A general solution to the P4P problem for camera with unknown focal length

Martin Bujnak, Zuzana Kukelova, Tomas Pajdla
Center for Machine Perception
Czech Technical University, Prague
{bujnam1,kukelova,pajdla}@cmp.felk.cvut.cz

Abstract

This paper presents a general solution to the determination of the pose of a perspective camera with unknown focal length from images of four 3D reference points. Our problem is a generalization of the P3P and P4P problems previously developed for fully calibrated cameras. Given four 2D-to-3D correspondences, we estimate camera position, orientation and recover the camera focal length. We formulate the problem and provide a minimal solution from four points by solving a system of algebraic equations. We compare the Hidden variable resultant and Gröbner basis techniques for solving the algebraic equations of our problem. By evaluating them on synthetic and on real-data, we show that the Gröbner basis technique provides stable results. [1]

1. Introduction

Determining the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections is known as the Perspective-n-Point (PnP) problem. The state of the art recognition [14, 15] and structure from motion [1, 3, 16] systems build on efficient solutions to various minimal problems [10, 18, 22, 13, 6].

In this paper we provide an efficient and robust floating point solution to the P4P problem for estimating the pose of a camera with unknown focal length from a general 3D scene. The solution to this problem has been previously known for planar scenes only [2].

PnP problems for fully calibrated cameras for three and more than three points have been extensively studied in the literature. Haralick el al. [12], Quan and Lan [19], Ameller et al. [4], Zhi and Tang [24], Reid et al. [21], Wu and Hu [23], and Quan et al.[20] developed many variations to the P4P problem from planar as well as non-planar points. An accurate non-iterative solution with the complexity scaling linearly with the number of points was presented in [11].

[1]This work has been supported by the EC project MRTN-CT-2004-005439 VISIONTRAIN, EU FP6-IST-027787 DIRAC, MSM6840770038 DMCM III, STINT Dur IG2003-2 062 and MSMT KONTAKT 9-06-17.

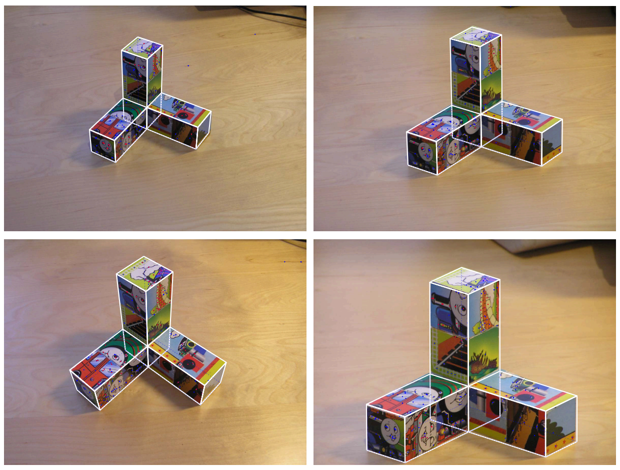
Figure 1. Camera pose and focal length are estimated automatically using four 2D-to-3D point correspondences. The accuracy of the estimate is demonstrated by predicting the projections of the edges of known 3D model of the structure.

The most relevant previous work for this paper is the technique for solving P4P problem for a perspective camera with unknown focal length from four coplanar points by Abidi and Chandra [2]. They formulated the problem using areas of triangular subdivisions of a planar quadrangle and arrived to a closed form solution. We generalize this solution to four arbitrary points in space. Our solution is more complex but completely general.

One of the important applications of our solution is in estimating the focal length, orientation, and position of a zooming camera. Unlike with other internal camera parameters, which can safely be set for most of present digital cameras a priory, the effective focal length can significantly change when using a zoom camera and thus needs to be estimated from images altogether with the camera position and orientation, Figure 1.

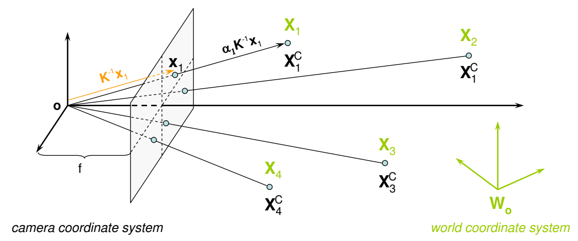

Figure 2. The camera and the world coordinate systems, image measurements ($\mathbf{x}$), ray direction vectors ($\mathtt{K}^{-1}\mathbf{x}$), and 3D points in the world coordinate system ($\mathbf{X}$) and in the camera coordinate system ($\alpha\, \mathtt{K}^{-1}\mathbf{x}_1$).

Next we provide our formulation of the problem, and design and compare two approaches to its solution.

2. Problem formulation

In this section we formulate the problem as a solution to a system of polynomial equations in four variables. Let $\{\mathbf{x}_i\}_{i=1}^4$ be image projections

$$\alpha_i\, \mathbf{x}_i = \mathtt{P}\, \mathbf{X}_i, \tag{1}$$

of four general 3D reference points, represented by their homogeneous coordinates $\{\mathbf{X}_i\}_{i=1}^4$, by a projection matrix $\mathtt{P} = \mathtt{K}\,[\mathtt{R}\,|\,\mathbf{t}]$. In our case the camera calibration matrix is known up to a focal length. Thus, we can write

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Let $\mathbf{X}_i^C = [R\,|\,t]\,\mathbf{X}_i$ be the affine coordinates of the set of the reference 3D points in the camera coordinate system. From Eq. 1, we get

$$\alpha_i\, \mathtt{K}^{-1}\mathbf{x}_i = \mathbf{X}_i^C, \tag{3}$$

and thus the direction vectors of the camera rays towards the reference points become

$$\mathbf{x}_i^C = \mathtt{K}^{-1}\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix}, \tag{4}$$

where x_i and y_i stand for the image coordinates.

Unlike in the most existing approaches for solving PnP problems, we will not search for distances from camera center to 3D point but we will solve for unknown α_i. Scale α_i stretches the camera ray $\mathbf{x}_i^C$ until it reaches 3D point $\mathbf{X}_i^C$, Figure 2. We use the fact that the transformation $[\mathtt{R}\,|\,t]$, transforming $\mathbf{X}_i$ into $\mathbf{X}_i^C$, preserves distances between 3D points. Hence, we can write

$$\|\mathbf{X}_i - \mathbf{X}_j\|^2 = \|\mathbf{X}_i^C - \mathbf{X}_j^C\|^2 = \|\alpha_i\, \mathbf{x}_i^C - \alpha_j\, \mathbf{x}_j^C\|^2, \tag{5}$$

where $\|.\|$ stands for the Euclidean norm.

Using Eq. 5 with different i and j, we can build a system of six 4^{th} degree polynomial equations in five unknowns, four unknown α_i and the unknown focal length f.

Since all α_i are nonzero and positive, we can fix one of them, e.g. α_1, and express all α_i as $\alpha_i = \alpha_1 \lambda_i$. Note that $\lambda_1 = 1$. Then, Eqn. 5 can be rewritten as

$$\begin{aligned} \|\mathbf{X}_i - \mathbf{X}_j\|^2 &= \|\alpha_1\, \lambda_i\, \mathbf{x}_i^C - \alpha_1\, \lambda_j\, \mathbf{x}_j^C\|^2 \\ &= \alpha_1^2\, \|\lambda_i\, \mathbf{x}_\mathbf{i}^\mathbf{C} - \lambda_j\, \mathbf{x}_\mathbf{j}^\mathbf{C}\|^2, \end{aligned}$$

for every $i, j \in \{1, \ldots, 4\}$. If all 3D reference points are distinct, then for each $i \neq j$ we have both the left and the right side of the equation Eq. 6 nonzero and positive. Hence we can form the follwing ratios

$$r_{ijkl} = \frac{\|\mathbf{X}_i - \mathbf{X}_j\|^2}{\|\mathbf{X}_k - \mathbf{X}_l\|^2} = \frac{\alpha_1^2\, \|\lambda_i\, \mathbf{x}_i^C - \lambda_j\, \mathbf{x}_j^C\|^2}{\alpha_1^2 \|\lambda_k\, \mathbf{x}_k^C - \lambda_l\, \mathbf{x}_l^C\|^2}. \tag{6}$$

in which α_1^2 on the right hand side cancels, thus yielding

$$r_{ijkl}\, \|\lambda_k\, \mathbf{x}_k^C - \lambda_l\, \mathbf{x}_l^C\|^2 = \|\lambda_i\, \mathbf{x}_i^C - \lambda_j\, \mathbf{x}_j^C\|^2. \tag{7}$$

By exhausting all reasonable permutations of $i, j, k, l \in \{1, \ldots, 4\}$ in Eqn. 7, we get a system of 15 polynomial equations in four unknowns $(\lambda_2, \lambda_3, \lambda_4, f)$. Note that each x_i^C contains one unknown parameter, which is the focal length f. Unknown f appears in even powers only, and thus the substitution $\varphi = f^2$ reduces the total degree of these equations to three. We observed that only five out of these 15 equations are linearly independent. Thus we get five equations in four unknowns and 20 monomials.

Once all solutions to this system of polynomial equations are found, we calculate the camera pose and orientation. For each solution of the system, we first calculate 3D positions $\mathbf{X}_i^C$ of points $\mathbf{X}_i$ in the camera coordinate system using Eqn. 3. Then we search for a rigid transformation mapping all $\mathbf{X}_i$ to $\mathbf{X}_i^C$. For this purpose we use the least-squares fitting method [5].

Now we describe how such system of polynomial equations can be solved.

3. Solving systems of polynomial equations

Our goal is to solve a system of algebraic equations

$$f_1\,(\mathbf{x}) = \ldots = f_m\,(\mathbf{x}) = 0 \tag{8}$$

which are given by a set of m polynomials $F = \{f_1, \ldots, f_m |\ f_i \in \mathbb{C}\,[x_1, \ldots, x_n]\}$ in n variables $\mathbf{x} = (x_1, \ldots, x_n)$ over the field of complex numbers.

Solving systems of algebraic polynomial equations is very challenging problem. There doesn't exist one robust, numerically stable and efficient method for solving such systems in general case. Therefore, special algorithms have to be designed for specific problems.

Two important methods for solving systems of polynomial equations are the Gröbner basis method and the Hidden variable resultant method. Now we describe these two methods and propose solvers to our problem based on these methods.

4. Hidden variable method

One of the best known resultant techniques for solving systems of polynomial equations is the hidden variable method [9]. The basic idea of this method is to consider one of variables as a parameter and eliminate other variables from the system. To illustrate how this works, consider the system (8) of m polynomial equations in n variables $\mathbf{x} = (x_1, ..., x_n)$. In this system we can regard one of the variables (for example x_1) as a parameter, i.e. hide this variable. Then, the system of polynomial equations can be rewritten in a matrix form

$$M(x_1)\,\mathbf{X} = 0, \tag{9}$$

where $M(x_1)$ is a coefficient matrix depending on the hidden variable x_1, and X is a vector of all monomials including 1 in the remaining $n-1$ variables (in this case $x_2, x_3, \ldots, x_n$).

It is known that if the number of equations equals to the number of monomials in $\mathbf{X}$ (i.e. the matrix $M(x_1)$ is square), then the system of equations has non-trivial solutions if and only if

$$det\,(M(x_1)) = 0. \tag{10}$$

The "resultant" equation (10) is a polynomial in single variable x_1 obtained by eliminating the other $n-1$ variables. This reduces the problem of solving a system of polynomial equations to solving the resultant equation (10) in x_1 and back-substituting solutions to $M(x_1)$.

It may happen that the matrix $M(x_1)$ is not square. Then we can add new equations, monomial multiples of equations (8), to this system. These new equations have same solutions as the initial polynomial equations and if we are lucky they may help us to obtain a square matrix. This method can be extended to several hidden variables [9].

5. Hidden variable solver

One way how to solve our system of polynomial equations (7) using the hidden variable method is to eliminate variables one by one. Unfortunately this often leads to a very high degree polynomial in one variable. The degree of this polynomial (defined by Equation 10) scales with the size of the matrix M. Computation times as well as numerical stability of the solver are to large extent determined by the degree of this polynomial (10). First, finding coefficients of this polynomial is equivalent to computing the determinant of the matrix M. This in some cases, including our, requires to multiply and sum numbers with different orders of magnitude. Thus "double" precision arithmetic may not be sufficient here. Second, finding the roots of a higher degree polynomial involves finding the eigenvalues of big companion matrix. This has again impact both on solver speed and numerical stability.

We therefore do not apply the resultant method directly but we apply it to a preprocessed set of equations as follows. First note that Equations 7 can be used to solve the P3P problem. For P3P it is assumed that the calibration matrix is known. So in this case the focal length variable becomes known in all equations and the system (7) has only three unknown depths $(\lambda_2, \lambda_3, \lambda_4)$. If we fix $i = 1$ and $k = 1$ in these equations (7), then we remove one variable from each equation (because $\lambda_1 = 1$). Now each equation from (7) contains only two unknown depths (λ_j, λ_l). Each of them can be easily removed using hidden variable resultant (10). This yields a polynomial which we denote as $P3P_{j,l}$. It is a polynomial in single variable λ_l and j, l are indices of the selected 2D-to-3D correspondences. By solving this polynomial one obtain solutions to the associated P3P problem.

Now return back to our original problem. One may look at this problem as on four parametric P3P problems, where the parameter is an unknown focal length f. Then, polynomial $P3P_{j,l}$ becomes a polynomial $P3P_{j,l}(f)$ in two variables, λ_l and f. Given four 2D-to-3D point correspondences, we can build a system of 6 polynomial equations $P3P_{2,3}(f), P3P_{3,2}(f), P3P_{2,4}(f), P3P_{4,2}(f), P3P_{3,4}(f), P3P_{4,3}(f)$ in four unknowns $(\lambda_2,\ \lambda_3,\ \lambda_4, f)$, where each equation contains just two unknowns. From this set of equations we can select two equations in two unknowns, for instance, $P3P_{2,3}(f)$ and $P3P_{4,3}(f)$ in unknown λ_3 and f, and solve them using the hidden variable technique. In our case we "hide" the variable f. To obtain the square matrix M we need to add 6 equations (monomials multiples of equations $P3P_{2,3}(f)$ and $P3P_{4,3}(f)$), so getting a 8×8 resultant matrix (9) for unknown focal length. Unfortunately, computing the determinant of this matrix is numerically unstable in double precision arithmetic. Therefore, we evaluate this determinant in rational arithmetic and convert its coefficients to double precision before computing the eigenvalues of M.

6. Gröbner basis method

The Gröbner basis method is another well known technique for solving systems of polynomial equations (8). The polynomials $F = \{f_1, ..., f_m |\, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ define *ideal* I, which is a set of all polynomials that can be generated as polynomial combinations of initial polynomials F

$$I = \left\{ \Sigma_{i=1}^{m} f_i \, p_i \, | \, p_i \in \mathbb{C}\left[x_1, ..., x_n\right] \right\}, \tag{11}$$

where p_i are arbitrary polynomials from $\mathbb{C}\left[x_1, ..., x_n\right]$.

We can define division by an ideal I in $\mathbb{C}\left[x_1, ..., x_n\right]$ as the division by the set F of generators of I. It is known that such multivariate polynomial division depends on the ordering of the polynomials in F and also on the monomial ordering used.

There is a special set of generators G, called a Gröbner basis of the ideal I, for which this division by the ideal I is well defined and doesn't depend on the ordering of the polynomials in G. This means that the remainder of an arbitrary polynomial $f \in \mathbb{C}\left[x_1, ..., x_n\right]$ under the division by G is uniquely determined. Furthermore, $f \in I$ if and only if the reminder of f under the division by G is zero ($\overline{f}^G = 0$). This implies that $\overline{f}^G + \overline{g}^G = \overline{f+g}^G$ and $\overline{\overline{f}^G . \overline{g}^G}^G = \overline{f.g}^G$

Thanks to these properties of the Gröbner basis G, we can consider the space of all possible reminders under the division by I. This space is know as a *quotient ring* and we will denote it as $A = \mathbb{C}\left[x_1, ..., x_n\right]/I$. It is known that if I is a radical ideal [9] and the set of equations F has a finite number of solutions N, then A is a finite dimensional space with $dim(A) = N$. Now we can use nice properties of a special matrix defined in this space, to find solutions to our system of equations (8).

Consider the multiplication by some polynomial $p \in \mathbb{C}\left[x_1, ..., x_n\right]$ in the quotient ring A. This multiplication defines a linear mapping T_p from A to itself. Since A is a finite-dimensional vector space over $\mathbb{C}$, we can represent this mapping by its matrix m_p with respect to some basis B of A. This matrix is known as an *action matrix* and can be viewed as a generalization of a companion matrix used in solving one polynomial equation in one unknown. It is because solutions to our system of polynomial equations (8) can be easily obtained from the eigenvalues and eigenvectors of this action matrix. The values of p on our solutions are exactly the eigenvalues of this matrix [8].

Now we use an important observation made in [13]. This observation tells, that we can construct the action matrix without computing the complete Gröbner basis G. All we need is to construct polynomials from the ideal I with leading monomials from the set $p \cdot B \setminus B$ and the remaining monomials from B. For more details about the form of these polynomials see [13].

Since these polynomials are from the ideal I, we can generate them as algebraic combinations of the initial generators F. This can be done using several methods. One possible way is to start with F and then systematically generate new polynomials from I by multiplying already generated polynomials by individual variables and reducing them each time by the Gauss-Jordan (G-J) elimination. This method was, for example, used in [13] and resulted in several G-J eliminations. Another possible way is to generate all new polynomials in one step by multiplying polynomials from F with selected monomials and reducing all generated polynomials at once using one G-J elimination. This method was used in [6] and proved to be numerically more stable. Therefore, we use this method to construct the action matrix for our problem and in this way solve the problem.

Here we extend this method by reducing the number of generated polynomials. This is done in a simple and intuitive way. Imagine that we have a set of monomials which should be used for multiplying initial polynomials F and in this way generating new polynomials. In most cases, this set of monomials contains all monomials up to some degree. All generated polynomials including initial polynomial equations F can be written in a matrix form

$$\mathtt{M}\, X = 0, \tag{12}$$

where $\mathtt{M}$ is the coefficient matrix and X is the vector of all monomials. Consider that we know that after G-J elimination of this matrix $\mathtt{M}$ we obtain all polynomials that we need for constructing the action matrix. Since we know how these necessary polynomials should look (i.e. which monomials they contain), we can systematically reduce the number of generated polynomials in the following way:

1. For all rows from $\mathtt{M}$ starting with the last row r (with the polynomial of highest degree) do
   (a) Perform G-J elimination on the matrix $\mathtt{M}$ without the row r
   (b) If the eliminated matrix contains all necessary polynomials $\mathtt{M} := \mathtt{M} \setminus r$
   (c) Go to step 1

This elimination procedure is performed only once in the offline process. The online solver works with the eliminated set of polynomials which speeds up the process of creating the action matrix and also improves the numerical stability.

7. Gröbner basis solver

The Gröbner basis solver starts with 15 equations (7) in four unknowns $\lambda_2, \lambda_3, \lambda_4$ and $\varphi = f^2$. Only five from these 15 equations are linearly independent so in fact we have 5 equations in 4 unknowns. This is an overconstrained system of polynomial equations which can be solved in several ways. We have decided to use four from these five equations to find solutions to the four unknowns and use the fifth equation to test for possible degeneracy and to select the best root.

We use the last four equations which we obtain after G-J elimination of the initial 15 equations. The reason for this is that the system of the first four equations has 14 solutions and results in more complicated computations of the action matrix. On the other hand, the system of the last four equations has only 10 solutions and the action matrix can be obtained easier.

To create the action matrix, we use the method described in Section 6. This method assume that the basis B of A is known. This assumption can be made because this problem like many other in computer vision has the convenient property that the monomials which appear in the initial system of polynomial equations are always same irrespectively from the concrete coefficients arising from non-degenerate image measurements. Therefore, the leading monomials of the corresponding Gröbner basis, and thus the monomials in the basis B are generally the same. So they can be found once in advance.

To compute B, we solve our problem in a random chosen finite prime field $\mathbb{Z}_p$ where exact arithmetic can be used and numbers can be represented in a simple and efficient way. It speeds up computations, minimizes memory requirements and especially avoids numerical instability.

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.

Using Macaulay2, we have found that our problem has 10 solutions and the basis $B = (\lambda_3^2, \varphi\lambda_4, \lambda_2\lambda_4, \lambda_3\lambda_4, \lambda_4^2, \varphi, \lambda_2, \lambda_3, \lambda_4, 1)$ w.r.t. the used graded reverse lexicographic ordering $\varphi > \lambda_2 > \lambda_3 > \lambda_4$. Once this is known, the action matrix for floating point coefficients can be created.

We construct the action matrix m_{λ_2} for multiplication by λ_2 because the creation of the action matrix for φ requires to generate polynomials of higher degree.

The method described in Section 6 calls for generating polynomials with leading monomials from the set $\lambda_2 \cdot B \setminus B$ and the remaining monomials from B. To generate these polynomials from the ideal, we use the method described in Section 6 in which these polynomials are generated in one step by multiplying initial four polynomial equations with selected monomials and reducing all generated polynomials at once using one G-J elimination.

The set of monomials which should be used in this process for multiplying initial polynomial equations can be found once in advance. To do this we again use Macaulay 2. We have found that to obtain all necessary polynomials for crating the action matrix we need to generate all monomial multiples of the initial four polynomial equations up to the total degree seven. This means that we need to multiply our four 3^{rd} degree polynomial equations with all monomials up to the degree four.

In this way we generate 276 new polynomials which, together with the initial four polynomial equations, form a system of 280 polynomials in 283 monomials. Only 243 from these polynomials are linearly independent. So, in the next step we select only 243 linearly independent polynomial equations. Then we remove all unnecessary polynomials by the procedure described in Section 6 and obtain 154 polynomial equations in 180 monomials. These polynomial equations can be written in the matrix form (12). After the G-J elimination of the coeficient matrix $\mathtt{M}$ we obtain all polynomials with leading monomials from the set $\lambda_2 \cdot B \setminus B$ and the remaining monomials from B which we need for constructing the action matrix m_{λ_2}.

Note that all steps until now (computation in Macaulay 2, finding the basis B and finding the polynomials form the ideal which should be generated to obtain all necessary polynomials for constructing the action matrix) were done only once by us to study this problem and provide the solution to it. The online solver consists only from one G-J elimination of the 154×180 coefficient matrix $\mathtt{M}$. This matrix contains coefficients which arise from concrete image measurements. After the G-J elimination of this matrix, and using its rows which correspond to the polynomials with leading monomials from the set $\lambda_2 \cdot B \setminus B$, the action matrix m_{λ_2} can be created. The solutions to four unknowns $\lambda_2, \lambda_3, \lambda_4$ and $\varphi = f^2$ can be found using eigenvectors of this action matrix.

8. Experiments

In this section we evaluate both hidden variable and Gröbner basis solutions of the problem.

8.1. Synthetic data set

We study the performance of the methods on synthetically generated ground-truth planar and general 3D scenes. These scenes were generated randomly with the Gaussian distributions on a plane or in a 3D cube depending on the testing configuration. Each 3D point was projected by a camera, where the camera orientation and position were selected randomly. Then, Gaussian noise with standard deviation σ was added to each image point.

8.1.1 Noise free data set

The first experiment shows behavior of both Gröbner basis and hidden variable solvers on the exact measurements. For each of ten focal lengths $f_{gt} = \{20, 50, 80, 100, 140, 170, 200, 230, 260, 290\}$ millimetres, we have generated 100 random scenes and cameras poses. We have estimated focal lengths using both our algorithms and compared them with the ground truth by evaluating the logarithm of the relative error

$$\epsilon = \log_{10}(|\frac{f - f_{gt}}{f_{gt}}|). \tag{13}$$

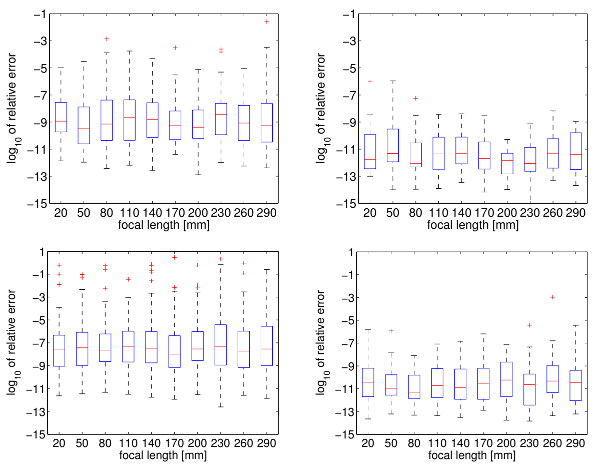

Figure 3. Performance evaluation without noise on synthetic 3D scenes. Results for (Top) general 3D scenes and (Bottom) planar scenes computed using (Left) Gröbner basis solver and (Right) hidden variable solver.

Values of ϵ for general non-planar 3D scenes are shown in Figure 3 (Top). Plots are displayed using the Matlab function *boxplot* which shows a blue box from the 25% to the 75% quantile, the red horizontal line at the median. The red crosses show data beyond 1.5 times the interquartile range. Results for planar scenes are shown in Figure 3 (Bottom).

We see that the hidden variable solver is more accurate. This is because hidden variable solver uses more accurate rational arithmetic while Gröbner basis solver uses standard "double" precision arithmetic. The Gröbner basis solver is less accurate but provided results are far more accurate than required in any real situation.

We have already mentioned that computing coefficients of the resultant polynomial in hidden variable method is not stable using double precision arithmetic. This is because coefficients of the matrix which is used to calculate the resultant polynomial contains numbers with large difference in exponents. Then, during calculations of the determinant, these numbers multiply together to numbers with exponents bordering ± 300. This happened for every chosen focal length and every scene configuration we have tested.

8.1.2 Data affected by a noise

Boxplots in Figure 4 (Top) show behaviour of the solvers for one fixed focal length $f = 60mm$, a general non-planar 3D scene and different levels of noise added to image projections. Results for planar scenes are shown in Figure 4 (Bottom).

Here, both algorithms return comparable results. As expected, with growing noise level both algorithms return focal length more and more deviated from the ground truth value. However, even for relative large noise of eight pixels, the median of estimated focal lengths is still close to the ground truth value.

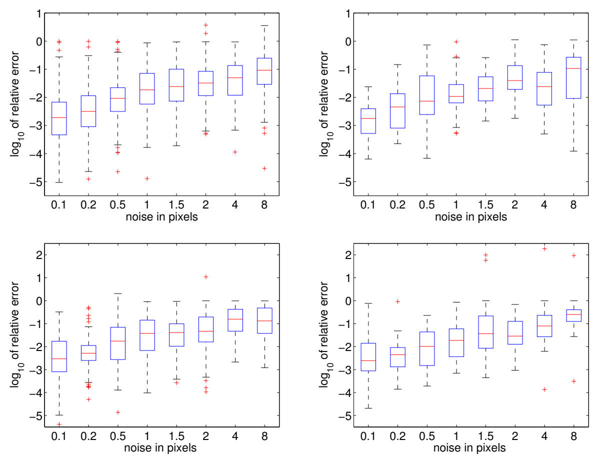

Figure 4. Results for fixed focal length at 60mm and different levels of noise. Results for (Top) general 3D scenes and (Bottom) planar scenes computed using (Left) Gröbner basis solver and (Right) hidden variable solver.

8.1.3 Number of solutions

By solving the system of polynomial equations (7) using Gröbner basis solver we obtain ten roots. Hidden variable solver returns 32 candidates for the focal length. In this subsection we discuss how many of them usually lead to meaningful results.

Gröbner basis solver

First, we work in real coordinate system and thus we can drop all complex roots. Also all negative focal lengths and focal lengths greater than $10^6 mm$ can be ignored. After removing them we get only 2.5 solutions in average (measured on 1000 runs using synthetic data sets). Moreover, not only focal lengths but also all negative roots can be bypassed because all λ_i should be positive too. Then we get 1.90 solutions per algorithm run in average.

Second, as we have already mentioned in Section 7, our Gröbner basis solver uses only four out of five equations to solve the problem. Thus we have one more equation, which can be used to filter incorrect roots. They can be filtered as follows. Let eq_i be the i-th equation where eq_1 is the equation which is omitted in the solver. Then we say that solution is correct if

$$|eq_1(\lambda_2, \lambda_3, \lambda_4, f)| < \delta \max |eq_i(\lambda_2, \lambda_3, \lambda_4, f)|, \quad (14)$$

where δ is conveniently chosen value and by $eq_i(\lambda_2, \lambda_3, \lambda_4, f)$ we mean evaluation of the polynomial eq_i using estimated roots $\lambda_2, \lambda_3, \lambda_4, f$. Note that

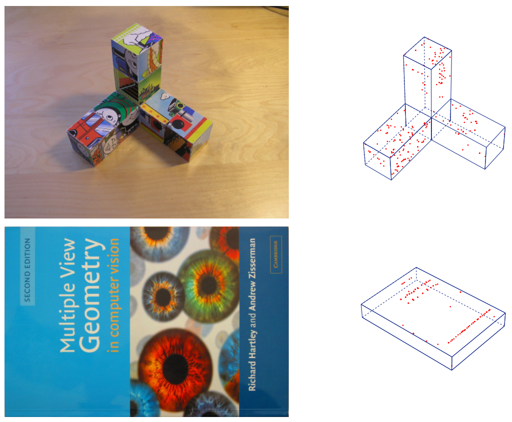

Figure 5. Real data set models. 3D models (Right) of real objects were created manually and image textures (Left) were mapped on the models. Red dots show feature points detected in the textures.

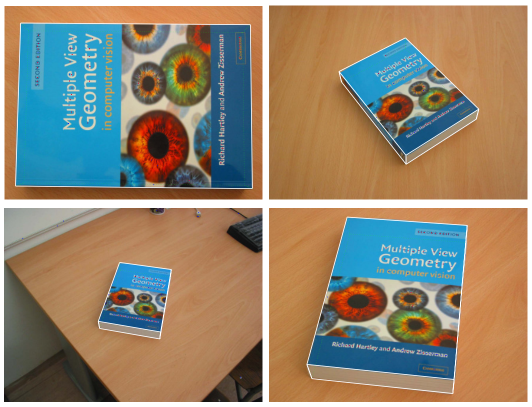
Figure 6. Camera position, orientation and focal length were estimated automatically using 2D-to-3D point correspondences on planar scene.

all five equations have the same total degree and all polynomials were normalised by highest degree monomial.

Using this filter even with relatively large $\delta = 10^8$ we can remove about 23% of the remaining solutions. None of the removed solutions was correct (with respect to the ground truth camera). This number raised to 37% in the experiments with 30% outliers.

Hidden variable solver

In the case of hidden variable solver we have to back-substitute focal lengths to $P3P_{2,3}(f)$ and $P3P_{4,3}(f)$ first. By solving these two P3P problems we obtain a set of candidate solutions. Then we can filter them as described in the previous section.

8.1.4 Solvers speed

We have implemented both solvers in Matlab. For Gröbner basis solver it takes about $233ms$ in average to find a solution. Hidden variable solver is about $20 - 80\times$ slower due to rational arithmetic depending on the required length of the number used in the computations.

8.2. Real data set

For the real data experiments we used both planar and non-planar objects (see Figure 5). In the case of the non-planar object, we created its 3D model, detected MSER [17] image features on all its textures and calculated their 3D position. For the experiment on the planar scene, we used the picture of the front side of the book and detected image features on it. Then, we transferred them to 3D by setting $z = 0$ and scaling them to the size obtained by measuring the book. We captured several images for both testing objects and used focal lengths stored in image header (*jpeg-exif*) as ground truth.

Using MSER detector, we found image features in the each image. Tentative correspondences were chosen as mutually best matches between the image and the 3D model features. We used locally optimised RANSAC [7] with our solver to find the camera parameters. In the camera calibration matrix we put principal point to the middle of the image and set skew to zero. Then, the camera calibration matrix became

$$K = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (15)$$

where focal length f was estimated using our solver.

Due to long computation times in real experiments, we used only the Gröbner basis solver in RANSAC to find correct matches since both proposed solvers are equivalent for this purpose. To evaluate the accuracy and stability of the solvers, we computed camera parameters by both of them from the correct matches.

Figure 7 (Left) and Figure 6 show results for the experiment with a planar object. Differences between focal lengths estimated by the hidden variable and the Gröbner basis solvers are smaller than 0.1mm (wrt. 35mm film). Logarithm of the relative error (13) to exif focal length is around -1.4.

In the experiments with the general non-planar scene, we have captured several triplets of images such that the camera zoom settings were same for images in each triplet but different for different triplets. We used autofocus camera and therefore the focal lengths of one zoom setting might slightly differ due to camera focussing. Comparison of both solvers with focal length extracted from (*jpeg-exif*) are shown in Figure 7 (Right). Unlike in the planar test scene, where we matched the reference image with each captured image, here we matched the model textures with images. Hence the quality of the result is affected also by the quality

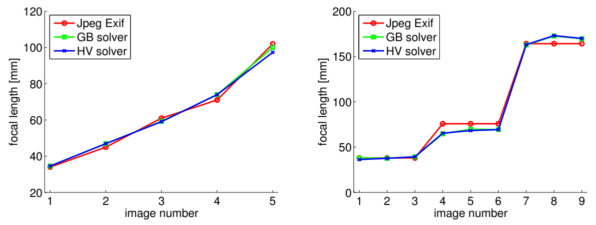

Figure 7. Focal lengths estimated by the hidden variable solver (HV) and the Gröbner basis solver (GB) are very close to "exif" focal lengths. Planar scene (Left), general non-planar scene (Right).

of the reference 3D models and by resolution of the textures. Nevertheless, as shown in Figure 7 (Right), the results are satisfactory and visually plausible (see Figure 1).

We tried to further improve the quality of the results by minimizing the image reprojection error

$$\gamma = \sum_i \|P(\mathbf{X}_i) - \mathbf{x}_i\| \tag{16}$$

where $\mathbf{X}_i$ are 3D points from the inlier set returned by RANSAC, $\mathbf{x}_i$ are the corresponding image points, and $P(\mathbf{X}_i)$ represents the projection of the 3D point $\mathbf{X}_i$ to the image plane using camera matrix $\mathtt{P}$. The camera matrix $\mathtt{P} = \mathtt{K}[\mathtt{R}|\mathbf{t}]$, with calibration matrix as in (2), rotation matrix $\mathtt{R}$ and translation vector $\mathbf{t}$.

We used the Levenberg-Marquardt algorithm to optimize image reprojection error (16). To initialize optimization we used results from both our solvers. Focal lengths resulting from the optimization were very close to the initial values provided by the solvers. They differed in less than 0.1%. Reprojection errors were improved by 0.3 pixels in average.

9. Conclusion

In this paper we presented a general solution to the previously unsolved problem of pose and orientation determination of a perspective camera with unknown focal length given images of four 3D reference points. We formulated the problem as a system of polynomial equations and solved them using hidden variable method and Gröbner basis based method. Both solutions are general and can be used in both cases where all points lay on the plane as well as in a non-planar configuration. Experiments have shown that both our solvers behave well both on synthetic and real data-sets, however the Gröbner basis method provides stable results.

References

[1] 2D3. Boujou. www.2d3.com.

[2] M. a. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE PAMI*, 17(5):534–538, 1995.

[3] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewéius, R. Yang, G. Welch, H. Towles, D. Nistér, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT*, May 2006.

[4] M.-A. Ameller, M. Quan, and L. Triggs. Camera pose revisited – new linear algorithms. In *ECCV 2000*.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE PAMI*, 9(5), 1987.

[6] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of grbner basis polynomial equation solver. In *ICCV*, 2007.

[7] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Proc DAGM*, pages 236–243, 2003.

[8] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry, Second edition*, volume 185. Springer Verlag, Berlin - Heidelberg - New York, 2005.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer Verlag, 2007.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–95, 1981.

[11] F.Moreno-Noguer, V.Lepetit, and P.Fua. Accurate non-iterative o(n) solution to the pnp problems. In *ICCV*, 2007.

[12] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim. Pose estimation from corresponding point data. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(6):1426–1446, 1989.

[13] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *CVPR*, 2007.

[14] B. Leibe, N. Cornelis, K. Cornelis, and L. J. V. Gool. Dynamic 3d scene analysis from a moving vehicle. *CVPR* 2007.

[15] B. Leibe, K. Schindler, and L. J. V. Gool. Coupled detection and trajectory estimation for multi-object. In *ICCV*, 2007.

[16] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *CVPR*, 2007.

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

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

[19] L. Quan and Z.-D. Lan. Linear n-point camera pose determination. *IEEE PAMI*, 21(8):774–780, August 1999.

[20] L. Quan, B. Triggs, and B. Mourrain. Some results on minimal euclidean reconstruction from four points. *Journal of Mathematical Imaging and Vision*, 24(3):341–348, 2006.

[21] G. Reid, J. Tang, and L. Zhi. A complete symbolic-numeric linear method for camera pose determination. In *ISSAC 2003*, pages 215–223, New York, NY, USA, 2003. ACM.

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

[23] Y. Wu and Z. Hu. Pnp problem revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, January 2006.

[24] L. Zhi and J. Tang. A complete linear 4-point algorithm for camera pose determination. *AMSS, Academia Sinica*, (21), December 2002.