking $\det(\mathtt{F}) = 0$ constraint into account. 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].

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

Fitzgibbon [6] did not use the algebraic constraints on the fundamental matrix. 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.

[1]This work has been supported by grants EU FP6-IST-027787 DIRAC and MSM6840770038 DMCM III.

1-4244-1180-7/07/$25.00 ©2007 IEEE

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. This brings three benefits. First, we directly obtain a valid fundamental matrix. Secondly, the number of samples in RANSAC is reduced 1.15 (1.45, 2.53) times for 10% (30%, 60%) outliers. Finally, the solver is more stable than previously known 9-point algorithms [6, 14].

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].

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 the necessary 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 the polynomials F can be written as $I = \{\sum_{i=1}^{m} f_i p_i \,|\, p_i \in \mathbb{C}[x_1, ..., x_n]\}$ with $f_1, ..., f_m$ being generators of I. 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 one 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.

Therefore for some problems, a Gröbner basis G under another ordering, e.g. the graded reverse lexicographic 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 "action" matrix $\mathtt{M}_f$ of the linear operator $T_f \colon A \to A$ of the multiplication by a suitably chosen polynomial f w.r.t. the basis $B = \left\{\mathbf{x}^\alpha | \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ of A, where $\mathbf{x}^\alpha$ is a monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$ and $\overline{\mathbf{x}^\alpha}^G$ is the reminder of $\mathbf{x}^\alpha$ on the division by G, can be constructed. The solutions to the set of equations can then be read off directly from the eigenvalues and the eigenvectors of the action matrix [2].

2.1. 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 the algebra A is known or can be computed for a class of problems in advance.

Many minimal problems in computer vision, including this one, 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 generally 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 $c_{ji} \in \mathbb{C}$, $i = 1, ..., N$. To get the action matrix $\mathtt{M}_f = (c_{ij})$, it suffice to generate polynomials $q_i = f\mathbf{x}^{\alpha(i)} + \sum_{j=1}^{N} c_{ji}\mathbf{x}^{\alpha(j)}$ for all these $\mathbf{x}^{\alpha(i)} \in B$ from the initial generators F. 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 already generated polynomials 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.

2.2. 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. Simplify the original set of polynomial equations if possible. Otherwise use the original set.
3. Fix a monomial ordering (The graded reverse lexicographic 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. Construct the polynomials q_i for a suitably chosen polynomial f 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 the 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] by constructing the action matrix without constructing a complete Gröbner basis. This brings an important advantage for some problems. Next we show how we use this method 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 λ. Therefore, we will need 8 point correspondences to estimate λ 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

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

$$\mathtt{F} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{pmatrix} \tag{2}$$

and the singularity of $\mathtt{F}$

$$\det(\mathtt{F}) = 0, \tag{3}$$

where $\mathbf{p}'_u(\lambda), \mathbf{p}_u(\lambda)$ represent homogeneous coordinates of a pair of undistorted image correspondences. Assuming $f_{3,3} \neq 0$, we can set $f_{3,3} = 1$.

The one-parameter division model is given by the formula

$$\mathbf{p}_u \sim \mathbf{p}_d/(1 + \lambda r_d^2), \tag{4}$$

where λ 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. $r_d^2 = x_d^2 + y_d^2$.

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. Therefore, in the first step we simplify the original set of equations by eliminating some variables.

3.1. Eliminating variables

The epipolar constraint gives 8 equations with 15 monomials $(f_{1,3}\lambda, f_{2,3}\lambda, f_{3,1}\lambda, f_{3,2}\lambda, \lambda^2, f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda, 1)$ and 9 variables $(f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda)$.

We have four variables which appear in one monomial only $(f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2})$ and four variables which appear in two monomials $(f_{1,3}, f_{2,3}, f_{3,1}, f_{3,2})$. Since we have 8 equations from which each contains all 15 monomials, we can eliminate 6 variables, four variables which appear in one monomial only and two from variables which appear in two monomials. We have selected $f_{1,3}$ and $f_{2,3}$.

We reorder monomials contained in 8 equations such that monomials containing $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}$ and $f_{2,3}$ are

at the beginning. Reordered monomial vector will be $X = (f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda, f_{1,3}, f_{2,3}\lambda, f_{2,3}, f_{3,1}\lambda, f_{3,2}\lambda, \lambda^2, f_{3,1}, f_{3,2}, \lambda, 1)^T$.

Then, 8 equations from the epipolar constraint can be written in a matrix form $\mathtt{M}\,X = 0$, where $\mathtt{M}$ is the coefficient matrix. After performing Gauss-Jordan (G-J) elimination we obtain 8 equations of the form

$$f_i = LT(f_i) + g_i(f_{3,1}, f_{3,2}, \lambda) = 0, \tag{5}$$

where $LT(f_i) = f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda, f_{1,3}, f_{2,3}\lambda$ resp. $f_{2,3}$ for $i = 1, 2, 3, 4, 5, 6, 7$ resp. 8 and $g_i(f_{3,1}, f_{3,2}, \lambda)$ are 2^{nd}order polynomials in three variables $f_{3,1}, f_{3,2}, \lambda$. So we can express 6 variables, $f_{1,1}, f_{1,2}, f_{1,3}$, $f_{2,1}$, $f_{2,2}, f_{2,3}$ as the functions of the remaining three variables $f_{3,1}, f_{3,2}, \lambda$

$$f_{1,1} = -g_1(f_{3,1}, f_{3,2}, \lambda) \tag{6}$$
$$f_{1,2} = -g_2(f_{3,1}, f_{3,2}, \lambda) \tag{7}$$
$$f_{1,3} = -g_6(f_{3,1}, f_{3,2}, \lambda) \tag{8}$$
$$f_{2,1} = -g_3(f_{3,1}, f_{3,2}, \lambda) \tag{9}$$
$$f_{2,2} = -g_4(f_{3,1}, f_{3,2}, \lambda) \tag{10}$$
$$f_{2,3} = -g_8(f_{3,1}, f_{3,2}, \lambda). \tag{11}$$

We can substitute these expressions to the remaining two equations from the epipolar constraint and also to the singularity constraint for F. In this way we obtain 3 polynomial equations in 3 unknowns (two 3^{rd}order polynomials and one 5^{th}order polynomial)

$$\lambda(-g_6(f_{3,1}, f_{3,2}, \lambda)) + g_5(f_{3,1}, f_{3,2}, \lambda) = 0 \tag{12}$$
$$\lambda(-g_8(f_{3,1}, f_{3,2}, \lambda)) + g_7(f_{3,1}, f_{3,2}, \lambda) = 0 \tag{13}$$
$$\det \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & -g_8 \\ f_{3,1} & f_{3,2} & 1 \end{pmatrix} = 0. \tag{14}$$

We will use these 3 equations to create the action matrix for the polynomial $f = \lambda$. Without this elimination of variables, the creation of the action matrix for this problem will be almost impossible in a reasonable time.

3.2. Computing B and the number of solutions

To compute B, we solve our problem in a random chosen 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 geometry 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 [28].

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 $f_{3,1} > f_{3,2} > \lambda$. With this ordering, we get the basis $B = (f_{3,1}^3, f_{3,1}^2 f_{3,2}, f_{3,1} f_{3,2}^2, f_{3,2}^3, f_{3,2}^2\lambda, \lambda^3, f_{3,1}^2, f_{3,1}f_{3,2}, f_{3,2}^2, f_{3,1}\lambda, f_{3,2}\lambda, \lambda^2, f_{3,1}, f_{3,2}, \lambda, 1)$ of the algebra $A = \mathbb{C}[f_{3,1}, f_{3,2}, \lambda]/I$.

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 2.1 calls for generating polynomials $q_i = \lambda\mathbf{x}^{\alpha(i)} + \sum_{j=1}^{N} c_{ji}\mathbf{x}^{\alpha(j)} \in I$ for $\mathbf{x}^{\alpha(i)} \in B$.

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 G-J elimination.
4. If all polynomials q_i have been generated, stop.
5. Otherwise:
   (a) If a new polynomial with degree $< d$ has been generated by G-J elimination (this polynomial wasn't between polynomials before G-J elimination), return to the step 1.
   (b) If no new polynomials with degree $< d$ were generated by G-J elimination, set $d = d + 1$ and return to the step 1.

In this way we can systematically generate all necessary polynomials.

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 steps of generating the polynomials necessary for constructing the action matrix for the minimal radial distortion problem are as follows:

1. We begin with two 3^{rd} degree polynomials and one 5^{th} degree polynomial. In the first step we multiply these two 3^{rd} degree polynomials with all three variables $f_{3,1}, f_{3,2}, \lambda$. These two 3^{rd} degree polynomials and their multiples can be represented by 22 monomials and a 8×22 matrix of rank 8 which we simplify by G-J elimination.
2. We obtain one new 3rd degree polynomial and 5 new 4^{th} degree polynomials. In this step we add $f_{3,1}, f_{3,2}, \lambda$ multiples of this new 3^{rd} degree polynomial to already generated 8 polynomials. Thus we obtain 11 polynomials representable by a 11×30 matrix which has rank 11. We simplify it by G-J elimination.
3. We obtain one new 3^{rd} degree polynomial and 2 new 4^{th} degree polynomials. In this last step we add $f_{3,1}, f_{3,2}, \lambda$ multiples of this new 3^{rd} degree polynomial and all seven 4^{th} degree polynomials (2 new and the 5 generated in Step 1) to already generated 11 polynomials. Together with 5^{th} degree polynomial from $\det(F)$, we obtain 36 polynomials representable by a 36×50 matrix. It has rank 26. We perform another G-J elimination on this matrix.
4. All polynomials needed for constructing the action matrix are obtained. Action matrix M_λ is constructed.

3.4. Solving equations using eigenvectors

The eigenvectors of $\mathtt{M}_\lambda$ give solutions for $f_{3,1}, f_{3,2}, \lambda$. Using a backsubstitution, we obtain solutions also for $f_{1,1}$, $f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}$. In this way we obtain 16 (complex) solutions. Generally less than 10 solutions are real.

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 get 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] or kernel voting [14] for several 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 λ is found as the position of the largest peak. See [14] for more on kernel voting for this problem.

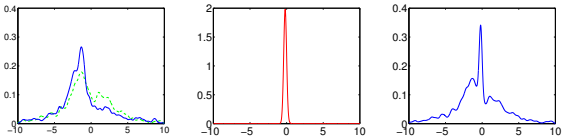

Figure 2. Distribution of real roots in $[-10, 10]$ using kernel voting for 500 noiseless point matches, 200 estimations and $\lambda_{true} = -0.2$. (Left) Parasitic roots (green) vs. roots for mismatches (blue). (Center) Genuine roots. (Right) All roots, 100% of inliers.

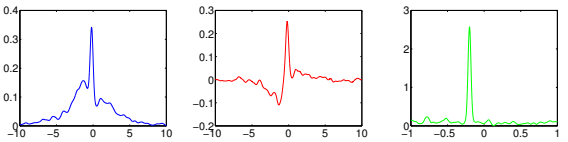

Figure 3. Distribution of real roots using kernel voting for 500 noiseless point matches, 100% inliers, 200 groups and $\lambda_{true} = -0.2$. (Left) Distribution of all roots in $[-10, 10]$. (Center) Distribution of all roots minus the distribution of roots from mismatches in $[-10, 10]$. (Right) Distribution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

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 λ 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.

Figure 2 shows the result of a simulation demonstrating that parasitic roots beware quite randomly. The distribution of all real roots for mismatches is similar to the distribution of the parasitic roots Figure 2 (Left). 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 possible to estimate λ as the position of the largest peak, Figure 2 (Right).

These experiments show that it is suitable to use kernel voting and that it makes 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 λ'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.

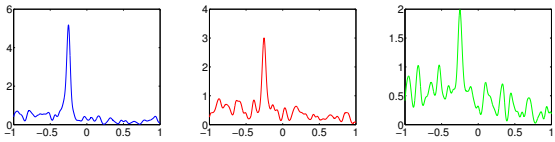

Figure 4. Kernel voting results, for $\lambda_{true} = -0.25$, noise level $\sigma = 0.2$ (1 pixel), image size 768×576 and (Left) 100% inliers, (Center) 90% inliers (Right) 80% inliers. Estimated radial distortion parameters were (Left) $\lambda = -0.2485$ (Center) $\lambda = -0.2522$ (Right) $\lambda = -0.2445$.

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. Generate $(100 - M)\%$ random points distributed uniformly in both images. 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 σ 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 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 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 768×576 and $\lambda_{true} = -0.25$. In all cases, a good estimate, very close to the true λ, 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 λ_{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

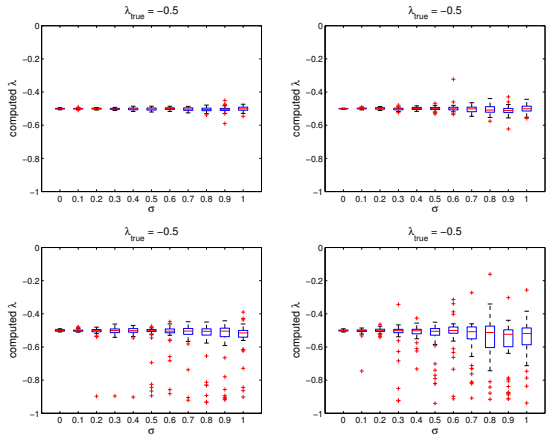

Figure 5. Estimated λ as a function of noise σ, ground truth $\lambda_{true} = -0.5$ and (Top) $inliers = 90\%$, (Bottom) $inliers = 80\%$. Blue boxes contain values from 25% to 75% quantile. (Left) 8-point algorithm. (Right) 9-point algorithm.

5 pixels is relatively large but we get good results even for this noise level and 20% of outliers.

Figure 5 (Left) shows λ's computed by our 8-point algorithm as a function of noise level σ. Fifty lambdas were estimated from fifty 8-tuples of correspondences randomly drawn for each noise level for (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 the 8-point as well as the 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 λ was estimated by our 8-point algorithm and the kernel voting method. The input and corrected images are presented in Figures 1 and 6. Figure 6 (Right) shows the distribution of real roots, for image from Figure 6 (Left), from which $\lambda = -0.22$ was estimated

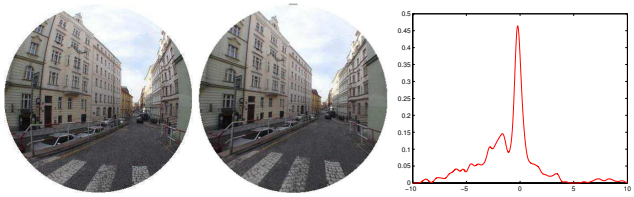

Figure 6. Real data. (Left) Input image with significant radial distortion. (Center) Corrected image. (Right) Distribution of real roots obtained by kernel voting for this image and estimated $\lambda = -0.22$.

as the argument of the maximum.

5. Conclusion

We presented a robust and efficient solution to the minimal problem for the autocalibration of radial distortion. It was obtained by a careful specialization of a general technique for solving polynomial equations. Our algorithm provides singular fundamental matrices, reduces the number of samples in RANSAC and is more stable than previously known 9-point algorithms. Our current MATLAB implementation of the algorithm runs about 0.01 s on a P4/2.8GHz CPU. Most of this time is spent in the Gauss-Jordan elimination. This time can still be reduced by further optimization. For comparison, our MATLAB implementation of Fitzgibbon's algorithm runs about 0.004 s and the original implementation of Hongdong Li's algorithm [14] based on MATLAB Symbolic-Math Toolbox runs about 0.86 s.

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. Faugère. 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*, pp. 485–490.

[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*, pp. 253–264.

[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] C. Traverso. Gröbner trace algorithms. In *ISSAC*, pages 125–138, 1988.

[29] 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.

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

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