Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Algebraic Methods in Computer Vision

Doctoral Thesis

Zuzana Kúkelová

Prague, February 2013

Ph.D. Programme: Electrical Engineering and Information Technology, P2612

Branch of study: Mathematical Engineering, 3901V02

Supervisor: Ing. Tomáš Pajdla, Ph.D.

Algebraic Methods in Computer Vision

A Dissertation Presented to the Faculty of the Electrical Engineering of the Czech Technical University in Prague in Partial Fulfillment of the Requirements for the Ph.D. Degree in Study Programme No. P2612 - Electrotechnics and Informatics, branch No. 3901V021 - Mathematical Engineering, by

Zuzana Kúkelová

Prague, February 2013

Supervisor

Ing. Tomáš Pajdla, Ph.D.

Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
fax: +420 224 357 385, phone: +420 224 357 465
http://cmp.felk.cvut.cz

Abstract

Many problems in computer vision require efficient solvers for solving systems of non-linear polynomial equations. For instance relative and absolute camera pose computations and estimation of the camera lens distortion are examples of problems that can be formulated as minimal problems, i.e. they can be solved from a minimal number of input data and lead to solving systems of polynomial equations with a finite number of solutions.

Often, polynomial systems arising from minimal problems are not trivial and general algorithms for solving systems of polynomial equations are not efficient for them. Therefore, special algorithms have to be designed to achieve numerical robustness and computational efficiency.

In this thesis we review two general algebraic methods for solving systems of polynomial equations, the Gröbner basis and the resultant based methods, and suggest their modifications, which are suitable for many computer vision problems.

The main difference between the modified methods and the general methods is that the modified methods use the structure of the system of polynomial equations representing a particular problem to design an efficient specific solver for this problem.

These modified methods consist of two phases. In the first phase, preprocessing and computations common to all considered instances of the given problem are performed and an efficient specific solver is constructed. For a particular problem this phase needs to be performed only once. In the second phase, the specific solver is used to efficiently solve concrete instances of the particular problem. This efficient specific solver is not general and solves only systems of polynomial equations of one form. However, it is faster than a general solver and suitable for applications that appear in computer vision and robotics.

Construction of efficient specific solvers can be easily automated and therefore used even by non-experts to solve technical problems leading to systems of polynomial equations. In this thesis we propose an automatic generator of such efficient specific solvers based on the modified Gröbner basis method.

We demonstrate the usefulness of our approach by providing new, efficient and numerical stable solutions to several important relative pose problems, most of them previously unsolved. These problems include estimating relative pose and internal parameters of calibrated, partially calibrated (with unknown focal length), or completely uncalibrated perspective or radially distorted cameras observing general scenes or scenes with dominant plane. All these problems can be efficiently used in many applications such as camera localization, structure-from-motion, scene reconstruction, tracking and recognition. The quality of all presented solvers is demonstrated on synthetic and real data.

Acknowledgments

I would like to express my thanks to my colleagues at CMP who I had the pleasure of working with, especially to Martin Bujňák for his ideas and collaborative efforts in a large part of my work. I am greatly indebted to my advisor Tomáš Pajdla for guiding me throughout my research. Their friendly support, patience and theoretical and practical help have been paramount to the successful completion of my PhD study.

Furthermore, I would like to thank Martin Byröd, Klas Josephson and Kalle Åström at the Mathematical Imaging Group in Lund for interesting discussions, useful comments and collaboration on two papers.

Finally, I would like to thank my family and my friends for all their support that made it possible for me to finish this thesis.

I gratefully acknowledge EC projects FP6-IST-027787 DIRAC, FP7-SPA-218814 PRoVisG and FP7-SPACE-241523 PRoViScout which supported my research.

Contents

Keywords: Gröbner basis, resultants, polynomial eigenvalue problems, minimal problems, relative pose problems, radial distortion.

Table of Definitions

Table of Examples

List of Algorithms

Notation

h	scalar value
$\mathbf{h}$	column vector
$\mathtt{H}$	matrix
H	set
$G \subset H$	G is subset of H
$G \setminus H$	set difference
$G \cup H$	set union
$G \cap H$	set intersection
$\|H\|$	cardinality of H
$h \in H$	h is an element of the set H
$\mathbb{C}$	set of complex numbers
$\mathbb{R}$	set of real numbers
$\mathbb{Q}$	set of rational numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}_p$	finite prime field $\mathbb{Z}/p$
$M_{m \times s}(\mathbb{C})$	vector space of all matrices of size $m \times s$ with elements from $\mathbb{C}$
$f(x_1, \ldots, x_n)$	polynomial in variables $x_1, \ldots, x_n$
$\mathbb{C}[x_1, \ldots, x_n]$	set of all polynomials in variables $x_1, \ldots, x_n$ with coefficients from $\mathbb{C}$
$\mathbf{x}^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$	monomial in variables $x_1, \ldots, x_n$
$\mathbf{0} = (0 \ldots 0)^\top$	vector of zeros ($\mathbf{0}_m$ denotes $m \times 1$ vector of zeros)
$\mathbf{1} = (1 \ldots 1)^\top$	vector of ones ($\mathbf{1}_m$ denotes $m \times 1$ vector of ones)
$\mathtt{0} = (\mathbf{0} \ldots \mathbf{0})$	matrix of zeros
$\mathtt{1} = (\mathbf{1} \ldots \mathbf{1})$	matrix of ones
$[\mathbf{h}]_\times = \begin{pmatrix} 0 & -h_3 & h_2 \\ h_3 & 0 & -h_1 \\ -h_2 & h_1 & 0 \end{pmatrix}$	cross product matrix of vector $\mathbf{h}$
$\det(\mathtt{H})$	determinant of matrix $\mathtt{H}$
$\mathrm{trace}(\mathtt{H})$	trace of matrix $\mathtt{H}$

Commonly used symbols and abbreviations

G-J	Gauss-Jordan (elimination)
GB	Gröbner basis
LT	leading term of a polynomial
LC	leading coefficient of a polynomial
LM	leading monomial of a polynomial
lex	lexicographic ordering of monomials
grevlex	graded reverse lexicographic ordering of monomials
PEP	polynomial eigenvalue problem
GEP	generalized eigenvalue problem
QEP	quadratic eigenvalue problem
SfM	Structure from Motion
RANSAC	Random Sample Consensus
EXIF	Exchangeable image file format
$x_1, \ldots, x_n$	variables
$f_1, \ldots, f_m$	polynomials
I	ideal
$\sqrt{I}$	radical ideal
$V(f_1, \ldots, f_m)$	affine variety defined by $f_1, \ldots, f_m$
$\succ$	monomial ordering
F	set of polynomials $F = \{f_1, \ldots, f_m\}$
$\mathtt{F} = \Psi_{T_\succ(F)}(F)$	matrix representation of a set of polynomials F
$\overline{\mathtt{F}}$	matrix F extended by some rows corresponding to new polynomials added to F
$\widetilde{\mathtt{F}}$	matrix F eliminated by G-J elimination
$\overline{f}^F$	remainder of a polynomial f on division by F
G	Gröbner basis
A	quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$
$[f]$	coset, $[f] = f + I = \{f + h, h \in I\}$
$Res_{d_0,\ldots,d_n}(F_0, \ldots, F_n)$	resultant of polynomials $F_0, \ldots, F_n$ with degrees $d_0, \ldots, d_n$

1 Introduction

"Equations are more important to me,
because politics is for the present,
but an equation is something for eternity."

Albert Einstein

Many important problems in computer vision, as well as in other fields, can be formulated using systems of polynomial equations. Examples of problems that require solving complex systems of non-linear polynomial equations are problems of estimating camera geometry, such as relative or absolute camera pose problems. These problems have a broad range of applications, e.g., in structure-from-motion and 3D reconstruction [2, 68, 123, 125] (see Figure 1.1), recognition [94, 95], video-based rendering [10], robotics, and augmented reality.

The problem of solving systems of non-linear polynomial equations is a very old problem with many different well-studied solution methods. In general, the methods can be divided into numerical and algebraic methods. In this thesis we focus on the algebraic methods. We review two classes of standard algebraic methods, i.e. the Gröbner basis and the resultant based methods. These methods are very useful mathematical methods but since they are general, i.e. they were developed for general systems of polynomial equations, they are usually not suitable for efficiently solving systems which appear in computer vision problems.

It is because problems like estimating relative or absolute pose of a camera are usually parts of some large systems, e.g., structure-from-motion pipelines or recognition systems, which require high or even real-time performance. Moreover, the input measurements used for the computation are contaminated with a large number of outliers (incorrect inputs) most of the time. Therefore, these problems have to be solved for many different inputs to find the "best solution", i.e. the solution consistent with as many measurements as possible, e.g., when using in RANSAC [50] based algorithms.

This means that computer vision problems usually require very fast, efficient, and numerically stable solvers which are able to solve many instances of a particular problem, i.e. many system of polynomial equations of "one form" only with different "non-degenerate" coefficients,in milliseconds. Unfortunately, this requirement usually cannot be fulfilled by using standard general methods for solving systems of polynomial equations.

Therefore, in recent years various specific algorithms based on algebraic geometry concepts have been proposed to achieve numerical robustness and computational efficiency when solving computer vision problems. The main property of these algorithms is that they use a specific

Figure 1.1: (Left) Sample images from a set comprising 57 images of lion statue. (Center) External camera calibration and a sparse 3D point cloud obtained using method [125]. (Right) 3D surface reconstruction computed by method [69] once external camera calibration was known.

structure of a system of polynomial equations arising from a particular problem to efficiently solve this problem only.

In other words, if we know that our problem always results in a system of three equations in three unknowns representing, for example, the intersection of a sphere, an ellipsoid and a hyperboloid, then we do not need to use general algorithms, such as Buchberger's algorithm for computing Gröbner bases [39]. We can design a specific algorithm which takes the structure of these three equations into account and solves efficiently only this system with arbitrary "non-degenerate" coefficients. Such a specific algorithm may be more efficient and significantly faster than a general algorithm.

Recently, many efficient specific algorithms for solving various computer vision problems have been proposed [112, 134, 131, 135, 136, 132, 114, 33, 98, 96, 33, 34, 32, 70]. They are mostly based on standard algebraic methods and designed manually for a particular problem. This manual design usually requires deeper knowledge of algebraic geometry and considerable amount of craft, which makes the process of generation of these specific solvers complex and virtually impenetrable for a non-specialist. Moreover, for some of these problems it is not clear how their solvers were created and therefore non-specialists often use them as black boxes; they are not able to reimplement them, improve them, or create similar solvers for their own new problems.

In this thesis we suggest modifications of two standard algebraic techniques for solving systems of polynomial equations; the Gröbner basis and the resultant based technique. These modifications enable the construction efficient specific solvers for particular problems, i.e. particular systems of polynomial equations. Construction of efficient specific solvers can be easily automated and therefore used even by non-experts to solve technical problems resulting in systems of polynomial equations.

We propose an automatic generator of efficient specific solvers based on the presented modified Gröbner basis method. This automatic generator can be used to create solvers for new or existing problems and usually results in smaller, more efficient and more stable solvers than manually created ones.

We demonstrate the usefulness of both modified methods and the usefulness of our automatic generator by providing new efficient and numerical stable solutions to several important relative pose problems, most of them previously unsolved. The quality of all solvers is demonstrated on synthetic and real data.

2 Contribution of the thesis

"Each man is capable of doing one thing well.
If he attempts several, he will fail to achieve distinction in any."

Plato

This thesis focuses on algebraic methods for solving systems of polynomial equations appearing especially in computer vision problems. Its main goal is to improve algebraic based methods previously used to manually create efficient solvers for some computer vision problems, automate these algebraic methods, and use them to efficiently solve previously unsolved minimal computer vision problems.

The content of this thesis is based on the material published in the following papers:

Main papers

Impacted journal articles

[86] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, July 2012.

[92] Z. Kukelova and T. Pajdla. A minimal solution to radial distortion autocalibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2410–2422, December 2011.

[87] Z. Kukelova, M. Byröd, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. *Computer Vision and Image Understanding*, 114(2):234–244, February 2010.

Conference papers

[138] A. Torii, Z. Kukelova, M. Bujnak, and T. Pajdla. The six point algorithm revisited. In *10th Asian Conference on Computer Vision (ACCV'10 Workshop)*, volume 6469 of *Lecture Notes in Computer Science*, pages 184–193, 2011

[26] M. Bujnak, Z. Kukelova, and T. Pajdla. 3D reconstruction from image collections with a single known focal length. In *IEEE International Conference on Computer Vision (ICCV'09)*, pages 1803–1810, 2009.

[35] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), Vols 1-12*, pages 234–244, 2008.

[84] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In *British Machine Vision Conference (BMVC'08)*, 2008.

[83] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic Generator of Minimal Problem Solvers. In *10th European Conference on Computer Vision (ECCV'08)*, volume 5304 of *Lecture Notes in Computer Science*, pages 302–315, 2008.

[91] Z. Kukelova and T. Pajdla. Two minimal problems for cameras with radial distortion. In *7th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS'07)*, 2007.

[89] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[90] Z. Kukelova and T. Pajdla. Solving polynomial equations for minimal problems in computer vision. In *Computer Vision Winter Workshop (CVWW'07)*, Graz, Austria, 2007.

Additional papers

Peer-reviewed journal articles

[29] M. Bujnak, Z. Kukelova, and T. Pajdla. Efficient solutions to the absolute pose of cameras with unknown focal length and radial distortion by decomposition to planar and non-planar cases. *IPSJ Transaction on Computer vision and Application (CVA)*, 4:78–86, May 2012.

Conference papers

[88] Z. Kukelova, J. Heller and T. Pajdla. Hand-Eye Calibration without Hand Orientation Measurement Using Minimal Solution. In *11th Asian Conference on Computer Vision (ACCV'12)*, 2012.

[30] M. Bujnak, Z. Kukelova, and T. Pajdla. Making Minimal Solvers Fast. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*, 2012.

[28] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *10th Asian Conference on Computer Vision (ACCV'10)*, volume 6492 of *Lecture Notes in Computer Science*, pages 11–24, 2011.

[85] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *10th Asian Conference on Computer Vision (ACCV'10)*, volume 6493 of *Lecture Notes in Computer Science*, pages 216–229, 2011.

[27] M. Bujnak, Z. Kukelova, and T. Pajdla. Robust focal length estimation by voting in multi-view scene reconstruction. In *9th Asian Conference on Computer Vision (ACCV'09)*, pages 13–24, 2009.

[25] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), Vols 1-12*, pages 3506–3513, 2008.

Contribution

The contribution of the proposed thesis can be divided into three main groups:

1. Algebraic methods for solving systems of polynomial equations [89, 92, 86].
2. Automatic generator of Gröbner basis solvers [83].
3. Solutions to minimal problems in computer vision [90, 89, 91, 84, 87, 92, 26, 35, 138].

Next we describe contributions in these groups in more detail.

2.1 Algebraic methods for solving systems of polynomial equations

The first group of contributions of this thesis consists of modifications of two standard algebraic techniques for solving systems of polynomial equations; the Gröbner basis and the resultant based technique. These modifications enable the creation of efficient specific solvers for particular problems, i.e. particular systems of polynomial equations.

The main difference between the proposed specialized methods and the general methods is that the specialized methods use the structure of the system of polynomial equations representing a particular problem to design an efficient specific solver for this problem.

These specialized methods consist of two phases. In the first phase, preprocessing and computations common to all considered instances of the given problem are performed and an efficient specific solver is constructed. For a particular problem this phase needs to be performed only once, therefore, we will call it the "offline phase".

In the second "online phase", the efficient specific solver is used to solve concrete instances of the particular problem. This specific solver is not general and solves only systems of polynomial equations of one form, i.e. systems which coefficients always generate the same "path" to the solution. However, the specific solver is faster than a general solver and suitable for applications which appear in computer vision.

In the case of the specialized Gröbner basis method presented in Section 4.4, we summarize and extend the Gröbner basis method used to manually create solvers for some previously solved computer vision problems [131, 132, 135, 134]. Thanks to the proposed extensions, e.g., the identification of the form of necessary polynomials, new strategies for generating these polynomials, and new procedures for removing unnecessary polynomials, we are able

to create smaller, more efficient, and more stable solvers than the previously manually created ones [131, 132, 135, 134]. Moreover, all these extensions can be easily automated and therefore the presented specialized Gröbner basis method can be used even by non-experts to solve technical problems leading to systems of polynomial equations.

The second proposed specialized method described in Section 5.5 is based on the hidden variable resultants and the polynomial eigenvalue problems [9]. In this thesis we propose several strategies for transforming the initial system of polynomial equations to polynomial eigenvalue problem [9] and for reducing the size of this problem. In this way we are able to create efficient and numerically stable specific solvers for many problems appearing in computer vision. Again, this method can be easily automated.

2.2 Automatic generator of Gröbner basis solvers

Since the presented specialized Gröbner basis method requires non-trivial knowledge of algebraic geometry from its user, we propose an automatic generator of Gröbner basis solvers which could be used even by non-experts to easily solve problems leading to systems of polynomial equations. The input to our solver generator is a system of polynomial equations with a finite number of solutions. The output of our solver generator is a Matlab code that computes solutions of this system for arbitrary "non-degenerate" coefficients. Generating solvers automatically opens possibilities for solving more complicated problems that could not be handled manually or solving existing problems in a better and more efficient way. We demonstrate that our automatic generator constructs efficient and numerically stable solvers that are comparable or better than known manually constructed solvers in terms of computational time, space requirements, and numerical stability.

2.3 Solutions to minimal problems in computer vision

To demonstrate the usefulness of the proposed specialized techniques and the usefulness of our automatic generator of Gröbner basis solvers, we provide efficient solutions to several important relative pose problems. In this thesis we describe solutions to five minimal problems which haven't been solved before:

- the problem of simultaneous estimation of the fundamental matrix and a common radial distortion parameter for two uncalibrated cameras from eight image point correspondences,
- the problem of simultaneous estimation of the essential matrix and a common radial distortion parameter for two partially calibrated cameras and six image point correspondences,
- the problem of simultaneous estimation of the fundamental matrix and radial distortion parameters for two uncalibrated cameras with different radial distortions from nine image point correspondences,

- the 6-point relative pose problem for one fully calibrated and one up to focal length calibrated camera, and
- the problem of estimating epipolar geometry and unknown focal length from images of four points lying on a plane and one off-the-plane point, i.e. the "plane + parallax" problem for cameras with unknown focal length.

Our solutions to these five new minimal problems are based on the specialized Gröbner basis method presented in Section 4.4 and are generated using our automatic generator of Gröbner basis solvers presented in Chapter 6. Moreover, for the "8-point radial distortion problem", the "6-point problem for one fully calibrated and one up to focal length calibrated camera", and the "plane + parallax + focal length" problem, we also propose the "polynomial eigenvalue solutions" based on the specialized resultant based method presented in Section 5.5.

Beside these new solutions to previously unsolved minimal problems, we provide new solutions to two well known important relative pose problems:

- the 5-point relative pose problem for two calibrated cameras, and
- the 6-point relative pose problem for two cameras with unknown equal focal length.

We show that these two problems lead to polynomial equations that can be solved robustly and efficiently as cubic and quadratic eigenvalue problems. These new solutions are fast, and in the case of the 6-pt solver, also slightly more stable than the existing solution [134]. They are in some sense also more straightforward and easier to implement than the previous solutions [112, 132, 134]. In the case of the "6-point equal focal length problem", we also propose a new Gröbner basis solution generated by our automatic generator and we show that this solution is slightly more stable and also faster then the previous Gröbner basis solutions to this problem [134, 33].

3 State-of-the-Art

"I believe in intuition and inspiration.
Imagination is more important than knowledge.
For knowledge is limited, whereas imagination embraces the entire world,
stimulating progress, giving birth to evolution.
It is, strictly speaking, a real factor in scientific research."

Albert Einstein

The overview of the state-of-the-art will be given in two areas. We begin with a review of general methods for solving systems of polynomial equations. Then, we continue with applications of algebraic methods for solving *minimal problems* appearing in computer vision.

3.1 Solving systems of polynomial equations

Solving systems of polynomial equations is a classical problem with many applications, e.g., in computer vision, robotics, computer graphics and geometric modeling. This problem has its origin in ancient Greece and China. Therefore it is not surprising that there exists a large number of methods for solving systems of polynomial equations.

We can divide them according to several criteria. One possible and very simple division is the division into two classes:

1. numerical methods, and
2. symbolic methods.

In the following we will mostly concentrate on symbolic (algebraic) methods. However, we first briefly review numerical methods.

3.1.1 Numerical methods

The numerical methods for solving systems of polynomial equations can be classified into iterative methods and homotopy methods.

Iterative techniques [139, 4, 64] attempt to solve a system of equations by finding successive approximations to the solutions by starting from an initial guess.

There exist many different types of iterative methods. One of the best known is Newton's method and its variations [62, 63]. Newton's method is known to converge quadratically to local

minima. Different iterative methods with higher orders of convergence [58, 65, 144, 148] appeared recently. Iterative methods are useful for problems involving a large number of variables, but they require a good initial guess to each solutions.

The difficulty in finding a suitable initial guess is avoided by using homotopy continuation methods. The main idea of these methods is to deform a system with known roots into the system that we want to solve. Homotopy continuation methods work in two stages. Firstly, homotopy methods exploit the structure of the system that we want to solve and find the number of roots of this system. Then, using this knowledge, an initial system for which we know solutions and which has exactly as many solutions as our system is constructed. The solutions of this initial system are then gradually transformed to the solutions of the system that we want to solve. In the second stage, continuation methods are applied to follow numerically the paths that originate at the solutions of the initial system toward the solutions of our target system. More about homotopy continuation methods can be found in [6, 99, 143]. In practice, these methods may have problems with robustness and the continuation phase may be also computationally very demanding.

3.1.2 Symbolic methods

The main idea of symbolic methods for solving systems of polynomial equations is to eliminate variables from the system, and in this way, to reduce the problem to finding the roots of univariate polynomials. These methods have their origin in algebraic geometry and are sometimes called algebraic methods or symbolic elimination methods.

In general, algorithms for symbolic methods are efficient for smaller systems. For bigger systems, most of the general algorithms based on algebraic methods suffer from accuracy or efficiency problems. It is because these algorithms often require exact or multiple-precision arithmetic, which slows them down considerably.

The symbolic methods may be divided into:

1. resultant methods,
2. Gröbner bases, and
3. Ritt-Wu's methods.

An overview of classical symbolic methods can be found in [75, 39]. A nice survey of these methods with examples of applications in computer vision is in [116]. In this thesis we propose specialized methods for solving systems of polynomial equations based on the Gröbner bases and the resultant based methods. Therefore, we will review these two methods in more detail.

The Ritt-Wu's methods [120, 145, 146, 147] are based on the reduction of input polynomial set to a family of triangular sets. This method has been used especially to prove geometric theorems for which the method was initially designed.

Resultant methods

Resultant methods are the basis of classical elimination theory. These methods are based on the theory of determinants, and were developed in the late 19^{th} and the early 20^{th} century. They

were created to determine whether a system of polynomial equations has a common solution. However, these methods can be also used for solving these systems.

The main idea of the resultant based methods is to take an initial system of non-linear polynomial equations and generate a possibly larger system of independent polynomial equations such that there are as many equations as monomials in these equations. Then, each monomial can be treated as an unknown and the theory of linear systems can be applied. In other words, these methods linearize a non-linear systems of polynomial equations and the resultant can be therefore expressed in terms of matrices and determinants.

There exist many different types of resultants and methods for obtaining these resultants. One of the best known resultants is the *Sylvester resultant* defined for two univariate polynomials. Let these two polynomials be of the form

$$\begin{aligned} f(x) &= a_r x^r + \dots a_0, \\ g(x) &= b_s x^s + \dots b_0, \end{aligned} \tag{3.1}$$

then the Sylvester resultant of f and g is defined as the determinant of the well known $(r+s) \times (r+s)$ Sylvester matrix

$$Res(f,g) = \det \begin{pmatrix} a_0 & a_1 & \dots & a_r & 0 & 0 & \dots & 0 \\ 0 & a_0 & \dots & a_{r-1} & a_r & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_0 & a_1 & a_2 & \dots & a_r \\ b_0 & b_1 & \dots & b_{s-1} & b_s & 0 & \dots & 0 \\ 0 & b_0 & \dots & b_{s-2} & b_{s-1} & b_s & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & b_0 & b_1 & b_2 & \dots & b_s \end{pmatrix}. \tag{3.2}$$

Then we have that $Res(f,g)$ is zero if and only if f and g have a common root.

Formulas where a resultant is expressed as a determinant of a single matrix are the best that can be obtained for a resultant. Even though in some cases such formulas exist (e.g. for the resultant of three ternary quadrics which can be expressed by a 6×6 determinant [121] or for the resultant of three polynomials of degree r), most of the time we can't express the resultant in the form of a single determinant. Moreover, it is not known if this is possible in general.

Macaulay [100] showed that a multivariate resultant can be written as the ratio of two determinants. Such formula is general, but it has some disadvantages and sometimes is not very practical. For example, it requires dividing two very large polynomials, which can be very time-consuming.

Besides the well known Macaulay's formula there are other ways how to represent multivariate resultants as quotients. The best known resultants are *Bezoutians* or *Bezout's resultants* [44] and *Dixon's resultants* [76].

Other popular multivariate resultants are the *u-resultant* and the *hidden variable resultants* [40], which can be nicely used to solve systems of polynomial equations. Hidden variable resultants were, for example, used by Li to solve two minimal problems in computer vision [98, 96] and

also the problem of estimating radial distortion parameter from nine image point correspondences [97]. The basic idea of this method is to consider one or more of variables as constants or parameters and eliminate other variables from the system.

Many extensions of classical resultants have been considered over the years. Very popular in recent years are sparse resultants, which take into account the sparse structure of the polynomials. Sparse resultants are based on Newton polytopes and mixed subdivisions. A good introduction to sparse resultants and their applications, especially in kinematics, can be found in [45, 137]. A sparse resultant solution to the well known 5-point relative pose problem for calibrated cameras was proposed in [45].

Implementations of different algorithms for computing resultants are contained in many standard computer algebra systems such as Maple [126] or Magma [15, 16].

We will discuss multivariate resultants, Macaulay's method for their computation, and applications of resultants for solving systems of polynomial equations in more detail in Chapter 5.

Gröbner bases

One way how to solve a system of polynomial equations is to construct a new system of polynomial equations with the same solutions as the initial one, but with a simpler structure and then solve this "simpler" system.

This is actually what the second method for symbolic elimination based on Gröbner bases does. This method is based on polynomial ideal theory and multivariate polynomial division and generates special bases of these ideals, called Gröbner bases.

Gröbner bases generate the same ideal [39] as the initial polynomial equations (have the same solutions) but are easier to solve (e.g. the reduced Gröbner basis w.r.t. the lexicographic ordering contains polynomial in one variable only). Computing such basis and "reading off" the solutions from it is one standard method for solving systems of polynomial equations.

Although this sounds trivial the reality is much harder. The problem is that the computation of Gröbner bases is an EXPSPACE-complete problem [82], i.e. large space is necessary for storing intermediate results. Fortunately, in many cases, the behaviour of algorithms for their computation is much better and Gröbner bases can be obtained much faster.

Gröbner bases were developed in 1965 by Bruno Buchberger [18, 22], who named them after his advisor Wolfgang Gröbner. Since then they have been extensively studied and developed.

Bruno Buchberger was also the first who gave an algorithm for computing these bases. This algorithm, known as Buchberger's algorithm, is very simple to describe and implement but in many cases very impractical. The algorithm is based on the construction of S-polynomials and on polynomial division of these S-polynomials [39, 40]. Multivariate polynomial division requires a monomial ordering and different orderings can give rise to radically different Gröbner bases.

For some problems and some orderings, especially lexicographic ordering, the construction of Gröbner bases using this standard Buchberger's algorithm or its variations [18, 19, 21] is very time-consuming and sometimes even does not finish in reasonable time.

Therefore, many improvements of Buchberger's algorithm have been proposed in recent years [20, 36, 53, 55, 57]. They are mostly divided into two groups.

The first group of improvements is dealing with so-called strategies of selection. During the Gröbner basis computations, several choices can be made. We can select an S-pair and also polynomials for reduction. Buchberger [18] proved that the order in which we choose and reduce S-pairs is not important for the correctness of the algorithm, but it is known that both these choices can dramatically affect the overall performance of the algorithm. Several strategies based on different criteria and heuristics have been proposed during the last years. The best known are Buchberger's first and second criterion [20], the Gebauer-Möller criterion [53], the sugar strategy [57], and pair minimization [36]. The well known algorithm, with an improved selection strategy, is the F4 algorithm developed by Faugère [49], though this algorithm is slightly different from standard Buchberger's algorithm. The F4 algorithm not only improves the selection strategy but it also replaces multiple polynomial divisions by row reduction (Gauss-Jordan elimination) of a single sparse matrix. In this way the F4 algorithm transforms computations with polynomials to linear algebra computations. This results in a significant speedup of Gröbner basis computations. Thanks to these improvements, Faugere's F4 algorithm becomes very popular. There exist several implementations of this algorithm [105, 122] including implementations of the F4 algorithm in standard computer algebra systems such as Maple [126] or Magma [15, 16].

The computationally most demanding part of the standard Buchberger's algorithm is the reduction of S-polynomials. Experiments show that for many problems almost 90% of the time is spent by computing the reductions which are zero. This is very inefficient because such reductions have no effect on the computation of the Gröbner basis itself.

Therefore the second group of improvements is trying to remove such useless computations by removing unnecessary S-polynomials. One way how this can be done is to apply a selection strategy [20] which will eliminate S-polynomials that would reduce to zero. The best known algorithm which solves this problem is another algorithm from Faugére called F5 [37]. This algorithm is based on ideas from the paper [108], and in many cases, results in computations without reductions to zero. Using this algorithm Faugére solved some previously untractable problems like cyclic 10. Implementations of this algorithm can be found for example in [127, 122].

When a Gröbner basis is obtained there are several ways it can be used to solve a system of polynomial equations. This mostly depends on the used monomial ordering [39]. Different orderings have different advantages.

For solving systems of polynomial equations, the most suitable ordering is the lexicographic ordering. This ordering results in the system of equations in a "triangular form" from which one equation is a polynomial in one variable only. Solving this equation and back-substituting solutions to the remaining equations, one obtains solutions of the whole system. Unfortunately, computations of Gröbner bases w.r.t. the lexicographic ordering are very time-consuming and for most of the problems can't be used. From a computational complexity point of view, the best ordering is the graded reverse lexicographic ordering.

One way how to use the Gröbner basis computed for another ordering than the lexicographic is to convert this basis to the basis w.r.t. the lexicographic ordering. There are several methods for converting a Gröbner basis from one ordering to another. The best known algorithms are the FGLM algorithm [48] and the Gröbner walk algorithm [38].

Another way how to use the Gröbner basis computed, for example, for the graded reverse

lexicographic ordering, to solve a system of polynomial equations, is to use this basis for constructing a special multiplication matrix also called the "action" matrix [128, 129, 109, 39]. This multiplication matrix is the matrix of the linear operator of the multiplication by a suitably chosen polynomial f in the quotient ring [39]. This matrix can be viewed as a generalization of a companion matrix for solving one polynomial equation in one unknown. It is because solutions of the system of polynomial equations can be recovered from the eigenvalues and eigenvectors of this action matrix [39].

We will describe all these methods for solving systems of polynomial equations based on Gröbner bases in more detail in Chapter 4.

The disadvantage of Gröbner bases is that their construction by a general algorithm may be numerically very unstable when performed in floating point arithmetic and for many larger problems requires computations in exact arithmetic. The reason for this is that in the algorithm for constructing Gröbner bases many polynomial divisions are performed successively. Each division can bring some round-off error. These round-off errors accumulate and at some point it may be completely impossible to tell whether some coefficient is zero or not. Similar problems appear in algorithms where multiple polynomial divisions are replaced for Gauss-Jordan elimination of single matrix like in the algorithm F4 [49]. Here again, after several Gauss-Jordan eliminations, it is impossible to determine whether some coefficients in the matrix are zero or whether two rows are linearly independent or dependent.

These numerical problems have appeared also in some Gröbner based solvers for minimal problems in computer vision [91]. Some techniques for improving the numerical stability of such Gröbner based solvers in computer vision have been proposed in [33, 34, 31].

There are many books available on Gröbner bases, for example [5, 14, 39, 40, 78]. The surveys on the application of the Gröbner bases method can be found in [23].

Implementations of Gröbner bases algorithms and many algorithms based on applications of Gröbner bases are contained in all of the current mathematical software systems like Mathematica, Maple, Magma, Axiom, Derive, Reduce, etc. Also, special software systems exist that are mainly based on the Gröbner bases technique, for example, CoCoA [3], Macaulay [59], Singular [42].

3.2 Minimal problems

Many problems in computer vision, especially problems of computing camera geometry, can be formulated using systems of polynomial equations. Such systems of polynomial equations can have an infinite number of solutions, i.e. are under-determined, no solution, i.e. are over-determined, or these systems can have a finite number of solutions.

In the case of problems of computing camera geometry, the number of equations in the system and the corresponding number of its solutions depend on the number of geometric constraints and the number of input data (usually 2D-2D, 2D-3D or 3D-3D point or line correspondences) used to formulate the problem. The problems solved from a minimal number of input data and using all the possible geometric constrains that lead to a finite number of solutions are often called "minimal problems" or "minimal cases". Most of these minimal problems are named

according to the smallest number of correspondences that are required to solve these problems and to obtain a finite number of possible solutions, e.g., the five point relative pose problem or the six point focal length problem.

Various minimal problems have been recently studied extensively in computer vision [141, 112, 134, 131, 135, 136, 132, 114, 33, 98, 96]. It is because a smaller number of input data is less likely to contain incorrect inputs and therefore considerably reduces the number of samples needed in RANSAC-based algorithms [50], which are widely used in many applications.

In this thesis we add several new minimal problem solutions to the family of recently solved minimal problems, e.g., the perspective three point problem [50, 52], the generalized three point pose problem [114], the five point relative pose problem [112, 132, 98], the six point focal length problem [134, 96], the six point generalized camera problem [135] or the nine-point problem for estimating para-catadioptric fundamental matrices [56].

All these problems lead to systems of polynomial equations. For some problems, like the perspective three point problem, these systems are not so complicated and the solution to them is known for years and can be found using some manipulations and tricks in a closed form [50]. However, more often these problems lead to very complex systems of non-linear polynomial equations and numerical or symbolic methods based on resultants or Gröbner bases need to be used to solve them.

Macaulay's resultant was, for example, used in [141] to solve the problem of estimating absolute pose of a camera with unknown focal length from four 2D-to-3D correspondences and the problem of estimating absolute pose of a camera with unknown focal length and unknown principal point from five 2D-to-3D correspondences. Sparse resultants were used to solve the well know five point relative pose problem [45]. This five point problem [98], as well as the six point relative pose problem for a camera with unknown focal length [96], were also solved using the hidden variable resultant based method [39].

Unfortunately, standard symbolic methods for solving general systems of polynomial equations, like Buchberger's algorithm for computing Gröbner bases [39], or the previously described standard resultant based solutions [141, 45, 98, 96] may be very inefficient when solving minimal problems in computer vision.

It is because minimal problems are usually parts of some large systems which require high or even real-time performance, e.g., structure-from-motion pipelines or recognition systems. Moreover, due to incorrect inputs and noise these minimal problems need to be solved for many different inputs in RANSAC-like algorithms [50] in micro or milliseconds.

Thus, in recent years, various specific algorithms based on algebraic geometry concepts have been proposed. They were focusing on numerical robustness and computational efficiency when solving minimal problems. The main property of these algorithms is that they use specific properties of a system of polynomial equations arising from a particular problem to efficiently solve this problem only.

Stewénius [131] proposed a method for constructing solvers for problems leading to systems of polynomial equations based on Gröbner bases and multiplication matrices. The resulting solvers are based on facts that in a class of problems the "path" to the Gröbner basis is always the same and that algorithms computing Gröbner bases can be realized using Gauss-Jordan elimination [49]. Based on these facts effective and practical algorithms for solving minimal problems

can be designed. Using this method Stewénius et. al. solved problems such as the five point relative pose problem [132], the six point relative pose problem for a camera with unknown constant focal length [134] the six point generalized camera problem [135], the nine point problem for estimating para-catadioptric fundamental matrices [56], the minimal problem for infinitesimal camera motion [133] as well as some other minimal problems [131].

A similar Gröbner basis method was used by other authors to solve the 3-point problem for panorama stitching for a camera with unknown focal length and radial distortion [32], the absolute pose problem for a camera with unknown focal length and radial distortion [70] and the 3-point relative pose problem for a camera with known vertical direction [74].

An efficient solutions to several relative pose problems based on hidden variable resultant method appeared very recently in [60].

All mentioned minimal problems are based on point correspondences; however, there also exist solutions to some minimal problems based on line correspondences [43], combinations of point and line correspondences [115, 119] and combinations of 2D-2D with 2D-3D point correspondences [71].

In this thesis we propose new solutions to several minimal relative pose problems. We describe previous solutions to these problems in more detail in Chapter 7.

4 Gröbner basis methods for solving systems of polynomial equations

"The most practical solution is a good theory..."

Albert Einstein

In the following two chapters we review several classical algebraic methods for solving systems of polynomial equations and propose their specializations suitable for solving problems arising in computer vision.

Throughout these chapters we will consider the following problem

Problem 1. *Given a set* $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}\left[x_1, ..., x_n\right]\}$ *of* m *polynomials in* n *variables over the field* $\mathbb{C}$ *of complex numbers, determine the complete set of solutions of the system*

$$\begin{aligned} f_1\left(x_1, \ldots, x_n\right) &= 0, \\ &\ldots, \\ f_m\left(x_1, \ldots, x_n\right) &= 0. \end{aligned} \tag{4.1}$$

In general such a system may have an infinite number of solutions. However here we are only interested in systems which have a finite number of N solutions and thus $m \geq n$.

Before we start with the description of classical algebraic methods for solving systems of polynomial equations (4.1), we introduce some basic concepts from algebraic geometry.

We use the nomenclature from excellent monographs [39, 40], where all these basic concepts from polynomial algebra, algebraic geometry, and solving systems of polynomial equations are described and most of the theorems we mention also proved.

4.1 Basic Concepts

We start with the definition of the basic element of a polynomial

Definition 1. (Monomial) *A monomial in variables* $x_1, \ldots, x_n$ *is a product*

$$\mathbf{x}^{\alpha} = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}, \tag{4.2}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ *is a vector of exponents in the monomial and* $\alpha_i \in \mathbb{N}_0$*. The total degree of the monomial* $\mathbf{x}^{\alpha}$ *is the sum of the exponents, i.e.* $\alpha_1 + \cdots + \alpha_n$*.*

Definition 2. (Polynomial) *A polynomial f in n variables $(x_1, \ldots, x_n)$ with coefficients from $\mathbb{C}$ is defined as a finite linear combination of monomials (4.2) with coefficients from $\mathbb{C}$*

$$f = \sum_{i=1}^{s} c_i \mathbf{x}^{\alpha(i)}. \tag{4.3}$$

We write $f \in \mathbb{C}[x_1, \ldots, x_n]$, where $\mathbb{C}[x_1, \ldots, x_n]$ is a set of all polynomials in variables $(x_1, \ldots, x_n)$ with coefficients from $\mathbb{C}$.

Monomials $\mathbf{x}^\alpha$ appearing in polynomials (4.3) may be ordered in many different ways. This ordering is important in the polynomial division [40] (a generalization of the division of polynomials in one variable) which gives, in general, different results for different orderings.

Considering monomials $\mathbf{x}^\alpha$ (4.2) in n variables, it is first important to set up an ordering on these variables. Unless stated otherwise, we will use here the following ordering of variables

$$x_1 > x_2 > \cdots > x_n. \tag{4.4}$$

Using this ordering of variables, the i^{th} element of the exponent vector α in the monomial $\mathbf{x}^\alpha$ corresponds to the variable x_i.

With this ordering of variables, there are still many ways to order monomials. However, all useful orderings have to satisfy the following definition of a monomial ordering.

Definition 3. (Monomial Ordering) *A monomial ordering $\succ$ on the set of monomials $\mathbf{x}^\alpha$, or, equivalently, on the exponent vectors $(\alpha_1, \ldots, \alpha_n)$, is a total (linear) ordering satisfying:*

1. *ordering $\succ$ is compatible with multiplication, i.e. if $\mathbf{x}^\alpha \succ \mathbf{x}^\beta$ and $\mathbf{x}^\gamma$ is any monomial, then $\mathbf{x}^\alpha \mathbf{x}^\gamma \succ \mathbf{x}^\beta \mathbf{x}^\gamma$*
2. *ordering $\succ$ is a well-ordering, i.e. every nonempty collection of monomials has the smallest element under $\succ$.*

Here we present two important monomial orderings.

Definition 4. (Lexicographic Ordering) *Let $\mathbf{x}^\alpha$ and $\mathbf{x}^\beta$ be some monomials. We say $\mathbf{x}^\alpha \succ_{lex} \mathbf{x}^\beta$ if, in the difference $\alpha - \beta \in \mathbb{Z}^n$, the leftmost nonzero entry is positive.*

Definition 5. (Graded Reverse Lexicographic Ordering) *Let $\mathbf{x}^\alpha$ and $\mathbf{x}^\beta$ be some monomials. We say $\mathbf{x}^\alpha \succ_{grevlex} \mathbf{x}^\beta$ if $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, or if $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$, and in the difference $\alpha - \beta \in \mathbb{Z}^n$, the rightmost nonzero entry is negative.*

Having a monomial ordering we can define a leading term of a polynomial.

Definition 6. (Leading Term) *The leading term of a polynomial $f = \sum_{i=1}^s c_i \mathbf{x}^{\alpha(i)}$ (4.3) with respect to the monomial ordering $\succ$ is the product $LT_\succ(f) = c_j \mathbf{x}^{\alpha(j)}$, where $\mathbf{x}^{\alpha(j)}$ is the largest monomial appearing in f w.r.t. the ordering $\succ$. Sometimes we just write $LT(f)$ instead of $LT_\succ(f)$.*

Moreover, if $LT(f) = c_j \mathbf{x}^{\alpha(j)}$, then $LC(f) = c_j$ is the leading coefficient and $LM(f) = \mathbf{x}^{\alpha(j)}$ the leading monomial of f.

Example 1. (Leading Term) *Consider a polynomial $f = y^2z + 3xz^2 + x^2$ in $\mathbb{C}[x, y, z]$ (with the usual ordering of variables, $x > y > z$). Then*

$$LT_{\succ_{lex}}(f) = x^2, \tag{4.5}$$

$$LT_{\succ_{grevlex}}(f) = y^2z. \tag{4.6}$$

Definition 7. (Polynomial Division) *Let $\succ$ be some monomial ordering in $\mathbb{C}[x_1, \ldots, x_n]$ and let $F = (f_1, \ldots, f_m)$ be an ordered set of polynomials from $\mathbb{C}[x_1, \ldots, x_n]$. Then every polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$ can be written as*

$$f = q_1 f_1 + q_2 f_2 + \cdots + q_m f_m + r, \tag{4.7}$$

where $q_i, r \in \mathbb{C}[x_1, \ldots, x_n]$, $i = 1, \ldots, m$ are polynomials such that $LT_{\succ}(q_i f_i) \leq LT_{\succ}(f)$ w.r.t. the chosen monomial ordering $\succ$ and the polynomial $r = 0$ or it is a linear combination of monomials that are not divisible by any of $LT_{\succ}(f_i)$, $i = 1, \ldots, m$. We will call r the remainder of f on division by F and write $\overline{f}^F = r$. Note that $\overline{f}^F$ in general depends on the ordering of polynomials in F.

Let's return to our Problem 1 of finding all solutions of the system of polynomial equations (4.1) defined by a set $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ of m polynomials in n variables.

Definition 8. (Affine Variety) *The set of all solutions $(a_1, \ldots, a_n) \in \mathbb{C}^n$ of the system of polynomial equations (4.1) is called the affine variety defined by $f_1, \ldots, f_m$, and is denoted by $V(f_1, \ldots, f_n)$.*

One of the basic concepts in polynomial algebra is the ideal defined by a set of polynomials F.

Definition 9. (Ideal) *The ideal defined by a set of polynomials $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ is the set of all polynomials that can be generated as polynomial combinations of the initial polynomials $f_1, ..., f_m$*

$$I = \left\{ \sum_{i=1}^{m} f_i \, h_i : h_i \in \mathbb{C}[x_1, ..., x_n] \right\}, \tag{4.8}$$

where h_i are arbitrary polynomials from $\mathbb{C}[x_1, ..., x_n]$. We also write $I = \langle f_1, \ldots, f_m \rangle = \langle F \rangle$.

We say that the ideal $I = \langle f_1, \ldots, f_m \rangle$ is *zero-dimensional* if the affine variety $V(I) = V(f_1, \ldots, f_n)$ is finite, i.e. if the system of polynomial equations (4.1) has a finite number of solutions.

Definition 10. (Radical Ideal) *Let I be an ideal. The radical of I is defined as the set*

$$\sqrt{I} = \{g \in \mathbb{C}[x_1, \ldots, x_n] : g^m \in I \text{ for some } m \geq 1\}. \tag{4.9}$$

An ideal I is said to be a radical ideal if $\sqrt{I} = I$.

This means that the radical ideal defined by a set of polynomials $\{f_1, \ldots, f_m\}$ is the largest set of polynomials which vanish on $V(f_1, \ldots, f_m)$.

In general, an ideal I can be generated by many different sets of generators which all share the same solutions. There are special sets of generators called Gröbner bases $G = \{g_1, ..., g_l\}$ w.r.t. some monomial ordering, that have some special properties and are very useful in solving systems of polynomial equations.

Definition 11. (Gröbner basis) *Let I be an ideal and $\succ$ a monomial ordering on $\mathbb{C}[x_1, \ldots, x_n]$. A Gröbner basis for I w.r.t. the monomial ordering $\succ$ is a finite set of polynomials $G = \{g_1, \ldots, g_l\}$, $G \subset I$ with the property that for every nonzero polynomial $f \in I$, $LT(f)$ is divisible by $LT(g_i)$ for some i.*

It can be easily proved that every ideal $I \subset \mathbb{C}[x_1, \ldots, x_n]$ other than $\{0\}$ has a Gröbner basis and that any Gröbner basis satisfying Definition 11 is a basis of I, i.e. polynomials $G = \{g_1, \ldots, g_l\}$ are generators of I. The proof of this can be found for example in [40] (Corollary 6).

Gröbner bases have several interesting properties. The two main properties resulting from the definition are:

- the remainder of some polynomial f on division by a Gröbner basis G is uniquely determined, i.e. it depends only on the used monomial ordering and not on the ordering of the polynomials in G, and
- a polynomial $f \in I$ iff the remainder of f on division by a Gröbner basis G for I is zero, i.e. $\overline{f}^G = 0$.

Gröbner bases also have other interesting properties that we will discuss in Section 4.4 and use to solve Problem 1.

There exist many algorithms for computing Gröbner bases of an ideal. The best known and very simple algorithm is Buchberger's algorithm [39]. Another very popular algorithm for computing Gröbner bases is the F4 [49] algorithm based on linear algebra computations.

To describe Buchberger's algorithm we first define the S-polynomial which was designed to cancel the leading terms of two polynomials.

Definition 12. (S-polynomial) *Let $f, g \in \mathbb{C}[x_1, \ldots, x_n]$ be nonzero polynomials and $\succ$ some fixed monomial ordering on $\mathbb{C}[x_1, \ldots, x_n]$. The S-polynomial of f and g, denoted $S(f, g)$, is the polynomial*

$$S(f,g) = \frac{LCM(LM(f), LM(g))}{LT(f)} f - \frac{LCM(LM(f), LM(g))}{LT(g)} g, \tag{4.10}$$

where $LCM(LM(f), LM(g))$ is the least common multiple of the monomials $LM(f)$ and $LM(g)$.

Example 2. (S-polynomial) *Consider polynomials* $f = x^3y^2 - y^3 + x$ *and* $g = 4x^4y + y^2$ *in* $\mathbb{C}[x, y]$*, and the graded reverse lexicographic ordering* $\succ_{grevlex}$ *with* $x > y$*. Then we have*

$$\begin{aligned} S(f,g) &= \frac{LCM\left(x^3y^2, x^4y\right)}{x^3y^2} f - \frac{LCM\left(x^3y^2, x^4y\right)}{4x^4y} g = xf - \frac{1}{4}yg = \\ &= x^4y^2 - xy^3 + x^2 - x^4y^2 - \frac{1}{4}y^3 = -xy^3 - \frac{1}{4}y^3 + x^2. \end{aligned} \tag{4.11}$$

As it can be seen the leading terms of polynomials f *and* g *are canceled in the S-polynomial.*

An important property of a Gröbner basis results directly from the way of forming S-polynomial remainders. It is known as the Buchberger's Criterion.

Theorem 1. (Buchberger's Criterion) *A finite set of polynomials* $G = \{g_1, \ldots, g_l\}$*,* $G \subset I$ *is a Gröbner basis of* I *if and only if* $\overline{S(g_i, g_j)}^G = 0$ *for all pairs* $i, j \in 1, \ldots, t$*,* $i \neq j$*.*

Proof. The proof of this theorem can be found in [40]. □

The simplest version of the Buchberger's algorithm for computing a Gröbner basis of a given ideal is based on this criterion.

Algorithm 1 Buchberger's algorithm

Input: $F = \{f_1, \ldots, f_m\}$ **Output**: A Gröbner basis $G = \{g_1, \ldots, g_t\}$ for $I = \langle f_1, \ldots, f_m \rangle$, with $F \subset G$.

```
1: G := F
2: repeat
3:    G' := G
4:    for each pair (p, q) such that g_p, g_q ∈ G' and p ≠ q do
5:       S := S(g_p, g_q)^{G'}
6:       if S ≠ 0 then
7:          G := G ∪ {S}
8:       end if
9:    end for
10: until G = G'
```

Finally, we define one more algebraic structure, a quotient ring, which will play an important role in solving systems of polynomial equations.

Definition 13. (Quotient Ring) *Let* I *be an ideal. A quotient ring* $A = \mathbb{C}[x_1, \ldots, x_n]/I$ *is the set of equivalence classes for congruence modulo* I

$$A = \mathbb{C}[x_1, \ldots, x_n]/I = \{[f] : f \in \mathbb{C}[x_1, \ldots, x_n]\}, \tag{4.12}$$

where

$$[f] = [g] \Leftrightarrow f - g \in I. \tag{4.13}$$

The class $[f]$ is sometimes called a coset and is defined as

$$[f] = f + I = \{f + h : h \in I\}. \tag{4.14}$$

Let G be a Gröbner basis of an ideal I. Than we know that the remainder $\overline{f}^G$ is uniquely determined for all $f \in \mathbb{C}[x_1, \ldots, x_n]$. Moreover $f \in I$ iff $\overline{f}^G = 0$. Therefore

$$\overline{f}^G = \overline{g}^G \Leftrightarrow f - g \in I. \tag{4.15}$$

This comes from the fact that we can write $f = h_1 + \overline{f}^G$ and $g = h_2 + \overline{g}^G$ for some $h_1, h_2 \in I$. Therefore if $\overline{f}^G = \overline{g}^G$ we have $f - g = h_1 + \overline{f}^G - h_2 - \overline{g}^G = h_1 - h_2 \in I$. If $f - g \in I$ than $h_1 + \overline{f}^G - h_2 - \overline{g}^G = h_3 + \overline{f}^G - \overline{g}^G \in I$ for some $h_3 = h_1 - h_2 \in I$. Since remainders are uniquely determined and no term of $\overline{f}^G$ and $\overline{g}^G$ is divisible by any $LT(g_i),\ g_i \in G$ and $f - g \in I$ iff $\overline{f - g}^G = 0$, we have $\overline{f}^G - \overline{g}^G = 0$. This means that $\overline{f}^G = \overline{g}^G$.

Moreover, from (4.14) and from the fact that $f = h_1 + \overline{f}^G$ for some $h_1 \in I$ we have

$$\overline{f}^G \in [f]. \tag{4.16}$$

Together with (4.13) and (4.15) this gives us a one-to-one correspondence between the remainders and the cosets

$$\overline{f}^G \leftrightarrow [f]. \tag{4.17}$$

This is a very nice result because we can use the remainder $\overline{f}^G$ as a standard representative of the coset $[f] \in \mathbb{C}[x_1, \ldots, x_n]/I$, i.e. $\left[\overline{f}^G\right] = [f]$.

We can also identify the remainder arithmetic with the arithmetic in $A = \mathbb{C}[x_1, \ldots, x_n]/I$

$$\overline{f}^G + \overline{g}^G = \overline{f + g}^G \leftrightarrow [f] + [g], \tag{4.18}$$

$$\overline{\overline{f}^G . \overline{g}^G}^G = \overline{fg}^G \leftrightarrow [f].[g]. \tag{4.19}$$

This means that the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$ is in fact a vector space with multiplication, i.e. an algebra. It can be proved that this algebra A is finite-dimensional if and only if the ideal I is zero-dimensional [39, 40] (Finiteness Theorem). Since the remainder $\overline{f}^G$ is a linear combination of the monomials $\mathbf{x}^\alpha \notin \langle LT(I)\rangle$, the set

$$B = \{[\mathbf{x}^\alpha] : \mathbf{x}^\alpha \notin \langle LT(I)\rangle\} \tag{4.20}$$

is usually used as a standard basis of this algebra A. The set $\{\mathbf{x}^\alpha : \mathbf{x}^\alpha \notin \langle LT(I)\rangle\}$ is also called the normal set. Since $\mathbf{x}^\alpha \notin \langle LT(I)\rangle$ iff $\overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha$ we can also write

$$B = \left\{[\mathbf{x}^\alpha] : \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}. \tag{4.21}$$

This basis B (4.21) is sometimes called the standard monomial basis and can be easily obtained when we have some Gröbner basis of the ideal I.

In this thesis we will work with different sets of polynomials, e.g. input polynomials F or Gröbner bases G, and various sets of monomials, e.g. monomial basis B (4.21). We will usually denote these sets by curly brackets, e.g. $F = \left\{2x^2 + y^2 + 3y - 12, x^2 - y^2 + x + 3y - 4\right\}$, and not by parenthesis. However, when working with these sets of polynomials or monomials, we need some fixed ordering of polynomials or monomials in these sets, although it may be usually an arbitrary ordering. In such cases we will use the ordering in which elements of such sets are ordered as they are given on the input. For example, for the set $F = \left\{2x^2 + y^2 + 3y - 12, x^2 - y^2 + x + 3y - 4\right\}$ we will denote the first polynomial $f_1 = 2x^2 + y^2 + 3y - 12$ and the second polynomial $f_2 = x^2 - y^2 + x + 3y - 4$. When a result depends on the ordering then we will explicitly write that the set is ordered and we will use parentheses.

4.2 Gröbner bases and linear algebra

Before describing the first class of methods for solving systems of polynomial equations, i.e. methods based on Gröbner bases and on computations in quotient rings $\mathbb{C}[x_1, \ldots, x_n]/I$, we first show how polynomials and Gröbner bases are connected with linear algebra.

Let us consider a set $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ of m polynomials in n variables over the field $\mathbb{C}$ of complex numbers, and let $T_\succ(F)$ denote the set of monomials in F ordered with respect to the ordering $\succ$. Let s be the number of distinct monomials in F, i.e. $|T_\succ(F)| = s$.

Now we define a linear mapping

$$\psi_{T_\succ(F)} : \mathbb{C}[x_1, \ldots, x_n] \to \mathbb{C}^s, \tag{4.22}$$

such that for the polynomial $f = \sum_{i=1}^{s} c_i \mathbf{x}^{\alpha(i)} \in F$ with monomials $\mathbf{x}^{\alpha(i)}$ ordered with respect to the ordering $\succ$

$$\psi_{T_\succ(F)}(f) = (c_1, \ldots, c_s). \tag{4.23}$$

This means that we can interpret vectors of $\mathbb{C}^s$ as polynomials and vice versa. We will call the mapping ψ the *vector representation* of polynomials.

Similarly we can define a *matrix representation* of a set of m polynomials F as

$$\Psi_{T_\succ(F)} : (\mathbb{C}[x_1, \ldots, x_n])^m \to M_{m \times s}(\mathbb{C}), \tag{4.24}$$

such that

$$\Psi_{T_\succ(F)}(f_1, \ldots, f_m) = \begin{pmatrix} \psi_{T_\succ(F)}(f_1) \\ \cdots \\ \psi_{T_\succ(F)}(f_m) \end{pmatrix}. \tag{4.25}$$

Here $M_{m \times s}(\mathbb{C})$ stands for the vector space of all matrices of size $m \times s$ with elements from $\mathbb{C}$. If it is clear with respect to which ordering the representation is created and which set of monomials is used, the subscripts are omitted, i.e. we write only $\Psi_{T(F)}$ or even only Ψ.

These vector and matrix representations of polynomials allow us to use standard concepts of linear algebra and vector spaces when working with polynomials. Since many efficient linear algebra algorithms are available, such representations can significantly simplify and speed up computations with polynomials.

To show where linear algebra may be useful, we describe here how several polynomials can be reduced at the same time using Gaussian elimination, i.e. how to do polynomial division [40] of several polynomials by Gaussian elimination.

Let us first consider polynomial division of a single polynomial, i.e. let us consider a polynomial f which we want to reduce by polynomials $R = (r_1, \ldots, r_k)$. We want to emulate the polynomial division of f by R, i.e. to compute $\overline{f}^R$ using Gaussian elimination of a single matrix $\mathtt{M}$. This means that we need to create this matrix $\mathtt{M}$ as a matrix representation Ψ (4.25) of the polynomial f and suitable monomial multiples of polynomials $r_1, \ldots, r_k$. The main question is which monomial multiples of $r_1, \ldots, r_k$ should we add to this matrix. Before describing how these monomial multiples should look like for general polynomials f and $r_1, \ldots, r_k$, we show how the matrix $\mathtt{M}$ looks for a simple example.

Example 3. (Matrix polynomial division) *Let's consider polynomial $f = x^2 + y + 3$, which we want to reduce by polynomials $R = \left(x + y + 2, y^2 + 3y\right)$ using the graded reverse lexicographic ordering with $x > y$. The polynomial division algorithm performs these steps:*

1. $\left(x^2 + y + 3\right) - x\left(x + y + 2\right) = -xy - 2x + y + 3$,
2. $(-xy - 2x + y + 3) + y\left(x + y + 2\right) = y^2 - 2x + 3y + 3$,
3. $\left(y^2 - 2x + 3y + 3\right) - 1\left(y^2 + 3y\right) = -2x + 3$,
4. $(-2x + 3) + 2\left(x + y + 2\right) = 2y + 7$.

This means that the remainder of f on division by R is $2y + 7$, i.e. $\overline{f}^R = 2y + 7$, and we can write $x^2 + y + 3 = (x - y - 2)\left(x + y + 2\right) + \left(y^2 + 3y\right) + 2y + 7$.

Let's now consider the matrix representation $\Psi_{T_\succ(F)}$ (4.25) of polynomials $F = \{f, xr_1, yr_1, r_2, r_1\}$ w.r.t. the graded reverse lexicographic ordering $x > y$ of monomials, where r_1 and r_2 are polynomials from R. In this case the set of ordered monomials contained in F is $T_\succ(F) = \left(x^2, xy, y^2, x, y, 1\right)$ and the matrix representation $\Psi_{T_\succ(F)}(F)$ has the form

$$\mathtt{M} = \Psi_{T_\succ(F)}(F) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 3 \\ 1 & 1 & 0 & 2 & 0 & 0 \\ 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 \end{pmatrix} \begin{matrix} \leftarrow f \\ \leftarrow xr_1 \\ \leftarrow yr_1 \\ \leftarrow r_2 \\ \leftarrow r_1 \end{matrix} . \tag{4.26}$$

After Gaussian elimination of this matrix $\mathtt{M}$ we obtain the following matrix

$$\widetilde{\mathtt{M}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 3 \\ 0 & 1 & 0 & 2 & -1 & -3 \\ 0 & 0 & 1 & -2 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & -2 & -7 \end{pmatrix} . \tag{4.27}$$

As it can be seen, the last row of the matrix $\widetilde{\mathtt{M}}$ corresponds to the polynomial $-2x-7$ which is, up to some non-zero multiple, in this case -1, equal to the remainder of f on the division by R.

This means that Gaussian elimination on this matrix does polynomial division. The fact that in this case we only obtain the remainder up to some non-zero multiple is not so important. It is because for most of the applications we only need to generate some non-zero multiples of remainder polynomials, or only to know weather the remainder is zero or not.

Let's now show how the matrix $\mathtt{M}$ can be constructed for general polynomials f and $r_1, \ldots, r_k$.

The algorithmic description of the division algorithm can be found in [40]. Here it is sufficient to describe its main idea. In the j^{th} step of the division algorithm, the intermediate result h_j, also called the dividend, is reduced using the divisor r_i with the smallest possible i such that the leading term of r_i divides the leading term of h_j, i.e. $LT(r_i) \,|LT(h_j)$. If such r_i exists, the new dividend has the form $h_{j+1} = h_j - q_{i,j} r_i$, where $q_{i,j} = \frac{LT(h_j)}{LT(r_i)}$.

To emulate this using the matrix $\mathtt{M}$, we need to add to this matrix a row corresponding to the polynomial $q_{i,j} r_i$ for each intermediate dividend h_j and used divisor r_i. To "add a row corresponding to a polynomial p to $\mathtt{M}$" means that we add p to the set of polynomials F that is represented by the matrix $\mathtt{M} = \Psi_{T_\succ(F)}(F)$ (4.24).

Now the question is: which divisor r_i will be used in the j^{th} step of this algorithm? To predict this we do not need to know the exact form of h_j, where $h_j = h_{j-1} - q_{l,j-1} r_l$. It is sufficient to know which monomials appear in the polynomial $h_j = h_{j-1} - q_{l,j-1} r_l$. The polynomial h_j contains maximally all monomials of h_{j-1} and $q_{l,j-1} r_l$, except for the leading monomial $LM(h_{j-1})$. This means that it is sufficient to assume that the polynomial $h_{j-1} - q_{l,j-1} r_l$ contains all these monomials and with such an assumption we will surely not miss any possible divisor. This results in the following algorithm for constructing the matrix $\mathtt{M}$ for emulating polynomial division of polynomial f by polynomials $R = (r_1, \ldots, r_k)$:

Algorithm 2 Matrix polynomial division

Input: f, $R = (r_1, \ldots, r_k)$, monomial ordering $\succ$

Output: Matrix $\mathtt{M}$ in the matrix representation of polynomials F, i.e. $\mathtt{M} = \Psi_{T_\succ(F)}(F)$, such that the remainder $\overline{f}^R \in F$

1: Set $F := \{f\}$ and let $D = T_\succ(F)$, where $T_\succ(F)$ is the ordered set of monomials of F
2: **while** D is non-empty **do**
3: Let $\mathbf{x}^\alpha$ be the maximal monomial in D w.r.t. $\succ$
4: $D := D \setminus \{\mathbf{x}^\alpha\}$
5: **if** $\mathbf{x}^\alpha$ can be reduced by some $r_i \in R$ **then**
6: Let $r_i \in R$ be the divisor with the smallest possible i such that $LT(r_i)$ divides $\mathbf{x}^\alpha$ and let $q := \frac{\mathbf{x}^\alpha}{LT(r_i)}$
7: Set $F := F \cup \{q r_i\}$ and $\mathtt{M} = \Psi_{T_\succ(F)}(F)$
8: Set $D := D \cup \{$ all monomials of $q r_i$ except of $\mathbf{x}^\alpha$ $\}$ ordered w.r.t. $\succ$
9: **end if**
10: **end while**

It can be proved that this algorithm terminates after a final number of steps and that the re-

sulting matrix $\mathtt{M} = \Psi_{T_\succ(F)}(F)$ contains all necessary polynomials for emulating polynomial division by its Gaussian elimination. After Gaussian elimination of M, the final reduced polynomial $\overline{f}^R$ will either be zero, i.e. will correspond to a zero row of reduced M, or will have a leading term that cannot be reduced by any leading term of $r_1, \ldots, r_k$. This means that, in this case, this reduced polynomial will correspond to a row of reduced M with a pivot that was not a pivot in M. Recall that the result of polynomial division, i.e. the polynomial $\overline{f}^R$, will for general polynomials $R = (r_1, \ldots, r_k)$ depend on the ordering of polynomials $r_1, \ldots, r_k$ in R.

Of course the matrix M constructed using the proposed algorithm might also contain some unnecessary rows. It is because we have assumed that no terms cancel in $h_j - qr_i$, which might not always be true. However, this is not a problem since for polynomials with general coefficients only a small number of unnecessary rows is usually generated.

For the polynomials from Example 3, the presented algorithm generates directly the matrix (4.27) without any unnecessary rows.

This algorithm can be easily extended to emulate the polynomial division of several polynomials by Gaussian elimination of a single matrix. Let us consider that we want to reduce polynomials $f_1, \ldots, f_m$ by $R = (r_1, \ldots, r_k)$. The algorithm for constructing a matrix emulating polynomial division of these polynomials first sets $F = \{f_1, \ldots, f_m\}$ and then continues as Algorithm 2 for a single polynomial.

A similar algorithm is used in the F4 algorithm [49] to perform division of S-polynomials when constructing a Gröbner basis.

Note that using Gauss-Jordan elimination instead of Gaussian elimination, i.e. computing reduced row-echelon form of M, will correspond to fully reducing $f_1, \ldots, f_m$ using polynomial division. This means that not only the leading terms of $f_1, \ldots, f_m$ cannot be further reduced by R, but no terms of $f_1, \ldots, f_m$ can be further reduced.

Here we have shown how several polynomials can be reduced at the same time by using Gaussian elimination of a single matrix. Polynomial division is the main part of all algorithms for finding Gröbner bases, and therefore, its good implementation may speed them up. This is also the main idea of the F4 algorithm for computing Gröbner bases [49]. F4 speeds up the reduction step by exchanging multiple polynomial divisions of S-polynomials for Gaussian elimination of a single matrix created similarly as described above.

In Section 4.4 we will show that the matrix representation of polynomials and Gauss-Jordan elimination can be also used for efficient generation of polynomials from an ideal. Moreover in [33, 31] it was shown that the matrix representation and efficient matrix algorithms like QR or LU decomposition can be used for improving numerical stability of Gröbner bases computations.

4.3 Standard Gröbner basis methods for solving systems of polynomial equations

In this section we will describe several standard approaches to solving systems of polynomial equations based on Gröbner bases. We will start with very simple and straightforward approach which can be viewed as a generalization of Gaussian elimination for solving systems of linear

equations. This approach is based on the elimination properties of the Gröbner bases w.r.t. the lexicographic ordering. Then we will describe more complicated, but also more efficient methods, which are based on Gröbner bases w.r.t. other monomial orderings and on computations in the quotient rings $A = \mathbb{C}[x_1, \ldots, x_n]/I$.

Note that all these algebraic methods finally lead to solving one equation in one variable or to finding eigenvalues and eigenvectors, which are usually solved using some numerical method.

4.3.1 Solving systems of equations by elimination of variables

The first method for solving systems of polynomial equations is based on a very nice property of the Gröbner basis w.r.t. the lexicographic ordering, i.e. the property that for a zero-dimensional ideal I the lexicographic Gröbner basis contains a polynomial in one variable. Such a polynomial can be easily solved using some numerical methods, like Newton's or Newton-Raphson method [118, 62, 63], bracketing using Sturm sequences [66] or by finding the eigenvalues of the companion matrix [9]. Then the backsubstitution can be used to solve for other variables.

To describe this method in more detail, let us first define an elimination ideal.

Definition 14. (Elimination Ideal) *Let I be an ideal in $\mathbb{C}[x_1, \ldots, x_n]$, then the k^{th} elimination ideal is*

$$I_k = I \cap \mathbb{C}[x_{k+1}, \ldots, x_n]. \tag{4.28}$$

This means that the elimination ideal I_k consists of all polynomials from the ideal $I = \langle f_1, \ldots, f_m \rangle$ which contain only variables $x_{k+1}, \ldots, x_n$, i.e. these polynomials can be obtained as such polynomial combinations of the initial polynomials $f_1, \ldots, f_m$ that eliminate variables $x_1, \ldots, x_k$ from these initial polynomials.

Now we can state the following important theorem

Theorem 2. (Elimination Theorem) *Let G be the Gröbner basis of I w.r.t. the lexicographic ordering with $x_1 > x_2 > \cdots > x_n$. Then*

$$G_k = G \cap \mathbb{C}[x_{k+1}, \ldots, x_n] \tag{4.29}$$

is a Gröbner basis of the k^{th} elimination ideal I_k.

Proof. The proof of this theorem can be found in [40]. □

This theorem shows that the lexicographic Gröbner basis of a zero-dimensional ideal contains non-zero polynomials from all elimination ideals I_k, i.e. it successively eliminates more and more variables and ends with a polynomial in one variable.

Therefore, this Gröbner basis can be used to solve the system of polynomial equations (4.1) in the following way:

Algorithm 3 Elimination Gröbner Basis Method

1. Compute a Gröbner basis G of the ideal $I = \langle f_1, \ldots, f_m \rangle$ w.r.t. the lexicographic ordering.
2. Take a polynomial/polynomials in one variable from G and solve it/them algebraically or for higher order polynomials using some numerical method. Note that the existence of such polynomial in G is ensured only if I is zero-dimensional, i.e. the system (4.1) has a finite number of solutions.
3. Extend obtained partial solutions to solutions of the whole system (4.1), i.e. backsubstitute obtained solutions into the polynomials from G and repeat step 2.

Note that not all partial solutions can be always extended to solutions of the whole system. The condition when this extension is possible is described in the Extension Theorem, see [39, 40]. However, what is more important is that all solutions of the system (4.1) truncate to some partial solution. Therefore, using the presented method we obtain all solutions of our system of polynomial equations.

In the following by "solutions for x_i" we will mean x_i-coordinates of the points of $V(I)$, i.e. the roots of the generator of the elimination ideal $I \cap \mathbb{C}[x_i]$.

Example 4. (Elimination Gröbner basis method) *To show how this method works, let us consider a simple system of two polynomial equations in two unknowns*

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0. \end{aligned} \tag{4.30}$$

This system represents the intersection of an ellipse and a hyperbola in $\mathbb{C}^2$ and has four solutions, see Figure 4.1 (left).

These solutions can be found using different methods. Here we show how they can be found using the presented Elimination Gröbner Basis Method.

The first step of this method calls for computing the Gröbner basis G of the ideal

$$I = \left\langle 2x^2 + y^2 + 3y - 12, x^2 + x - y^2 + 3y - 4 \right\rangle \tag{4.31}$$

w.r.t. the lexicographic ordering. Such basis can be computed using some standard algorithm, e.g. Buchberger's algorithm [39]. Figure 4.1 (right) shows examples of polynomials from the ideal I (4.31).

In this case the reduced Gröbner basis of the ideal I w.r.t. the lexicographic ordering $x > y$ is

$$G = \left\{ 9y^4 - 18y^3 - 13y^2 + 30y - 8, 2x - 3y^2 + 3y + 4 \right\}. \tag{4.32}$$

As it can be seen this basis contains one polynomial in one variable, i.e. the polynomial $9y^4 - 18y^3 - 13y^2 + 30y - 8$ in y. This polynomial is a generator of the elimination ideal $I_1 = I \cap \mathbb{C}[y]$.

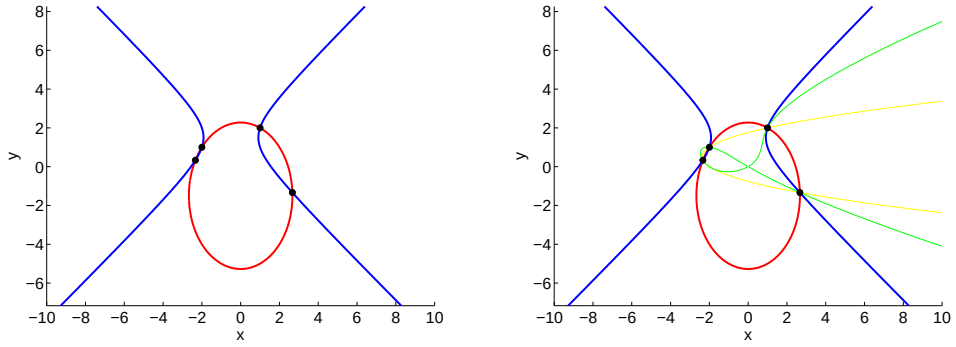

Figure 4.1: (Left) The intersection of the ellipse $2x^2 + y^2 + 3y - 12 = 0$ and the hyperbola $x^2 - y^2 + x + 3y - 4 = 0$. (Right) Polynomials from the ideal $I = \langle 2x^2 + y^2 + 3y - 12, x^2 - y^2 + x + 3y - 4 \rangle$.

Since this polynomial has degree four we can easily find all its zeros. Moreover, in this case all zeros are rational numbers and this polynomial can be nicely factored in $\mathbb{Q}[z]$ *as*

$$9\,(y-1)\,(y-2)\left(y-\frac{1}{3}\right)\left(y+\frac{4}{3}\right). \tag{4.33}$$

Thus we obtain four solutions for y*, i.e.* $y = 1,\ 2,\ \frac{1}{3},\ -\frac{4}{3}$.

In the final step we extend the obtained partial solutions for y *to solutions of the whole system (4.30). We do this by backsubstituting all obtained solutions for* y *to the second polynomial from the Gröbner basis* G *(4.32), i.e. to the polynomial* $2x - 3y^2 + 3y + 4$*. In this way we obtain four solutions of the system (4.30)*

$$(-2,1)\,,(1,2)\,,\left(-\frac{7}{3},\frac{1}{3}\right),\left(\frac{8}{3},-\frac{4}{3}\right). \tag{4.34}$$

This Elimination Gröbner basis method for solving systems of polynomial equations is straightforward and is in fact a generalization of the well known method for solving systems of linear polynomial equations based on Gaussian elimination [106]. However, it has two main limitations:

1. Gröbner bases w.r.t. the lexicographic ordering are often very large and very expensive to compute, and
2. most of the time polynomial equations in one variable can be solved only numerically and therefore the obtained solutions of these one-variable equations are only approximate. After the backsubstitution of such approximate solutions to the remaining polynomial equations we are solving only an approximate system. In this way we accumulate errors, which may be after several steps quite large.

Therefore this lexicographic Gröbner basis method is not feasible for many problems [39].

4.3.2 Solving systems of equations by Gröbner basis conversion

The second standard method for solving systems of polynomial equations tries to avoid the problem of large computational complexity of the Gröbner basis w.r.t. the lexicographic ordering by computing a Gröbner basis w.r.t. some other monomial ordering. Then this basis is converted to the lexicographic Gröbner basis and the method continues as the Elimination Gröbner Basis Method presented in the previous section.

Usually, the graded reverse lexicographic (grevlex) ordering from Definition 5, which often requires far less computational effort than a lexicographic Gröbner basis computation, is used in this method [40, 13]. Then some algorithm for converting a Gröbner basis with respect to one monomial ordering to a Gröbner basis with respect to a different monomial ordering is used. The FGLM algorithm [48] can be used for zero-dimensional ideals and the Gröbner walk algorithm [38] for general ideals.

Since we are considering zero-dimensional ideals, we briefly describe here the main ideas of the FGLM algorithm.

The algorithm starts with some Gröbner basis G of the ideal I and converts it to a lexicographic Gröbner basis G_{lex}, or a Gröbner basis w.r.t. some other monomial ordering, of the same ideal. The algorithm uses only two structures, a list $G_{lex} = \{g_1, \ldots, g_k\}$, which at each stage contains a subset of the lexicographic Gröbner basis, and a list B_{lex}, which contains a subset of the monomial basis of the quotient ring $\mathbb{C}[x_1, \ldots, x_n]/I$. Both these lists are initially empty.

For each monomial $\mathbf{x}^\alpha \in \mathbb{C}[x_1, \ldots, x_n]$ in increasing lexicographic ordering, starting with $\mathbf{x}^\alpha = 1$, the algorithm performs these three steps:

Algorithm 4 FGLM

1. For the input $\mathbf{x}^\alpha$, compute $\overline{\mathbf{x}^\alpha}^G$
   - If $\overline{\mathbf{x}^\alpha}^G = \sum_j c_j \overline{\mathbf{x}^{\alpha(j)}}^G$, where $\left[\mathbf{x}^{\alpha(j)}\right] \in B_{lex}$ and $c_j \in \mathbb{C}$, i.e. if $\overline{\mathbf{x}^\alpha}^G$ is linearly dependent on the remainders on division by G of the monomials in B_{lex}, then add polynomial $g = \mathbf{x}^\alpha - \sum_j c_j \mathbf{x}^{\alpha(j)} \in I$ to G_{lex} as the last element.
   - Otherwise add $[\mathbf{x}^\alpha]$ to B_{lex} as the last element.
2. Termination Test

   If a polynomial g having $LT(g)$ equal to a power of the greatest variable in the lexicographic ordering was added to G_{lex}, then STOP.
3. Next Monomial

   Replace $\mathbf{x}^\alpha$ with the next monomial, w.r.t. the lexicographic ordering, which is not divisible by any of the monomials $LT(g_i)$ for $g_i \in G_{lex}$. GOTO 1.

Note that the original ordering (not the lexicographic ordering) is used for division by G in this algorithm.

Theorem 3. *The FGLM algorithm terminates for every input Gröbner basis G of a zero-dimensional ideal I and correctly computes a lexicographic Gröbner basis G_{lex} for I and the lexicographic monomial basis B_{lex} for the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$.*

Proof. The proof of this theorem can be found in [39].

Here we only show that the first polynomial added to G_{lex} using this algorithm is a polynomial in one variable (the smallest in the used lexicographic ordering).

Let the ordering of the variables be $x_1 > x_2 > \cdots > x_n$. Using this ordering, the FGLM algorithm starts with the monomial $\mathbf{x}^\alpha = 1$ and continues with monomials x_n, x_n^2, x_n^3 and so on in increasing lexicographic order. Let us consider the set $S_1 = \left\{[1], [x_n], [x_n]^2, \ldots\right\}$ with $[x_n]^i \in A = \mathbb{C}[x_1, \ldots, x_n]/I,\ i \geq 0$. Since our ideal is zero-dimensional, the algebra A is finite dimensional. This means that the set S_1 is linearly dependent in A. Moreover, also the set $S_2 = \left\{1, \overline{x_n}^G, \overline{x_n^2}^G, \overline{x_n^3}^G, \ldots\right\}$ is linearly dependent.

Let i be the smallest positive integer for which $\left\{1, \overline{x_n}^G, \overline{x_n^2}^G, \ldots, \overline{x_n^i}^G\right\}$ is linearly dependent. This means that in the FGLM algorithm monomial cosets $[1], [x_n], \left[x_n^2\right], \ldots \left[x_n^{i-1}\right]$ will be added to B_{lex} and we can write

$$\overline{x_n^i}^G = \sum_{j=1}^{i-1} c_j \overline{x_n^j}^G, \tag{4.35}$$

with $c_j \in \mathbb{C}$. Therefore the polynomial

$$x_n^i - \sum_{j=1}^{i-1} c_j x_n^j \in I \tag{4.36}$$

is a polynomial in one variable (x_n), which is added to G_{lex} as the first polynomial. □

For some applications, e.g. where it is sufficient to find solutions only for one variable or where solutions for other variables can be found in a different "easier" way, the algorithm can be stopped after finding this one-variable polynomial.

Example 5. (FGLM method) *Let us consider the system of polynomial equations (4.30) from Example 4, i.e. the system*

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0. \end{aligned} \tag{4.37}$$

Here we show how this system can be solved using the FGLM conversion algorithm. This algorithm starts with a Gröbner basis G w.r.t. some "feasible" monomial ordering. Usually the graded reverse lexicographic ordering (grevlex) is used here. In this case the Gröbner basis G w.r.t. this grevlex ordering consists of the following two polynomials

$$G = \left\{3y^2 - 2x - 3y - 4, 3x^2 + x + 6y - 16\right\}. \tag{4.38}$$

The computation of this basis is simpler than the computation of the lexicographic Gröbner basis (4.32) for the same ideal. In this case both polynomials from the grevlex Gröbner basis G (4.38) can be easily obtained from the initial polynomials (4.37) as their simple linear combinations. Denoting the input polynomials as f_1 and f_2 and polynomials from the grevlex Gröbner basis G as g_1 and g_2, we have $g_1 = f_1 - 2f_2$ and $g_2 = f_1 + f_2$.

Now we transform this grevlex Gröbner basis G to the lexicographic Gröbner basis using the presented FGLM algorithm.

First we set $G_{lex} = \{\}$ and $B_{lex} = \{\}$. Then we go through monomials $\mathbf{x}^\alpha \in \mathbb{C}[x, y]$ in increasing lexicographic order for $x > y$, i.e. we start with 1 and continue with y, y^2 and so on.

For each of these monomials, we first perform the main loop of the FGLM algorithm. This main loop requires to compute $\overline{\mathbf{x}^\alpha}^G$ and to check if this remainder is linearly dependent on the remainders of the monomials in B_{lex} on the division by G. In this case we have

1. $\overline{1}^G = 1$, *and since B_{lex} is empty we set $B_{lex} = \{[1]\}$,*
2. $\overline{y}^G = y$, *and since the set $\{y, 1\}$ is linearly independent we set $B_{lex} = \{[1], [y]\}$,*
3. $\overline{y^2}^G = y^2 - \frac{1}{3}\left(3y^2 - 2x - 3y - 4\right) = \frac{2}{3}x + y + \frac{4}{3}$, *and since the set $\left\{\frac{2}{3}x + y + \frac{4}{3}, y, 1\right\}$ is linearly independent we set $B_{lex} = \left\{[1], [y], \left[y^2\right]\right\}$*
4. $\overline{y^3}^G = y^3 - \left(\frac{1}{3}y + \frac{1}{3}\right)\left(3y^2 - 2x - 3y - 4\right) = \frac{2}{3}xy + \frac{2}{3}x + \frac{7}{3}y + \frac{4}{3}$, *and since the set $\left\{\frac{2}{3}xy + \frac{2}{3}x + \frac{7}{3}y + \frac{4}{3}, \frac{2}{3}x + y + \frac{4}{3}, y, 1\right\}$ is linearly independent we set $B_{lex} = \{[1], [y], \left[y^2\right], \left[y^3\right]\}$*
5. $\overline{y^4}^G = y^4 - \left(\frac{1}{3}y^2 + \frac{2}{9}x + \frac{1}{3}y + \frac{7}{9}\right)\left(3y^2 - 2x - 3y - 4\right) - \frac{4}{27}\left(3x^2 + x + 6y - 16\right) = \frac{4}{3}xy + \frac{62}{27}x + \frac{25}{9}y + \frac{148}{27}$, *and since the set $\left\{\frac{4}{3}xy + \frac{62}{27}x + \frac{25}{9}y + \frac{148}{27}, \frac{2}{3}xy + \frac{2}{3}x + \frac{7}{3}y + \frac{4}{3}, \frac{2}{3}x + y + \frac{4}{3}, y, 1\right\}$ is linearly dependent we let $B_{lex} = \left\{[1], [y], \left[y^2\right], \left[y^3\right]\right\}$ and set $G_{lex} = \left\{y^4 - c_3y^3 - c_2y^2 - c_1y - c_0\right\}$, where c_j are coefficients from the linear combination*

$$\overline{y^4}^G = \sum_{j=0}^{3} c_j \overline{y^j}^G. \tag{4.39}$$

In this case

$$\overline{y^4}^G = 2\overline{y^3}^G + \frac{13}{9}\overline{y^2}^G - \frac{10}{3}\overline{y}^G - \frac{8}{9}\overline{1}^G \tag{4.40}$$

and therefore, the first element of the G_{lex}, i.e. the lexicographic Gröbner basis, will be polynomial $y^4 - 2y^3 - \frac{13}{9}y^2 + \frac{10}{3}y - \frac{8}{9}$.

Next we can continue with the next two steps of the FGLM algorithm, i.e. the termination test and the next monomial step. This means that we can continue with monomials containing x in the lexicographic ordering and for these monomials again perform the main loop. However, in this case, it is sufficient to stop here and find solution for y from the polynomial equation

$$y^4 - 2y^3 - \frac{13}{9}y^2 + \frac{10}{3}y - \frac{8}{9} = 0. \tag{4.41}$$

This equation gives us solutions $y = \left\{1,\ 2,\ \frac{1}{3},\ -\frac{4}{3}\right\}$. *Solutions for* x *can be obtained by substituting the solutions for* y *to one from the initial polynomial equations (4.37). In this way we obtain four solutions of the system (4.37)*

$$(-2,1)\,,(1,2)\,,\left(-\frac{7}{3},\frac{1}{3}\right),\left(\frac{8}{3},-\frac{4}{3}\right). \tag{4.42}$$

Note that the polynomial equation (4.41) is equivalent to the polynomial equation $9y^4 - 18y^3 - 13y^2 + 30y - 8 = 0$ *obtained from the lexicographic Gröbner basis (4.32) computed in Example 4.*

4.3.3 Solving systems of equations via eigenvalues end eigenvectors

The third method for solving systems of polynomial equations based on Gröbner bases and on computations in the quotient ring $A = \mathbb{C}\left[x_1,\ldots,x_n\right]/I$ tries to solve both problems of the lexicographic Elimination Gröbner Basis Method presented in Section 4.3.1, i.e. the problem with large computational complexity of lexicographic Gröbner bases, and the problem with accumulating errors by backsubstitution.

The first problem is solved as in the previous Gröbner basis conversion method, i.e. by computing a Gröbner basis w.r.t. some other, more favorable, monomial ordering, e.g. the grevlex ordering. The second problem with accumulating errors by backsubstituting approximate solutions of one-variable polynomials to the remaining polynomials and solving these approximate equations, tries to solve the presented method by finding all solutions independently as eigenvalues of several matrices or at once as eigenvalues and eigenvectors of a single matrix.

While the lexicographic Elimination Gröbner basis Method is related to Gaussian elimination used for solving systems of linear equations, this method will be related to the companion matrix method for solving a polynomial in one variable [9].

Let us consider the mapping $T_f\colon A \to A$ of the multiplication by a polynomial $f \in \mathbb{C}\left[x_1,\ldots,x_n\right]$ defined as

$$T_f\left([g]\right) = [f]\,.\,[g] = [fg] \in A. \tag{4.43}$$

It can be shown that T_f is a linear mapping for which $T_f = T_g$ iff $f - g \in I$.

In our case we are considering only systems with a finite number of solutions and therefore A is a finite-dimensional vector space over $\mathbb{C}$. For such a finite-dimensional vector space we can represent $T_f : A \to A$ by its matrix with respect to some basis B of A.

To be precise, let $B = \left([b_1],\ldots,[b_s]\right)$ be a basis of $A = \mathbb{C}\left[x_1,\ldots,x_n\right]/I$, where $b_i \in \mathbb{C}\left[x_1,\ldots,x_n\right],\ i = 1,\ldots,s$. Usually a standard monomial basis (4.20) $B = \left(\left[\mathbf{x}^{\alpha(1)}\right],\ldots,\left[\mathbf{x}^{\alpha(s)}\right]\right)$ or other basis consisting only of monomials is the most useful here. Then we can represent the multiplication mapping T_f by representing the image $T_f\left([b_i]\right)$ of every basis element $[b_i]\,,\ i = 1\ldots,s$, in terms of $B = \left([b_1],\ldots,[b_s]\right)$.

Definition 15. (Multiplication Matrix) *We say that the complex $s \times s$ matrix $\mathtt{M}_f := (m_{ij})$ such that*

$$T_f\left([b_j]\right) = [fb_j] = \sum_{i=1}^{s} m_{ij}\,[b_i] \tag{4.44}$$

is the multiplication (or the action) of T_f corresponding to $f \in \mathbb{C}[x_1, \ldots, x_n]$.

In this definition we follow the notation used in [39], where the image $T_f\left([b_j]\right)$ corresponds to the j^{th} column of the multiplication matrix. In many papers, however, this image corresponds to the j^{th} row and therefore the multiplication matrix is the transposed matrix $\mathtt{M}_f^{\top}$ from our definition.

Having a Gröbner basis G, the multiplication matrix $\mathtt{M}_f$ w.r.t. the standard monomial basis $B = \left(\left[\mathbf{x}^{\alpha(1)}\right], \ldots, \left[\mathbf{x}^{\alpha(s)}\right]\right)$ is constructed by computing the remainders $\overline{\mathbf{x}^{\alpha(i)} f}^{G}$ for all basis monomials $\left[\mathbf{x}^{\alpha(i)}\right] \in B,\ i = 1, \ldots, s$, as the following example shows.

Example 6. (Multiplication matrix) *Let us consider our well known system of polynomials equations form Example 4 and 5, i.e. the system*

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0. \end{aligned} \tag{4.45}$$

The grevlex Gröbner basis G of the ideal I defined by these two polynomials consists of the following two polynomials

$$G = \left\{3y^2 - 2x - 3y - 4, 3x^2 + x + 6y - 16\right\}. \tag{4.46}$$

Therefore, the corresponding standard monomial basis $B = \left\{[\mathbf{x}^{\alpha}] : \mathbf{x}^{\alpha} \notin \langle LT(I)\rangle\right\}$ of the quotient ring $A = \mathbb{C}[x, y]/I$ is as follows

$$B = \left([xy], [x], [y], [1]\right). \tag{4.47}$$

To compute the multiplication matrix $\mathtt{M}_x$ for multiplication by $[x]$ in A, it is sufficient to compute

1. $T_x\left([xy]\right) = \left[x^2 y\right] = \left[\overline{x^2 y}^{G}\right] = \left[-\frac{1}{3}xy - \frac{4}{3}x + \frac{10}{3}y - \frac{8}{3}\right] = -\frac{1}{3}[xy] - \frac{4}{3}[x] + \frac{10}{3}[y] - \frac{8}{3}[1]$,
2. $T_x\left([x]\right) = \left[x^2\right] = \left[\overline{x^2}^{G}\right] = \left[-\frac{1}{3}x - 2y + \frac{16}{3}\right] = -\frac{1}{3}[x] - 2[y] + \frac{16}{3}[1]$,
3. $T_x\left([y]\right) = [xy]$,
4. $T_x\left([1]\right) = [x]$.

Then the multiplication matrix $\mathtt{M}_x$ *has the form*

$$\mathtt{M}_x = \begin{pmatrix} -\frac{1}{3} & 0 & 1 & 0 \\ -\frac{4}{3} & -\frac{1}{3} & 0 & 1 \\ \frac{10}{3} & -2 & 0 & 0 \\ -\frac{8}{3} & \frac{16}{3} & 0 & 0 \end{pmatrix}. \tag{4.48}$$

Similarly, the multiplication matrix $\mathtt{M}_y$ *for multiplication by* y *in* A *can be constructed by computing*

1. $T_y([xy]) = \left[xy^2\right] = \left[\overline{xy^2}^G\right] = \left[xy + \frac{10}{9}x - \frac{4}{3}y + \frac{32}{9}\right] = [xy] + \frac{10}{9}[x] - \frac{4}{3}[y] + \frac{32}{9}[1]$,
2. $T_y([x]) = [xy]$,
3. $T_y([y]) = \left[y^2\right] = \left[\overline{y^2}^G\right] = \left[\frac{2}{3}x + y + \frac{4}{3}\right] = \frac{2}{3}[x] + [y] + \frac{4}{3}[1]$,
4. $T_y([1]) = [y]$.

Therefore the multiplication matrix $\mathtt{M}_y$ *has the form*

$$\mathtt{M}_y = \begin{pmatrix} 1 & 1 & 0 & 0 \\ \frac{10}{9} & 0 & \frac{2}{3} & 0 \\ -\frac{4}{3} & 0 & 1 & 1 \\ \frac{32}{9} & 0 & \frac{4}{3} & 0 \end{pmatrix}. \tag{4.49}$$

The multiplication matrix (4.44) has several nice properties which can be used to solve systems of polynomial equations (4.1). The first one is described by the following theorem.

Theorem 4. *Let* I *be a zero-dimensional ideal and let* $\mathtt{M}_f$ *be a matrix of the linear map* $T_f : A \to A$ *of multiplication by a polynomial* $f \in \mathbb{C}[x_1, \ldots, x_n]$ *in* $A = \mathbb{C}[x_1, \ldots, x_n]/I$. *Then* $\lambda \in \mathbb{C}$ *is an eigenvalue of the matrix* $\mathtt{M}_f$ *iff* λ *is a value of the function* f *on* $V(I)$.

Proof. The proof of this theorem can be found in [39]. □

This theorem says that the eigenvalues of the multiplication matrices $\mathtt{M}_{x_i}$, where x_i are our variables, correspond to the x_i-coordinates of the points of $V(I)$, i.e. to the roots of the generator of the elimination ideal $I \cap \mathbb{C}[x_i]$. This means that the solutions of the whole system of polynomial equations (4.1) can be found as eigenvalues of n matrices $\mathtt{M}_{x_i},\ i = 1, \ldots, n$.

Example 7. (Companion matrix) *Let's first consider an example of one polynomial in one variable* x. *Let this polynomial have the form*

$$f = \sum_{i=1}^{s} c_i x^i, \tag{4.50}$$

where c_i are some coefficients from $\mathbb{C}$ and let $c_s = 1$, i.e. the polynomial f is a monic polynomial. In this case the Gröbner basis consists only of this polynomial f, i.e. $G = \{f\}$ and the monomial basis B of the quotient ring $A = \mathbb{C}[x]/I$ has the form

$$B = \left([1], [x], \left[x^2\right], \ldots, \left[x^{s-1}\right]\right). \tag{4.51}$$

We know that $\overline{x^s}^G = -c_0 - c_1 x - \cdots - c_{s-1}x^{s-1}$. Therefore the multiplication matrix $\mathtt{M}_x$ will be $s \times s$ matrix of the form

$$\mathtt{M}_x = \begin{pmatrix} 0 & 0 & \ldots & 0 & -c_0 \\ 1 & 0 & \ldots & 0 & -c_1 \\ 0 & 1 & \ldots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -c_{s-1} \end{pmatrix}. \tag{4.52}$$

This is the well known Frobenius companion matrix [9] whose eigenvalues give solutions of the polynomial equation $f(x) = 0$.

Example 8. (Gröbner basis eigenvalue method) *In the case of the system of polynomial equations (4.45) from Example 6, we have*

$$\mathtt{M}_x = \begin{pmatrix} -\frac{1}{3} & 0 & 1 & 0 \\ -\frac{4}{3} & -\frac{1}{3} & 0 & 1 \\ \frac{10}{3} & -2 & 0 & 0 \\ -\frac{8}{3} & \frac{16}{3} & 0 & 0 \end{pmatrix}, \tag{4.53}$$

which gives us four eigenvalues $1, -\frac{7}{3}, -2, \frac{8}{3}$, solutions for x.

Similarly the eigenvalues of $\mathtt{M}_y$

$$\mathtt{M}_y = \begin{pmatrix} 1 & 1 & 0 & 0 \\ \frac{10}{9} & 0 & \frac{2}{3} & 0 \\ -\frac{4}{3} & 0 & 1 & 1 \\ \frac{32}{9} & 0 & \frac{4}{3} & 0 \end{pmatrix} \tag{4.54}$$

give us solutions for y, i.e. $y = 1, 2, \frac{1}{3}, -\frac{4}{3}$.

One has to try all $4 \times 4 = 16$ pairs to check which are the solutions. In this case, we obtain four solutions of the system (4.45), i.e.

$$(-2, 1), (1, 2), \left(-\frac{7}{3}, \frac{1}{3}\right), \left(\frac{8}{3}, -\frac{4}{3}\right). \tag{4.55}$$

Constructing n multiplication matrices and computing their eigenvalues is, however, not always the most efficient way, especially when the number of variables n is larger. First we need to compute eigenvalues of a large number of matrices. Secondly, we need to combine obtained solutions for particular variables x_k, i.e. obtained values of the x_k-coordinates of the points of

$V(I)$, to complete solutions of the whole system. This requires to perform a large number of polynomial evaluations which may be quite time-consuming (in the worst case N^n, where N is the number of solutions of the system (4.1).

To avoid this problem, the second nice property of multiplication matrices $\mathtt{M}_f$ is usually used. This property shows how the eigenvectors of multiplication matrices can be used to find solutions for all coordinates of the points of $V(I)$, i.e. to all variables.

Let us assume that the ideal $I = \langle f_1, \ldots, f_m \rangle$ is a radical ideal, i.e. $I = \sqrt{I}$ [39]. This means that the algebra $A = \mathbb{C}[x_1, \ldots, x_n]/I$ has dimension N, where N is the number of solutions of our system (4.1), i.e. the number of points in $V(I)$. Therefore, the monomial basis B (4.20) of the algebra A consists of N cosets

$$B = \left\{ \left[\mathbf{x}^{\alpha(1)} \right], \ldots, \left[\mathbf{x}^{\alpha(N)} \right] \right\}. \tag{4.56}$$

Then the left eigenvectors of $\mathtt{M}_f$ are related to the solutions of the system of polynomial equations (4.1), i.e. to the points of $V(I)$, in the following way:

Theorem 5. *Let $f \in \mathbb{C}[x_1, \ldots, x_n]$ be a polynomial such that the values $f(\mathbf{p})$ are pairwise distinct for $\mathbf{p} \in V(I)$. Then the left eigenvectors of the multiplication matrix $\mathtt{M}_f$ have the form $\mathbf{v} = \beta \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)} \right)$ for $\beta \in \mathbb{C},\ \beta \neq 0$, and the left eigenspaces have dimension one.*

Proof. Let us first take $\left[\mathbf{x}^{\alpha(j)} \right] \in B$. Then from the definition (4.44) of the multiplication matrix $\mathtt{M}_f$, we have

$$T_f \left(\left[\mathbf{x}^{\alpha(j)} \right] \right) = \left[f \mathbf{x}^{\alpha(j)} \right] = \sum_{i=1}^{N} m_{ij} \left[\mathbf{x}^{\alpha(i)} \right]. \tag{4.57}$$

From the definition of the quotient ring (4.13), we have $[f] = [g]$ iff $f - g \in I$, i.e. iff $f = g + h$ for some $h \in I$. This together with (4.57) give us

$$f \mathbf{x}^{\alpha(j)} = \sum_{i=1}^{N} m_{ij} \mathbf{x}^{\alpha(i)} + h \tag{4.58}$$

for some $h \in I$. For $\mathbf{p} \in V(I)$ we have $h(\mathbf{p}) = 0$ and therefore

$$f(\mathbf{p}) \, \mathbf{p}^{\alpha(j)} = \sum_{i=1}^{N} m_{ij} \mathbf{p}^{\alpha(i)}. \tag{4.59}$$

When we apply this to all $\left[\mathbf{x}^{\alpha(j)} \right] \in B$, we then obtain

$$f(\mathbf{p}) \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)} \right) = \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)} \right) \mathtt{M}_f \tag{4.60}$$

or, equivalently,

$$f(\mathbf{p}) \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)} \right)^{\top} = \mathtt{M}_f^{\top} \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)} \right)^{\top}. \tag{4.61}$$

Since for zero-dimensional ideal I we have $[1] \in B$, it implies that $\left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)}\right) \neq 0$. This means that $\left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)}\right)$ is a left eigenvector of $\mathtt{M}_f$ corresponding to the eigenvalue $f(\mathbf{p})$. Moreover, we are assuming that the values $f(\mathbf{p})$ are distinct for $\mathbf{p} \in V(I)$. Therefore the multiplication matrix $\mathtt{M}_f$ has N distinct eigenvalues and the corresponding 1-dimensional eigenspaces.

The proof of this theorem can be found also in [39]. □

This theorem can be used to find solutions of the system of polynomial equations (4.1) in the following way.

Let $\mathbf{v}$ be a left eigenvector of the multiplication matrix $\mathtt{M}_f$. From Theorem 5 we know that this vector has the form

$$\mathbf{v} = \beta \left(\mathbf{p}^{\alpha(1)}, \ldots, \mathbf{p}^{\alpha(N)}\right), \tag{4.62}$$

where $\beta \in \mathbb{C},\ \beta \neq 0$ and $\mathbf{p} \in V(I)$. Our goal is to find the coordinates of $\mathbf{p} = (a_1, \ldots, a_n)$ assuming that we have the eigenvector $\mathbf{v}$ (4.62).

If $[x_i] \in B$ for all variables $x_i,\ i = 1, \ldots, n$, then it is easy. In this case βa_i is a coordinate of $\mathbf{v}$ for all $i = 1, \ldots, n$. Since $[1] \in B,\ \beta$ is also a coordinate of $\mathbf{v}$. Therefore the x_i-coordinate of the solution of our system, i.e. the solution for x_i, can be found as the ratio of coordinates of $\mathbf{v}$, i.e. $a_i = \frac{\beta a_i}{\beta}$.

If for some i, $[x_i] \notin B$, then the way of obtaining the solution for x_i is not so straightforward; however, it can be proved that we can again obtain solutions for all x_i, $i = 1, \ldots, n$, from the coordinates of the left eigenvectors $\mathbf{v}$ of $\mathtt{M}_f$, see e.g. [39].

Example 9. (Gröbner basis eigenvector method) *For the system of polynomial equations (4.45) from Example 6 we can take one from the two multiplication matrices, e.g.* $\mathtt{M}_x$

$$\mathtt{M}_x = \begin{pmatrix} -\frac{1}{3} & 0 & 1 & 0 \\ -\frac{4}{3} & -\frac{1}{3} & 0 & 1 \\ \frac{10}{3} & -2 & 0 & 0 \\ -\frac{8}{3} & \frac{16}{3} & 0 & 0 \end{pmatrix}, \tag{4.63}$$

and find solutions of the system (4.45) by computing left eigenvectors of $\mathtt{M}_x$ *(4.63) or right eigenvectors of* $\mathtt{M}_x^\top$*. In this way we obtain four eigenvectors*

$$\begin{pmatrix} -2 \\ -2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -\frac{7}{9} \\ -\frac{7}{3} \\ \frac{1}{3} \\ 1 \end{pmatrix} \begin{pmatrix} -\frac{32}{9} \\ \frac{8}{3} \\ -\frac{4}{3} \\ 1 \end{pmatrix}. \tag{4.64}$$

In this case the first coordinate of these eigenvectors corresponds to $[xy] \in B$, the second coordinate to $[x]$, the third to $[y]$ and the fourth to $[1]$. Therefore, solutions for x can be obtained by dividing the second coordinate of the eigenvector by its fourth coordinate, and the corresponding solutions for y by dividing the third coordinate of the eigenvector by the fourth coordinate.

This means that from the first eigenvector (4.64) we obtain $x = \frac{-2}{1} = -2$ and $y = \frac{1}{1} = 1$. In this way we obtain all four solutions of the system (4.45)

$$(-2, 1), (1, 2), \left(-\frac{7}{3}, \frac{1}{3}\right), \left(\frac{8}{3}, -\frac{4}{3}\right). \tag{4.65}$$

The method for solving systems of polynomial equations based on eigenvalues and eigenvectors of multiplication matrices was proposed in [8] and is discussed in several works [109, 110, 128, 129, 130].

Note that this method does not necessarily need to construct Gröbner bases. All what we need is to have representatives of cosets $[fb_i]$, for all $[b_i] \in B$ and some polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$. However, the usual way is to use the remainders of fb_i on the division by a Gröbner basis as these representatives. Then the whole method for solving systems of polynomial equations based on Gröbner bases and multiplication matrices is described by the following algorithm.

Algorithm 5 Gröbner basis Eigenvalue Method

1. Compute a Gröbner basis G w.r.t. some monomial ordering (usually grevlex ordering).
2. Compute the standard monomial basis $B = \left\{[\mathbf{x}^\alpha] : \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$ (using the Gröbner basis G).
3. For a suitable polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$ compute $T_f\left(\left[\mathbf{x}^{\alpha(i)}\right]\right) = \left[f\mathbf{x}^{\alpha(i)}\right] = \overline{f\mathbf{x}^{\alpha(i)}}^G$, for all $\left[\mathbf{x}^{\alpha(i)}\right] \in B$.
4. Using computed $T_f\left(\left[\mathbf{x}^{\alpha(i)}\right]\right)$, create the multiplication matrix $\mathtt{M}_f$ for T_f.
5. Use eigenvalues and eigenvectors of $\mathtt{M}_f$ to find solutions of the input system.

4.3.4 Problems of Gröbner basis methods

Even the Gröbner basis eigenvalue method presented in Section 4.3.3 may lead for some systems of polynomial equations to very complicated computations. It is because the computation of a Gröbner basis may be intractable both in terms of running time and storage space, even for the graded reverse lexicographic ordering and when using the best known variation of Buchberger's algorithm [39].

There are several reasons for this:

1. the coefficients of intermediate polynomials can be very large rational numbers, even if the coefficients of the original polynomials generating the ideal and the coefficients of the final Gröbner basis are relatively small integers, and
2. the total degrees of intermediate polynomials that must be generated during the computation of the Gröbner basis can be quite large.

Both these limitations of Gröbner bases can be seen in the following two examples. The first example comes from [7] and shows the first problem of algorithms for computing Gröbner bases, i.e. the problem with huge coefficients of intermediate polynomials.

Example 10. (Problem of huge coefficients) *Consider the following system of four polynomial equations in three unknowns and with relatively small integer coefficients*

$$\begin{aligned} f_1 &= 8x^2y^2 + 5xy^3 + 3x^3z + x^2yz, \\ f_2 &= x^5 + 2y^3z^2 + 13y^2z^3 + 5yz^4, \\ f_3 &= 8x^3 + 12y^3 + xz^2 + 3, \\ f_4 &= 7x^2y^4 + 18xy^3z^2 + y^3z^3. \end{aligned} \tag{4.66}$$

The reduced Gröbner basis w.r.t. the grevlex ordering and $x > y > z$, for the ideal $I = \langle f_1, f_2, f_3, f_4 \rangle$ generated by these four polynomials in $\mathbb{Q}[x, y, z]$ is very simple

$$\begin{aligned} g_1 &= x, \\ g_2 &= y^3 + 1/4, \\ g_3 &= z^2. \end{aligned} \tag{4.67}$$

However, during the computation of this simple Gröbner basis $G = \{g_1, g_2, g_3\}$ the following intermediate polynomial appears

$$y^3 - 1735906504290451290764747182\ldots, \tag{4.68}$$

where the coefficient in the second term of this polynomial is a rational number with roughly $80,000$ digits in the numerator and also in the denominator. In fact this polynomial has six terms and four of these terms have such huge coefficients.

Computations with huge coefficients may significantly slow down the Gröbner basis algorithm and are very impractical.

The second example was originally presented in [93] and shows the problem of large total degrees of generated polynomials.

Example 11. (Problem of large degrees) *Consider the polynomials*

$$\begin{aligned} x^{n+1} - yz^{n-1}w, \\ xy^{n-1} - z^n, \\ x^nz - y^nw. \end{aligned} \tag{4.69}$$

Mora [93] showed that the reduced Gröbner basis w.r.t. the grevlex ordering and $x > y > z > w$ contains the polynomial

$$z^{n^2+1} - y^{n^2}w \tag{4.70}$$

of total degree $n^2 + 1$.

The bounds on the degrees of the intermediate polynomials in Gröbner basis computation are in general even much larger. Even for the grevlex ordering, which often produces a Gröbner basis with polynomials of the smallest total degree [13], the degrees of intermediate polynomials can be quite large. Mayr and Meyer [104] showed that the construction of a Gröbner basis for an ideal generated by polynomials of degree less than or equal to some d can, in general, involve polynomials of degree proportional to 2^{2^d}.

Fortunately the running time and storage space required to compute a Gröbner basis is "on average" much more feasible than in the worst case.

4.4 Specialized Gröbner basis methods for solving systems of polynomial equations

All Gröbner basis methods presented in previous Section 4.3 are general methods, i.e. they are developed for general systems of polynomial equations and they do not make any special assumptions about these systems. However, sometimes we know some properties of the systems of polynomial equations we are solving, e.g., we know the degree of input equations, the number of solutions or even the structure of these equations. In such a case we can use this information to develop more efficient solvers.

4.4.1 Basic Concepts

In this section we first define classes of systems of polynomial equations for which we can create efficient specific Gröbner basis solvers. To do this we first use the concept of Gröbner traces [140] related to Buchberger based algorithms for computing Gröbner bases. Since we use a method based on linear algebra and G-J eliminations for computing Gröbner bases in our specialized Gröbner basis method, we will later define an Elimination trace as an alternative of this Gröbner trace concept.

In this section we will denote by F a set of polynomials, by $\mathtt{F}$ the matrix representation (4.24) of F, i.e. $\mathtt{F} = \Psi_{T_\succ(F)}(F)$, by $\widetilde{\mathtt{F}}$ the matrix $\mathtt{F}$ eliminated by G-J elimination, and by $\overline{\mathtt{F}}$ the matrix $\mathtt{F}$ extended by some rows corresponding to new polynomials added to F.

Let us now consider some variation of Buchberger's algorithm [39], e.g., Algorithm 1, for constructing a Gröbner basis $G = \{g_1, \ldots, g_s\}$ of the ideal I defined by the input polynomials $F = \{f_1, \ldots, f_r\}$. Such an algorithm first sets $g_i = f_i$ for $i = 1, \ldots, r$, and then using some strategy selects critical pairs of integers (p, q), $p \neq q$. For each critical pair it computes the remainder of the S-polynomial $S(g_p, g_q)$ on division by G, and if this remainder is non-zero, it adds the remainder of the S-polynomial to the basis G.

In order to recover all steps of such an algorithm it is sufficient to know the indices of elements of the S-polynomial, from which each new polynomial $g_i \in G$, $i > r$ was created, and the sequence of polynomials which was used in the division of this S-polynomial. This information is captured by the Gröbner trace introduced in [140].

Definition 16. (Gröbner trace) *Let r and s be some natural numbers such that $0 < r \leq s$. A Gröbner trace T of type (r, s) consists of:*

1. *a sequence of monomials $\mathbf{x}^{\alpha(1)}, \ldots, \mathbf{x}^{\alpha(s)}$ (leading monomials),*
2. *a sequence of pairs of integers $\sigma_{r+1}, \ldots, \sigma_s$, $\sigma_i = (p_i, q_i)$, $p_i < q_i < i$ (critical pairs),*
3. *for each i, $r < i \leq s$, a finite sequence of m_i integers $n_{i,j} < i$, $j = 1, \ldots, m_i$ (simplifier sequence), and*
4. *for each $n_{i,j}$ in the simplifier sequence, a monomial $\mathbf{x}^{\beta(i,j)}$ (multiplier sequence).*

A Gröbner trace T of type (r, s) is a structure that can be used to describe a process of generating s polynomials (not necessarily a Gröbner basis) from the ideal defined by r input polynomials. The process starts with r polynomials $f_1, \ldots, f_r$ and ends with s polynomials $f_1, \ldots, f_s$ with leading monomials $\mathbf{x}^{\alpha(1)}, \ldots, \mathbf{x}^{\alpha(s)}$ from the leading monomial sequence. The i^{th} polynomial f_i, for $i = r+1, \ldots, s$, is created as a reduced S-polynomial $S(f_{p_i}, f_{q_i})$, where p_i and q_i are defined by the critical pairs sequence σ_i. The simplifier sequence specifies which polynomials $f_{n_{i,j}}$, $n_{i,j} < i$, are used to reduce the i^{th} S-polynomial, $i = r+1, \ldots, s$. Finally, the multiplier sequence says which monomials are used to multiply these polynomials $f_{n_{i,j}}$ during the reduction.

This means that the i^{th} generated polynomial f_i, $i = r+1, \ldots s$ is obtained as

$$f_i = S(f_{p_i}, f_{q_i}) - \mathbf{x}^{\beta(i,1)} f_{n_{i,1}} - \mathbf{x}^{\beta(i,2)} f_{n_{i,2}} - \cdots - \mathbf{x}^{\beta(i,m_i)} f_{n_{i,m_i}}. \tag{4.71}$$

In fact, only critical pairs and simplifier sequences are necessary to reconstruct the process of generating polynomials from the ideal. The leading monomials and the multiplier sequences can be deduced from the remaining data; however, these sequences are usually used for checking.

Note that this definition of the Gröbner trace neither implies that it is the trace of Gröbner basis computations, nor that these computations can be carried out for some input polynomials $f_1, \ldots, f_r$, or that the result is a Gröbner basis [140].

However, when the Gröbner trace corresponds to Gröbner basis computations by a Buchberger-like algorithm, then it reconstructs the Gröbner basis without creating all useless critical pairs, i.e. useless S-polynomials (4.10), and without all division tests. Moreover, the same Gröbner trace can be used to reconstruct Gröbner basis computations for many systems with the "same structure", independently of their coefficients. This is a great benefit which may save a large amount of time spent during Gröbner basis computations.

For a given Gröbner trace T of type (r, s) and polynomials $F = \{f_1, \ldots, f_r\}$, such that $LM(f_i) = \mathbf{x}^{\alpha(i)}$, $i = 1, \ldots, r$, where $\mathbf{x}^{\alpha(i)}$ are monomials from the leading monomial sequence of T, the following algorithm reconstructs the process of generating polynomials from the ideal (potentially Gröbner basis computations) or report failure when these computations cannot be carried out for F.

Algorithm 6 Gröbner trace reconstruction algorithm

Input: $F = \{f_1, \ldots, f_r\}$, a Gröbner trace $T = \left(\mathbf{x}^{\alpha(i)}, \sigma_i, n_{i,j}, \mathbf{x}^{\beta(i,j)}\right)$ of type (r, s), such that $LM(f_i) = \mathbf{x}^{\alpha(i)}$, $i = 1, \ldots, r$, and a monomial ordering $\succ$

Output: A basis $G = \{g_1, \ldots, g_s\}$ for $I = \langle f_1, \ldots, f_r \rangle$ (potentially a Gröbner basis), with $F \subset G$, or failure

```
Set g_i := f_i, for i = 1, ..., r
for i = r + 1, ..., s do
    Let σ_i = (p, q)
    Set f := S(g_p, g_q)
    Let the i^th simplifier sequence of T consists of m_i integers n_{i,j}
    for j = 1, ..., m_i do
        if LM(f) = LM(g_{n_{i,j}}) x^{β(i,j)} then
            Set f := LC(g_{n_{i,j}}) f − LC(f) x^{β(i,j)} g_{n_{i,j}}
        else if LM(f) ≺ LM(g_{n_{i,j}}) x^{β(i,j)} then
            Report a redundancy and continue
        else
            Report a failure and STOP
        end if
    end for
    if LM(f) = x^{α(i)} then
        Set g_i := f
    else
        Report a failure and STOP
    end if
end for
```

If this reconstruction algorithm reports a failure, it means that T is not the Gröbner trace associated to a process of generating polynomials starting from $F = \{f_1, \ldots, f_r\}$. A redundancy means that one of the simplifications in the trace has to be omitted.

Definition 17. (Correct Gröbner trace) *We say that the Gröbner trace T of type (r, s) is* correct *for $F = \{f_1, \ldots, f_r\}$, if the Gröbner trace reconstruction algorithm for F, T and some monomial ordering $\succ$ results in a Gröbner basis.*

Note that for one system of polynomials $F = \{f_1, \ldots, f_r\}$ and a fixed monomial ordering there may be many different correct Gröbner traces. It is because in Buchberger based algorithms many different strategies for selecting critical pairs and their reductions can be used and each such strategy can give us a different correct Gröbner trace.

Now let us show how the Gröbner trace looks for a very simple example of generating one new polynomial from the ideal generated by two polynomials.

Example 12. (Gröbner trace) *Let us consider the ideal I generated by two input polynomials*

$$f_1 = x^2 - x + 2y + 1, \tag{4.72}$$
$$f_2 = x - 4. \tag{4.73}$$

This ideal contains polynomial $f_3 = 2y + 13$ with leading monomial y. The polynomial f_3 can be obtained from the two input polynomials and using S-polynomials (4.10) as

$$f_3 = S(f_1, f_2) - 3f_2 = \tag{4.74}$$
$$= \frac{LCM\left(x^2, x\right)}{x^2} f_1 - \frac{LCM\left(x^2, x\right)}{x} f_2 - 3f_2 = \tag{4.75}$$
$$= (x^2 - x + 2y + 1) - x(x - 4) - 3(x - 4) = 2y + 13. \tag{4.76}$$

In this case, the Gröbner trace T related to the process of generating the polynomial f_3 from the polynomials $\{f_1, f_2\}$ is the trace of type $(2, 3)$ consisting of:

1. *the leading monomial sequence $\left(x^2, x, y\right)$,*
2. *the critical pair sequence $\sigma_3 = (1, 2)$,*
3. *the simplifier sequence $n_{3,1} = 2$, i.e. $S(f_1, f_2)$ is reduced by f_2, and*
4. *the multiplier sequence $\mathbf{x}^{\beta(3,1)} = 1$, i.e. the polynomial f_2 is multiplied by a constant.*

Note that the same Gröbner trace T can be used to generate a polynomial with leading monomial y also for another ideal generated by two polynomials with the "same structure" as (4.72) and (4.73) but with different coefficients, e.g., the ideal generated by the polynomials

$$f_1' = x^2 - 2x + y + 4, \tag{4.77}$$
$$f_2' = 2x - 6. \tag{4.78}$$

In this case

$$f_3' = S(f_1', f_2') - \frac{1}{2} f_2' =, \tag{4.79}$$
$$= \frac{LCM\left(x^2, x\right)}{x^2} f_1' - \frac{LCM\left(x^2, x\right)}{2x} f_2' - \frac{1}{2} f_2' = \tag{4.80}$$
$$= (x^2 - 2x + y + 4) - \frac{1}{2} x(2x - 6) - \frac{1}{2}(2x - 6) = y + 7. \tag{4.81}$$

Now let us show how the Gröbner trace looks when it corresponds to Gröbner basis computations performed by Buchberger's algorithm.

Example 13. (Gröbner trace - Buchberger's algorithm) *Let us consider the system of polynomial equations (4.30) form Example 4, i.e. the system*

$$\begin{aligned} f_1 = 2x^2 + y^2 + 3y - 12 &= 0, \\ f_2 = x^2 - y^2 + x + 3y - 4 &= 0. \end{aligned} \tag{4.82}$$

The standard Buchberger's algorithm [39] for the graded reverse lexicographic ordering first sets $G' := G := \{f_1, f_2\}$ *and then creates the S-polynomial (4.10) from these two polynomials* f_1 *and* f_2

$$\begin{aligned} S(f_1, f_2) &= \frac{LCM\left(x^2, x^2\right)}{2x^2} f_1 - \frac{LCM\left(x^2, x^2\right)}{x^2} f_2 = \\ &= x^2 + \frac{1}{2}y^2 + \frac{3}{2}y - 6 - x^2 + y^2 - x - 3y + 4 = \frac{3}{2}y^2 - x - \frac{3}{2}y - 2. \end{aligned} \tag{4.83}$$

Since $\overline{S(f_1, f_2)}^{G'} = \frac{3}{2}y^2 - x - \frac{3}{2}y - 2$*, the algorithm adds this polynomial to* G *and then sets* $G' := G$*.*

Denoting $S(f_1, f_2)$ *by* f_3*, the Buchberger's algorithm tests S-polynomials* $S(f_1, f_3)$ *and* $S(f_2, f_3)$ *in the next loop. In this case*

$$\begin{aligned} S(f_1, f_3) &= \frac{LCM\left(x^2, y^2\right)}{2x^2} f_1 - \frac{LCM\left(x^2, y^2\right)}{\frac{3}{2}y^2} f_3 \\ &= \frac{1}{2}y^4 + \frac{2}{3}x^3 + x^2 y + \frac{3}{2}y^3 + \frac{4}{3}x^2 - 6y^2, \end{aligned} \tag{4.84}$$

$$\begin{aligned} S(f_2, f_3) &= \frac{LCM\left(x^2, y^2\right)}{x^2} f_2 - \frac{LCM\left(x^2, y^2\right)}{\frac{3}{2}y^2} f_3 \\ &= -y^4 + \frac{2}{3}x^3 + x^2 y + xy^2 + 3y^3 + \frac{4}{3}x^2 - 4y^2. \end{aligned} \tag{4.85}$$

Since $\overline{S(f_1, f_3)}^{G'} = 0$ *and* $\overline{S(f_2, f_3)}^{G'} = 0$*, the algorithm in the next step stops and outputs the Gröbner basis* $G = \{f_1, f_2, f_3\}$*.*

In this case the Gröbner trace $T_{grevlex}$ *related to these computations is very simple. It is the trace of type* $(2, 3)$ *consisting of:*

1. *the leading monomial sequence* $\left(x^2, x^2, y^2\right)$*,*
2. *the critical pair sequence (for this simple example consisting of a single pair)* $\sigma_3 = (1, 2)$*,*
3. *the simplifier sequence which is in this case empty (S-polynomial* $S(f_1, f_2)$ *cannot be reduced by polynomials* f_1 *and* f_2*), and*
4. *the multiplier sequence which is again empty because the simplifier sequence is empty.*

In the case of the lexicographic monomial ordering the only S-polynomials that result in non-zero remainders in Buchberger's algorithm [39] are:

$$\begin{aligned} S(f_1, f_2) &= \frac{LCM\left(x^2, x^2\right)}{2x^2} f_1 - \frac{LCM\left(x^2, x^2\right)}{x^2} f_2 \qquad (4.86) \\ &= -x + \frac{3}{2}y^2 - \frac{3}{2}y - 2, \\ S(f_1, f_3) &= \frac{LCM\left(x^2, x\right)}{2x^2} f_1 - \frac{LCM\left(x^2, x\right)}{-x} f_3 \qquad (4.87) \\ &= \frac{3}{2}xy^2 - \frac{3}{2}xy - 2x + \frac{1}{2}y^2 + \frac{3}{2}y - 6. \end{aligned}$$

In this case

$$\begin{aligned} \overline{S(f_1, f_2)}^{G_1'} &= -x + \frac{3}{2}y^2 - \frac{3}{2}y - 2, \qquad (4.88) \\ \overline{S(f_1, f_3)}^{G_2'} &= \frac{9}{4}y^4 - \frac{9}{2}y^3 - \frac{13}{4}y^2 + \frac{15}{2}y - 2, \qquad (4.89) \end{aligned}$$

where $G_1' = \{f_1, f_2\}$ *and* $G_2' = \{f_1, f_2, f_3\}$ *with* $f_3 = \overline{S(f_1, f_2)}^{G_1'}$.

Here

$$\begin{aligned} S(f_1, f_3) &= \frac{3}{2}xy^2 - \frac{3}{2}xy - 2x + \frac{1}{2}y^2 + \frac{3}{2}y - 6 \qquad (4.90) \\ &= \left(\frac{3}{2}y^2 - \frac{3}{2}y - 2\right) f_3 + \left(\frac{9}{4}y^4 - \frac{9}{2}y^3 - \frac{13}{4}y^2 + \frac{15}{2}y - 2\right). \end{aligned}$$

Therefore, the Gröbner trace T_{lex} *related to these computations is the trace of type* $(2, 4)$ *consisting of:*

1. *the leading monomial sequence* $\left(x^2, x^2, x, y^4\right)$,
2. *the critical pair sequence* $\sigma_3 = (1, 2)$ *and* $\sigma_4 = (1, 3)$,
3. *the simplifier sequence which is for* $i = 3$ *empty and for* $i = 4$ *the sequence* $(3, 3, 3)$, *and*
4. *the multiplier sequence which contains for* $n_{4,1} = 3$ *the monomial* y^2, *for* $n_{4,2} = 3$ *the monomial* y, *and for* $n_{4,3} = 3$ *the monomial* 1.

As we have shown in Section 4.2, it is possible to represent a set of polynomials with a matrix (4.24) and to emulate the polynomial division by Gaussian elimination of certain matrices, see Algorithm 2. In this way we can emulate generation of polynomials from an ideal, as well as Buchberger's or another Gröbner basis computation algorithm, by Gaussian or G-J elimination. This fact is used in F4 algorithm for computing Gröbner bases [49].

The Gröbner trace, as defined in Definition 16, contains all data necessary for creating matrices used in the process of generating polynomials from an ideal. However, the process of

creating these matrices from the Gröbner trace data is not so straightforward and the Gröbner trace contains also unnecessary information. Therefore, we will define a special *Elimination trace* describing a process of generating polynomials from an ideal, but based on the matrix representation of polynomials and G-J elimination of such matrices. This trace is not related to Buchberger based algorithms or F4 algorithm and S-polynomials; however, it can be easily filled with data from all these algorithms for computing Gröbner bases.

Definition 18. (Elimination trace) *An Elimination trace T is a structure consisting of:*

1. *a sequence of integers $n_1, \ldots, n_l$ (numbers of rows),*
2. *for each i, $1 \leq i \leq l$ a sequence of monomials $\mathbf{x}^{\alpha(i,1)}, \ldots, \mathbf{x}^{\alpha(i,n_i)}$ (leading monomials), and*
3. *for each i, $1 \leq i \leq l-1$ and for each j, $1 \leq j \leq n_i$ a sequence of monomials $\mathbf{x}^{\beta(1,i,j)}, \ldots, \mathbf{x}^{\beta(m_{i,j},i,j)}$ (multiplier sequence).*

The Elimination trace contains all the data necessary to generate polynomials from the ideal by multiplying already generated polynomials by some monomials and dividing them by performing G-J elimination on matrices in their matrix representation (4.24).

The computations represented by the Elimination trace T start with n_1 polynomials $H_1 = \{h_1, \ldots, h_{n_1}\}$ with leading monomials $LM(h_j) = \mathbf{x}^{\alpha(1,j)}$, $j = 1, \ldots, n_1$. Then the extended set of polynomials $\overline{H}_1$ is created by multiplying polynomials from H_1 by monomials from the corresponding multiplier sequence $M_1 = \left\{\mathbf{x}^{\beta(1,1,j)}, \ldots, \mathbf{x}^{\beta(m_{1,j},1,j)}\right\}$. The extended set contains polynomials $\overline{H}_1 = H_1 \cup \left\{\mathbf{x}^{\beta(1,1,j)}h_j, \ldots, \mathbf{x}^{\beta(m_{1,j},1,j)}h_j \mid h_j \in H_1, j = 1, \ldots, n_1\right\}$. In the next step G-J elimination of the matrix $\overline{\mathtt{H}}_1$ produces the eliminated matrix $\widetilde{\overline{\mathtt{H}}}_1$. If the computations can be carried out for the polynomials in H_1, then after this G-J elimination we obtain sequence H_2 of n_2 non-zero polynomials with matrix representation $\mathtt{H}_2 = \widetilde{\overline{\mathtt{H}}}_1$, and with leading monomials $\mathbf{x}^{\alpha(2,1)}, \ldots, \mathbf{x}^{\alpha(2,n_2)}$. In the next step the computations continue for polynomials H_2. Note that the polynomials in H_2 are ordered according to their leading monomials, i.e. as they come from the G-J elimination.

This means that the sequence $n_1, \ldots, n_l$ in the Elimination trace T stores the numbers of polynomials in $H_1, \ldots, H_l$, i.e. the numbers of non-zero rows in matrices $\mathtt{H}_1, \ldots, \mathtt{H}_l$, where $\mathtt{H}_1$ represents the input polynomials and $\mathtt{H}_i = \widetilde{\overline{\mathtt{H}}}_{i-1}$, for $i = 2, \ldots, l$. The leading monomial sequence $\mathbf{x}^{\alpha(i,1)}, \ldots, \mathbf{x}^{\alpha(i,n_i)}$ stores the leading monomials of polynomials in H_i. Finally, the multiplier sequence stores monomials $\mathbf{x}^{\beta(1,i,j)}, \ldots, \mathbf{x}^{\beta(m_{i,j},i,j)}$ that are used to multiply polynomials $h_j \in H_i$ to get the new polynomials, which are used to extend the set of polynomials H_i. In this way we get the extended set of polynomials $\overline{H}_i = H_i \cup \left\{\mathbf{x}^{\beta(1,i,j)}h_j, \ldots, \mathbf{x}^{\beta(m_{i,j},i,j)}h_j \mid h_j \in H_i, j = 1, \ldots, n_i\right\}$ and its matrix $\overline{\mathtt{H}}_i$, which is eliminated using G-J elimination to get $\mathtt{H}_{i+1} = \widetilde{\overline{\mathtt{H}}}_i$.

The definition of the Elimination trace does not imply that the computations can be carried out for some input polynomials $f_1, \ldots, f_r$ or that the successful execution of the Elimination trace produces a Gröbner basis. In fact, in our method presented later, we will not require that this Elimination trace results in a Gröbner basis.

Similarly, like in the case of the Gröbner trace, we can create an algorithm that for a given Elimination trace T, a monomial ordering $\succ$, and polynomials $F = \{f_1, \ldots, f_r\}$ with $r = n_1$ and $LM(f_k) = \mathbf{x}^{\alpha(1,k)}$, $k = 1, \ldots, n_1$, reconstructs the process of generating polynomials from the ideal $I = \langle F \rangle$ or reports failure when these computations cannot be carried out for F.

Algorithm 7 Elimination trace reconstruction algorithm

Input: $F = \{f_1, \ldots, f_r\}$, a monomial ordering $\succ$, and an Elimination trace T such that $n_1 = r$ and $LT(f_k) = \mathbf{x}^{\alpha(1,k)}$, $k = 1, \ldots, n_1$

Output: A basis $G = \{g_1, \ldots, g_{n_l}\}$ for $I = \langle f_1, \ldots, f_r \rangle$ (potentially a Gröbner basis)

1: Set $H_1 := F$
2: **for** $i = 1, \ldots, l-1$ **do**
3: $\quad \overline{H}_i := H_i \cup \left\{ \mathbf{x}^{\beta(1,i,j)} h_j, \ldots, \mathbf{x}^{\beta(m_{i,j},i,j)} h_j \mid h_j \in H_i, j = 1, \ldots, n_i \right\}$
4: $\quad$ Perform G-J elimination on the matrix $\overline{\mathtt{H}}_i = \Psi_{T_\succ(\overline{H}_i)}(\overline{H}_i)$. Here $T_\succ(\overline{H}_i)$ is the ordered set of monomials of $\overline{H}_i$ w.r.t. the monomial ordering $\succ$. Store the result in $\widetilde{\overline{\mathtt{H}}}_i$.
5: $\quad$ Let H_{i+1} be the set of non-zero polynomials with matrix representation $\mathtt{H}_{i+1} = \widetilde{\overline{\mathtt{H}}}_i$, ordered according to their leading monomials
6: $\quad$ **if** $|H_{i+1}| \neq n_{i+1}$ **then**
7: $\quad\quad$ Report a failure and STOP
8: $\quad$ **else**
9: $\quad\quad$ **for** $j = 1, \ldots, n_{i+1}$ **do**
10: $\quad\quad\quad$ **if** $LM(h_j) \neq \mathbf{x}^{\alpha(i+1,j)}$, where $h_j \in H_{i+1}$ **then**
11: $\quad\quad\quad\quad$ Report a failure and STOP
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end for**
14: $\quad$ **end if**
15: **end for**
16: Set $G := H_l$

Again let's first show how the Elimination trace looks for a very simple example of generating one new polynomial from the ideal defined in Example 12.

Example 14. (Elimination trace) *In this case we have the ideal I generated by two input polynomials*

$$f_1 = x^2 - x + 2y + 1, \tag{4.91}$$

$$f_2 = x - 4 \tag{4.92}$$

and the polynomial $f_3 = 2y + 13 \in I$, which we want to generate from f_1 and f_2. We know that f_3 can be obtained from the two input polynomials as

$$f_3 = S(f_1, f_2) - 3f_2 = f_1 - xf_2 - 3f_2. \tag{4.93}$$

Therefore the Elimination trace T related to the process of generating polynomial f_3 consists of:

1. *the numbers of rows sequence* $(2, 3)$,

2. *the leading monomials sequence, which for* $i = 1$ *contains the monomials* $\left(x^2, x\right)$, *i.e. the leading monomials of the two input polynomials; and for* $i = 2$ *the monomials* $\left(x^2, x, y\right)$, *i.e. the leading monomials of the eliminated polynomials,*

3. *the multiplier sequence, which contains only one monomial* $\mathbf{x}^{\beta(1,1,2)} = x$. *This means that in the first step* ($i = 1$) *we multiply the second polynomial* ($j = 2$) *with* x *and add this polynomial* xf_2 *to the two input polynomials.*

Note that the same Elimination trace T *can be used to generate a polynomial with leading monomial* y *also for another ideal generated by two polynomials with the "same structure" as (4.91) and (4.92) but different coefficients, e.g., the ideal generated by polynomials (4.77) and (4.78).*

Example 15. (Elimination trace - Buchberger's algorithm) *For the system of polynomial equations (4.82) from Example 13, i.e. the system*

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0, \end{aligned} \tag{4.94}$$

the Elimination trace $T_{grevlex}$ *related to the Gröbner basis computations performed in the Buchberger's algorithm [39] w.r.t. the graded reverse lexicographic monomial ordering consists of:*

1. *the numbers of rows sequence* $(2, 2)$,

2. *the leading monomials sequence which for* $i = 1$ *contains the monomials* $\left(x^2, x^2\right)$, *i.e. the leading monomials of the two input polynomials; and for* $i = 2$ *the monomials* $\left(x^2, y^2\right)$, *i.e. the leading monomials of the eliminated polynomials,*

3. *the multiplier sequence which is in this case empty, since in the Buchberger's algorithm the first S-polynomial was obtained only by extracting the two input polynomials multiplied by the appropriate coefficients and such S-polynomial can't be further reduced by the input polynomials.*

For the lexicographic monomial ordering, the Elimination trace T_{lex} *related to the Buchberger's algorithm [39] consists of:*

1. *the numbers of rows sequence* $(2, 2, 5)$,

2. *the leading monomials sequence which for* $i = 1$ *contains the monomials* $\left(x^2, x^2\right)$, *i.e. the leading monomials of the two input polynomials; for* $i = 2$ *the monomials* $\left(x^2, x\right)$, *i.e. the leading monomials of the eliminated polynomials after the first G-J elimination; and for* $i = 3$ *the monomials* $\left(x^2, xy^2, xy, x, y^4\right)$, *i.e. the leading monomials of the eliminated polynomials after the second G-J elimination,*

3. *the multiplier sequence which is for $i = 1$ empty, for $i = 2$ and $j = 1$ again empty, and for $i = 2$ and $j = 2$ contains monomials $\left(x, y, y^2\right)$. It is because the second polynomial in H_2, which corresponds to the reduced S-polynomial $S\left(f_1, f_2\right)$, needs to by multiplied by x, y and y^2 to obtain, after G-J elimination, the reduced S-polynomial $S\left(f_1, f_3\right)$ (4.89) from Example 13.*

Similarly like in the case of the Gröbner trace, we define a correct Elimination trace.

Definition 19. (Correct Elimination trace) *We say that the Elimination trace T is* correct *for $F = \{f_1, \ldots, f_r\}$, if the Elimination trace reconstruction algorithm for F, T and some monomial ordering $\succ$ results in a Gröbner basis.*

This definition implies that the Elimination trace reconstruction algorithm doesn't report a failure, i.e. computations based on the Elimination trace T can be carried out for F and $\succ$, and the result is a Gröbner basis.

Based on the definition of the Gröbner trace and the Elimination trace, we now define systems of polynomial equations which are "in the form" of some system F.

Definition 20. (Systems in the form of F) *Let us consider the system of polynomial equations $F = \{f_1, \ldots, f_m\}$ in which*

$$\begin{aligned} f_1 &= c_{1,1}\mathbf{x}^{\alpha(1,1)} + \cdots + c_{1,i_1}\mathbf{x}^{\alpha(1,i_1)}, \\ f_2 &= c_{2,1}\mathbf{x}^{\alpha(2,1)} + \cdots + c_{2,i_2}\mathbf{x}^{\alpha(2,i_2)}, \\ &\cdots \\ f_m &= c_{m,1}\mathbf{x}^{\alpha(m,1)} + \cdots + c_{m,i_m}\mathbf{x}^{\alpha(m,i_m)}, \end{aligned} \tag{4.95}$$

where $c_{i,j} \in \mathbb{C}$ are some non-zero coefficients and $\mathbf{x}^{\alpha(i,j)} = x_1^{\alpha_1(i,j)} x_2^{\alpha_2(i,j)} \ldots x_n^{\alpha_n(i,j)}$ are some monomials in variables $x_1, \ldots, x_n$.

Let T be a Gröbner/Elimination trace that is correct for F w.r.t. some monomial ordering $\succ$.

Now, consider another system of m polynomial equations $H = \{h_1, \ldots, h_m\}$ of the form

$$\begin{aligned} h_1 &= d_{1,1}\mathbf{x}^{\beta(1,1)} + \cdots + d_{1,k_1}\mathbf{x}^{\beta(1,k_1)}, \\ h_2 &= d_{2,1}\mathbf{x}^{\beta(2,1)} + \cdots + d_{2,k_2}\mathbf{x}^{\beta(2,k_2)}, \\ &\cdots \\ h_m &= d_{m,1}\mathbf{x}^{\beta(m,1)} + \cdots + d_{m,k_m}\mathbf{x}^{\beta(m,k_m)}, \end{aligned} \tag{4.96}$$

where $d_{i,j} \in \mathbb{C}$, $d_{i,j} \neq 0$ are some non-zero coefficients and $\mathbf{x}^{\beta(i,j)}$ are monomials in variables $x_1, \ldots, x_n$.

Then we say that the system H (4.96) is in the form of the system F (4.95) w.r.t. the monomial ordering $\succ$ and the trace T if

1. $\left\{\mathbf{x}^{\beta(j,1)},\ldots,\mathbf{x}^{\beta(j,k_j)}\right\} \subset \left\{\mathbf{x}^{\alpha(j,1)},\ldots,\mathbf{x}^{\alpha(j,i_j)}\right\}$, *for* $j = 1,\ldots,m$, *i.e. the polynomial* h_j, $j = 1,\ldots,m$, *contains only monomials* $\left\{\mathbf{x}^{\alpha(j,1)},\ldots,\mathbf{x}^{\alpha(j,i_j)}\right\}$, *while some of these monomials may have zero coefficients in* h_j.

2. *The Gröbner/Elimination trace* T *is a correct Gröbner/Elimination trace for the system* H *(4.96) and the monomial ordering* $\succ$.

Note that the relation "being in the form of" is always connected to some trace T and it is not symmetric, i.e. if the system H is in the form of F then the system F may not be in the form of H because F may contain more non-zero monomials. Moreover, one system may be in the form of another system w.r.t. one monomial ordering but not w.r.t. another monomial ordering.

Example 16. (System in the form of F) *Let us consider the system* F_1 *representing the intersection of an axis-aligned ellipse and an axis-aligned hyperbola in* $\mathbb{C}^2$ *of the form*

$$\begin{aligned} a_0x^2 + a_1x + a_2y^2 + a_3y + a_4 &= 0, \\ b_0x^2 + b_1x - b_2y^2 + b_3y + b_4 &= 0, \end{aligned} \tag{4.97}$$

where all $a_0,\ldots,a_4,b_0,\ldots,b_4 \in \mathbb{C}$ *are some generic non-zero coefficients.*

Consider the system F_2 *from the previous Example 13, i.e. the system*

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0. \end{aligned} \tag{4.98}$$

It can be easily shown that both Gröbner traces $T_{grevlex}$ w.r.t. the graded reverse lexicographic ordering and T_{lex} w.r.t. the lexicographic ordering, computed in Example 13, as well as the Elimination traces computed in Example 15 are correct traces for both these systems, i.e. the system F_1 (4.97) with some generic non-zero coefficients and the system F_2 (4.98).

Since the system F_2 (4.98) contains only monomials which are contained in the system F_1 (4.97), i.e. $\left\{x^2,y^2,y,1\right\} \subset \left\{x^2,y^2,x,y,1\right\}$ and $\left\{x^2,y^2,x,y,1\right\} \subset \left\{x^2,y^2,x,y,1\right\}$, we can say that the system F_2 (4.98) is in the form of F_1 (4.97) w.r.t. the lexicographic as well as the graded reverse lexicographic ordering. However, the system F_1 is not in the form of F_2 because F_1 (4.97) contains in the first polynomial the monomial x, which is not included in the first polynomial of the system F_2 (4.98).

Example 17. (System in the form of F)

Now consider the system

$$\begin{aligned} 2x^2 - 2x + y^2 + 4y + \frac{1}{2} &= 0, \\ 2x^2 - 2x - y^2 + 4y + \frac{1}{2} &= 0, \end{aligned} \tag{4.99}$$

representing again the intersection of the ellipse and the hyperbola, see Figure 4.2.

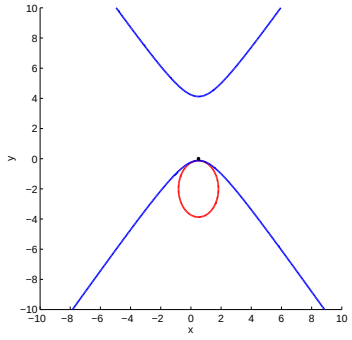

Figure 4.2: The intersection of the ellipse $2x^2 - 2x + y^2 + 4y + \frac{1}{2} = 0$ and the hyperbola $2x^2 - 2x - y^2 + 4y + \frac{1}{2} = 0$.

For this system the first S-polynomial computed using the Buchberger's algorithm [39] has the form

$$S(f_1, f_2) = f_1 - f_2 = 2y^2. \tag{4.100}$$

This is the only S-polynomial with non-zero remainder constructed during the computations of the Gröbner basis w.r.t. the graded reverse as well as the lexicographic ordering. Therefore, the Gröbner trace $T_{grevlex}$ w.r.t. the graded reverse lexicographic ordering computed in Example 13 is correct for this system (4.99). However, the Gröbner trace T_{lex} w.r.t. the lexicographic ordering is not correct for (4.99). This also holds true for the Elimination traces constructed using the Buchberger's algorithm [39].

Therefore, the system (4.99) is in the form of F_1 (4.97) w.r.t. the graded reverse lexicographic ordering; however, it is not in the form of F_1 (4.97) w.r.t. the lexicographic ordering.

Example 18. (System in the form of F) *An example of a system that contains only monomials that are contained in the system F_1 (4.97) but that is not in the form of F_1 either w.r.t. the graded reverse lexicographic ordering or w.r.t. the lexicographic ordering is the system in which the ellipse degenerates into a line, e.g. the system*

$$\begin{aligned} 2y - 4 &= 0, \\ x^2 - 2x - y^2 + 4y + 1 &= 0. \end{aligned} \tag{4.101}$$

In this case the leading monomial of the first polynomial $2y - 4 = 0$ is different from the leading monomial of the polynomial $a_0x^2 + a_1xy + a_2y^2 + a_3x + a_4y + a_5 = 0$ with some non-zero coefficients $a_0, \ldots, a_5 \in \mathbb{C}$ w.r.t. the graded reverse lexicographic ordering as well as the lexicographic ordering. Therefore these two systems, (4.101) and (4.97), have different Gröbner as well as Elimination traces w.r.t. these monomial orderings.

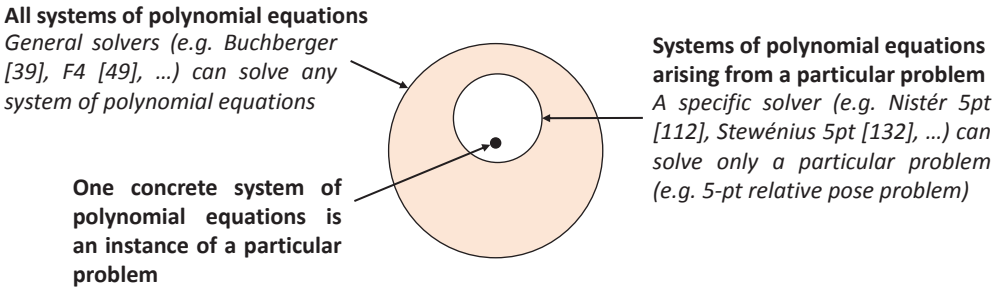

Figure 4.3: Illustration of the relation between general and specific solvers.

4.4.2 Motivation of specialized Gröbner basis method

In practical situations, when solving a particular problem, we often encounter systems of "similar form" or in fact systems which are in the form of one or only few "representing systems" (4.95) according to Definition 20. For example, coefficients c_{ij} in system (4.95) can arise from measurements in an image, and thus every new measurement produces a new system of equations. The common property of all such systems is that the monomials appearing in these systems can only disappear if some c_{ij} becomes zero, but no additional monomial can appear there.

System of polynomial equations, such as (4.95), and the issue of finding all solutions of this system for arbitrary coefficients $c_{i,j}$ will be called a *particular problem* in the following. The system which we obtain after substituting concrete values for c_{ij} will be called an *instance of a particular problem*, see Figure 4.3.

Every particular problem can be also defined in its matrix form $\mathtt{M}\mathbf{X} = \mathbf{0}$ using the matrix representation of polynomials $\mathtt{M} = \Psi_{\mathbf{X}}(F)$ (4.24) by fixing the vector of monomials $\mathbf{X}$ and the order of polynomials in matrix $\mathtt{M}$. An instance of a particular problem is then obtained by filling concrete values in $\mathtt{M}$.

General algorithms for constructing Gröbner bases presented in Section 4.3 work for every instance of every problem but can't exploit any specific properties of a given problem to be more computationally efficient. We strive to create here far more efficient algorithms for particular problems or even for some subset of instances of a particular problem.

Imagine, for instance, that we are interested in designing an algorithm working only for one instance of a particular problem, i.e. only for one system of equations with concrete coefficients. Such an algorithm does not have to perform a general procedure for Gröbner basis construction. In fact, it may construct only those S-polynomials that are necessary and avoid generation and testing all of those S-polynomials which reduce to zero. This means that this algorithm may follow the Gröbner trace reconstruction algorithm, i.e. Algorithm 6, for some correct Gröbner trace of this system.

To give another example, we can consider a system of linear polynomial equations. Here, we can use Gaussian elimination to construct its Gröbner basis. Now if the system is known, one can create a "template" to remember which rows in the matrix representation of this system can be used in which step of the elimination to avoid division by zero. A template for matrix

of order (n, n) stores n pivot positions. Since pivots are known, effort spent on finding them is saved. These savings may be negligible when solving small linear systems but considerable when solving large systems.

It has been observed that for some problems arising in computer vision, as well as in other fields, all "interesting" instances of these problems can be solved by following one, or only a very small number of "solution templates", e.g., Gröbner or Elimination traces. It is because all "interesting" instances are in the form of one or a small number of "representing systems" (4.95). This allows to consider only one (or a few) of the interesting instances (actually any of them) and use it to construct a "solution template" based, for example, on an Elimination trace of this instance.

Sometimes we will call all instances of a particular problem, i.e. all systems with concrete coefficients $c_{i,j}$, which are in the form of one system F (4.95), the *consistent* instances and the system F the *representing* system of these consistent instances.

Let us next discuss three applications. Two of these applications lead to problems that can solve many instances with a single "solution template". It is because systems resulting from these problems are in the form of only one system (4.95). The other application presents a problem for which each instance has a completely different form, different Gröbner and Elimination trace, and therefore different "solution template".

Example 19. (Intersection of an ellipse and a hyperbola) *The problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola in $\mathbb{C}^2$ is an example where all interesting "non-degenerate" instances are in the form of one system only. In this case each "non-degenerate" instance of this problem results in a system of polynomial equations of the following form*

$$\begin{aligned} a_0x^2 + a_1x + a_2y^2 + a_3y + a_4 &= 0, \\ b_0x^2 + b_1x - b_2y^2 + b_3y + b_4 &= 0, \end{aligned} \tag{4.102}$$

for some coefficients $a_0, \ldots, a_4, b_0, \ldots, b_4 \in \mathbb{C}$, where a_0, a_2 are both non-zero and with the same sign and the same holds true for b_0 and b_2.

Here a "non-degenerate" instance means a system where neither the ellipse nor the hyperbola degenerate to a line or a point. Moreover, if we take a system (4.102) with some generic (e.g. randomly generated) non-zero coefficients $a_0, \ldots, a_4, b_0, \ldots, b_4 \neq 0$, then all "non-degenerate" instances are in the form of this system, according to Definition 20, and this system can be used to find a common Gröbner or Elimination trace of all these instances.

Instances in which the ellipse or/and the hyperbola degenerate to a line or a point are for this problem "degenerate instances" and we are usually not interested in them or we solve them as a separate problem. These "degenerate instances" do not fulfill 2^{nd} point in Definition 20, i.e. the Gröbner or Elimination trace which is correct for some "non-degenerate" instance is not correct for these "degenerate" instances, and therefore these "degenerate" instances are not in the form of the system (4.102) with some random non-zero coefficients.

When the number of variables, equations or their degrees depend on the input data or input measurements, some "interesting" instances of the problem may have different form, and therefore different "solution templates". An example of such problem is the graph coloring.

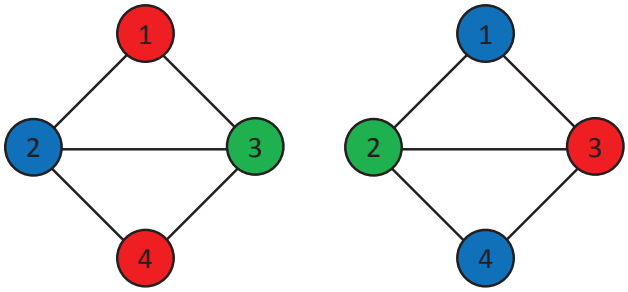

Figure 4.4: Examples of admissible 3-colorings.

Example 20. (Graph coloring) *A k-coloring of a graph $G = (V, E)$ is a labeling of the graph's vertices with colors taken from k colors such that no two vertices sharing the same edge have the same color. Figure 4.4 shows two examples of a 3-coloring of a graph.*

The problem of finding all k-colorings for a given graph can be reduced to the problem of finding all solutions of a system of polynomial equations. Then each possible k-coloring of the graph corresponds to one solution of this system.

In this system the number of variables corresponds to the number of vertices of the graph, i.e. variables represent vertices, the number of equations depends on the number of edges of the graph and the degree of these equations depends on the number of colors. Different colors correspond to different k^{th} roots of unity. Therefore the condition that the vertex represented by the variable x_i should be labeled with one of k-colors is encoded by the polynomial

$$x_i^k - 1. \tag{4.103}$$

The polynomial

$$\frac{x_i^k - x_j^k}{x_i - x_j} = x_i^{k-1} + x_i^{k-2}x_j + x_i^{k-3}x_j^2 + \cdots + x_i x_j^{k-2} + x_j^{k-1} \tag{4.104}$$

encodes the condition that the vertices corresponding to the variables x_i and x_j must have different colors.

Therefore the system representing the problem of finding all k-colorings for a graph $G = (V, E)$ with n vertices, i.e. $|V| = n$, and m edges, i.e. $|E| = m$, consists of $n+m$ equations in n variables. The first n equations have the form (4.103) and the next m equations the form (4.104).

For example, the possible 3-colorings of the graph in Figure 4.4 correspond to the solutions

of the following system of polynomial equations

$$\begin{aligned} x_1^3 - 1 &= 0, \\ x_2^3 - 1 &= 0, \\ x_3^3 - 1 &= 0, \\ x_4^3 - 1 &= 0, \\ x_1^2 + x_1x_2 + x_2^2 &= 0, \\ x_1^2 + x_1x_3 + x_3^2 &= 0, \\ x_2^2 + x_2x_3 + x_3^2 &= 0, \\ x_2^2 + x_2x_4 + x_4^2 &= 0, \\ x_3^2 + x_3x_4 + x_4^2 &= 0. \end{aligned} \tag{4.105}$$

This system has six solutions, one of which is $x_1 = z, x_2 = z^2, x_3 = z^3$ *and* $x_4 = z$*, where* $z = \frac{1}{2}\left(-1 + \sqrt{3}i\right)$*. This solution says that the vertices* 1 *and* 4 *have the same color corresponding to* $\frac{1}{2}\left(-1 + \sqrt{3}i\right)$ *and the vertices* 2 *and* 3 *have different colors.*

The problem of finding all k*-colorings for a given graph is a typical example of a problem in which each instance, each graph, results in a different system of polynomial equations with a different number of equations and variables. Therefore a Gröbner trace that is correct for one instance, i.e. for one graph, is not correct for another graph. Thus the instances corresponding to different graphs have different "solution templates".*

Fortunately, many problems in computer vision, as well as in other fields, have the nice property that all "interesting" instances of the problem or instances which are most common in real applications are in the form of one or only few systems and therefore can be solved using only one or few "solution templates".

Example 21. (5-point relative pose problem) *An example of such a problem is the well known and important problem of estimating the relative position and orientation of two calibrated cameras from five image point correspondences, see Figure 4.5.*

In the case of this 5-point relative pose problem, the epipolar constraint

$$\mathbf{x}_i'^{\top} \mathtt{E}\, \mathbf{x}_i = 0 \tag{4.106}$$

for $i = 1, \ldots, 5$ *results in five linear equations in nine unknowns of the essential matrix* $\mathtt{E}$ *and with coefficients arising from coordinates of image point correspondences* $\mathbf{x}_i$ *and* $\mathbf{x}_i'$*. Moreover, since the essential matrix* $\mathtt{E}$ *is a singular matrix with two equal singular values, we have another ten higher order algebraic equations, i.e. the rank constraint*

$$\det(\mathtt{E}) = 0 \tag{4.107}$$

and the trace constraint

$$2\,\mathtt{E}\mathtt{E}^{\top}\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^{\top})\mathtt{E} = \mathtt{0}, \tag{4.108}$$

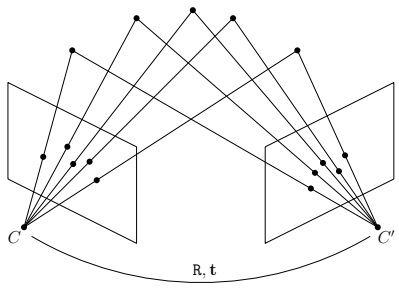

Figure 4.5: The illustration of the 5-point relative pose problem.

that do not depend on particular measurements.

For each "non-degenerate" instance, i.e. for image points $\mathbf{x}_i$ *and* $\mathbf{x}'_i$, $i = 1, \ldots, 5$ *that are not collinear and two or more points do not coincide, these equations are in the form of equations (4.106), (4.107), (4.108) arising from five random image correspondences* $\mathbf{x}_i$ *and* $\mathbf{x}'_i$*. Therefore all "non-degenerate" instances of the 5-point relative pose problem have the same "solutions template" as this "random" instance.*

Several nice properties of systems in the form of some system F arise from Definition 20.

Theorem 6. *Let's consider a system of polynomial equations* F *(4.95). Let* G *be the reduced Gröbner basis w.r.t. a monomial ordering* $\succ$ *of the ideal* $I = \langle F \rangle$*. Then for every system* F' *in the form of system* F *and the reduced Gröbner basis* G' *w.r.t. a monomial ordering* $\succ$ *of the ideal* $I' = \langle F' \rangle$*, it holds:*

1. *The system* F' *has the same number of solutions as* F*.*
2. *The reduced Gröbner basis* G' *w.r.t. the monomial ordering* $\succ$ *has the same leading monomials as the reduced Gröbner basis* G *w.r.t. this monomial ordering* $\succ$*.*
3. *The standard monomial basis* B' *(4.20) of the quotient ring* $A' = \mathbb{C}[x_1, \ldots, x_n]/I'$ *consists of the same cosets (same monomials) as the standard monomial basis* B *(4.20) of the quotient ring* $A = \mathbb{C}[x_1, \ldots, x_n]/I$*.*
4. *For a fixed strategy of generating polynomials from the ideal and a fixed monomial ordering* $\succ$ *the "path" from* F' *to* G' *is the same as the "path" from* F *to* G*. This means that for obtaining all polynomials from* G' *the same polynomial combinations (or monomial multiples) of the initial polynomials* F' *need to be generated as in the case of* G *and* F*.*

Proof. All these properties result directly from Definition 20.

From Definition 20 we know that the Gröbner/Elimination trace T that is correct for F w.r.t. the monomial ordering $\succ$ is correct also for F'. Therefore, the reduced Gröbner bases G and G', which result from the correct Gröbner/Elimination trace T, have the same leading monomials.

Since for the monomial ideal $\langle LT(I) \rangle$ and for the Gröbner basis $G = \{g_1, \ldots, g_s\}$ of I it holds $\langle LT(I) \rangle = \langle LT(g_1), \ldots, LT(g_s) \rangle$ and since the Gröbner bases G and G' have the same leading monomials, we have $\langle LT(I) \rangle = \langle LT(I') \rangle$.

Therefore the standard monomial basis B of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$, defined as $B = \{[\mathbf{x}^\alpha] : \mathbf{x}^\alpha \notin \langle LT(I) \rangle\}$, consists of the same cosets as B', i.e. $B = B'$. This also means that F and F' have the same number of solutions.

If we follow the Gröbner/Elimination trace T that is correct for F as well as F', then for obtaining all polynomials from G', the same polynomial combinations (or monomial multiples) of the initial polynomials F' are generated as in the case of G and F. □

Example 22. (Properties of the intersection of an ellipse and a hyperbola) *In the case of the problem of finding the intersection of an axis-aligned ellipse and an axis-aligned hyperbola in $\mathbb{C}^2$ from Example 19 and all its "non-degenerate" instances, i.e. instances where neither the ellipse nor the hyperbola degenerate to a line or a point, represented by the system of polynomial equations (4.102) with some non-zero coefficients, we have:*

1. *The number of solutions of each "non-degenerate" instance of this problem is four.*
2. *The Gröbner basis of the ideal defined by system (4.102) w.r.t. the graded reverse lexicographic ordering with $x > y$ consists of two polynomials of the following form*

$$G = \left\{c_0 y^2 + c_1 x + c_2 y + c_3, d_0 x^2 + d_1 x + d_2 y + d_3\right\}, \tag{4.109}$$

   where $c_0, \ldots, c_3, d_0, \ldots, d_3 \in \mathbb{C}$ are some coefficients with $c_0 \neq 0$ and $d_0 \neq 0$.
3. *The standard monomial basis B (4.20) of the quotient ring $A = \mathbb{C}[x, y]/I$ consists for each "non-degenerate" instance of the cosets*

$$B = \{[xy], [x], [y], [1]\}. \tag{4.110}$$

4. *Let's denote the two input polynomial equations in (4.102) by f_1 and f_2, i.e. $f_1 = a_0 x^2 + a_1 x + a_2 y^2 + a_3 y + a_4 = 0$ and $f_2 = b_0 x^2 + b_1 x - b_2 y^2 + b_3 y + b_4 = 0$. Then we can obtain the polynomials from the grevlex Gröbner basis G (4.109) as simple linear combinations*

$$g_1 = b_0 f_1 - a_0 f_2, \tag{4.111}$$

$$g_2 = b_2 f_1 + a_2 f_2. \tag{4.112}$$

As we have already mentioned, many problems have the nice property that all "interesting instances" of the problem are in the form of one system of polynomial equations (4.95). Therefore to solve these problems it is sufficient to find a solver that can solve all systems of polynomial equations that are in the form of this "representing system" (4.95).

Except for this nice property, problems appearing in computer vision have also several requirements:

1. These problems are usually parts of some large systems which require high or even real-time performance, e.g., structure-from-motion pipelines or recognition systems.

2. Most of the time the input measurements are contaminated by a large number of outliers (incorrect inputs). Therefore these problems have to be solved for many different inputs to find "the best solution", i.e. the solution consistent with as many measurements as possible. Usually the well known RANSAC paradigm [50] is used for this purpose.

This means that for computer vision problems we usually need to be able to solve many instances of the particular problem, all in the form of only one system (4.95), and we need to do it very fast. In such a case, standard methods for solving general systems of polynomial equations presented in Section 4.3 may be inefficient.

Considering all presented properties and requirements, we have developed a method for efficiently solving systems of polynomial equations which are in the form of one "representing system" F (4.95). In this method we use the fact that for a fixed monomial ordering and a fixed strategy for selecting and reducing S-polynomials (resp. a fixed strategy for generating polynomials from the ideal) all such systems have the same Gröbner (resp. Elimination) trace as the "representing system" F. Therefore this Gröbner (resp. Elimination) trace can be found once in advance for F and then used to efficiently solve all systems in the form of F. Moreover, in our method we use the fact that this trace can be found in some finite prime field $\mathbb{Z}_p$, i.e. for F with coefficients from $\mathbb{Z}_p$, where p is a sufficiently large "lucky" prime number [140].

Let F' be a system (4.95) with some generic non-zero coefficients $c_{i,j} \in \mathbb{C}$. By generic we mean that randomly chosen coefficients $c_{i,j}$ in (4.95) will give a system in the form of F' with the probability equal to 1. Let F' be a "representing system" of all instances which we are solving, i.e. all considered instances are in the form of the system F'. This means that for a fixed strategy and a fixed monomial ordering, the Gröbner (resp. Elimination) trace T' obtained for F' is the same as the Gröbner (resp. Elimination) trace T obtained for any instance of F (4.95) which we want to solve.

Let $F'_{\mathbb{Z}_p}$ be the system F (4.95) with some random non-zero coefficients $c_{i,j} \in \mathbb{Z}_p$ and let $T'_{\mathbb{Z}_p}$ be the Gröbner (resp. Elimination) trace obtained for $F'_{\mathbb{Z}_p}$ and the same strategy and monomial ordering as it was used for obtaining T' and T.

In [140] and [72] it has been proven that for a sufficiently large prime number p the probability that the Gröbner (resp. Elimination) trace $T'_{\mathbb{Z}_p}$ is the same as T' is close to 1. Therefore, it is usually sufficient to find the Gröbner (resp. Elimination) trace for $F'_{\mathbb{Z}_p}$ with coefficients from $\mathbb{Z}_p$ only once and then use this trace to efficiently solve all instances in the form of the system F (4.95) in $\mathbb{C}$. This is the main idea of our specialized Gröbner basis method.

Our specialized method is based on the Gröbner basis eigenvalue method described in Section 4.3.3. However, it can be easily modified also to the remaining two Gröbner basis methods presented in Sections 4.3.1 and 4.3.2 as we will discuss in Section 4.4.6.

The main difference between the proposed specialized Gröbner basis method and the general Gröbner basis methods presented in Section 4.3 is that the specialized method uses the structure of the system of polynomial equations representing a particular problem to design an efficient specific solver for this problem. For this purpose the specialized method does some preprocessing and computations common to all considered instances of the particular problem. In fact this specialized Gröbner basis method consists of two phases:

1. **Offline phase:** In the first phase we are trying to apply convenient properties of systems

of polynomial equations that are in the form of some "representing system" F to design an efficient specific solver for solving all these systems. In this phase we find an Elimination trace T that is common to all systems in the form of F. We do computations in some finite prime field $\mathbb{Z}_p$. Based on the Elimination trace T an efficient specific solver is constructed. For a particular problem and its set of "consistent" instances this phase needs to be performed only once and therefore we will call it the "offline phase".

2. **Online phase:** In the second "online" phase, the specific solver is used to efficiently solve concrete instances of the particular problem. This efficient specific solver is not general and solves only systems of polynomial equations in the form of F; however, it is faster than a general solver.

For problems for which "interesting" instances are in the form of several "representing systems", we can create an efficient specific solver for each "representing system" and then join these solvers into one solver solving the "whole" problem, e.g., joining a solver for points lying on a plane or a line with a solver for points in a general position [28].

Our specialized Gröbner basis method follows and extends ideas of the method proposed and used by Stewénius to manually create Gröbner basis solvers for some computer vision problems [131, 132, 135, 134].

Next we will describe two phases of our method in more detail.

4.4.3 Offline phase

Let us assume that we have a particular problem, and for this problem one set S of consistent instances, i.e. set of systems which are all in the form of one "representing system" of polynomial equations F. Our goal is to find a solver solving all instances from S. Usually this set S is a set of some generic "non-degenerate" instances of a given problem.

As we have already mentioned, in the offline phase we apply convenient properties of systems of polynomial equations that are in the form of one "representing system" F to design a specific solver for solving all these systems and in this way all instances of the particular problem belonging to S.

A straightforward approach to design such a specific solver is to run some known algorithm for computing Gröbner bases, e.g. the Buchberger's [39] or the F4 algorithm [49], from this algorithm construct a Gröbner or an Elimination trace, and use this trace to construct an efficient solver, efficient "solution template", for all instances from S.

Since we use the Gröbner basis eigenvalue method, see Section 4.3.3, that does not necessarily need a complete Gröbner basis, we have decided not to use existing algorithms for computing Gröbner bases. Instead, we only identify some parameters of F and use them for constructing an Elimination trace T, that contains all polynomials needed for constructing a multiplication matrix (4.44). Such a trace doesn't have to be correct, i.e. it does not have to necessarily produce a complete Gröbner basis.

Since for a sufficiently large prime number p, this trace T is, with large probability, equivalent to the trace constructed for $F_{\mathbb{Z}_p}$ with coefficients from a finite prime field $\mathbb{Z}_p$ [140, 72], we do all computations in this phase in $\mathbb{Z}_p$. In such a field, exact arithmetic can be used and numbers can

be represented in a simple and efficient way. This suppresses two main problems of standard Gröbner basis methods, i.e. problems with expensive growth of coefficients and rounding off problems. Moreover, using these finite prime fields speeds up computations and minimizes memory requirements.

For a particular problem and its instances that are in the form of one "representing system" F, this offline phase needs to be performed only once. The offline phase consists of two steps: the identification of parameters of F and the design of an Elimination trace which is then used for constructing a multiplication matrix. Next we will describe each from these steps in detail.

Identification of basic parameters

In the first step, the basic parameters of the system F representing all considered instances of the particular problem are identified. This means that we check if there exists a finite number of solutions of F, and if yes, how many solutions this "representing system", and therefore also all considered instances have, how "hard" it is to obtain the Gröbner basis, and how does the standard monomial basis B (4.20) of the quotient ring A look like.

This information can be obtained using a general algorithm for computing Gröbner bases and the basis B [39]. However, as we have already mentioned, in this case the input system of polynomial equations is the system in the form of the "representing system" F (4.95), but with non-zero coefficients $c_{i,j}$ from some finite prime field $\mathbb{Z}_p$, where p is a suitably chosen prime number.

By a suitably chosen prime number p we mean a prime number such that the probability of obtaining "degenerate" zero coefficient, i.e. coefficient which is zero in $\mathbb{Z}_p$ but non-zero in $\mathbb{Q}$, is very low. This means that also the probability of obtaining different Gröbner/Elimination traces for F in $\mathbb{Z}_p$ and in $\mathbb{Q}$ is very low. In fact there is always such a prime number p [7]. In our practical situations a three- or four-digit prime number was sufficient.

Nevertheless, for smaller prime numbers degenerate situations may sometimes appear. Therefore to avoid this, it is sometimes better to repeat the computations for several instances of the problem, i.e. several systems of polynomial equations from S with different coefficients from $\mathbb{Z}_p$. If all parameters of the problem remain stable for all these instances, then these parameters are generically equivalent to the parameters of the original "representing system" F with coefficients from $\mathbb{Q}$ [140], and therefore also to the parameters of all systems from S.

Construction of the multiplication matrix

After identifying the above-mentioned parameters of the considered instances of the particular problem, we can use this information to design a specific solver for solving these instances. This means that we design a solver for solving all systems of polynomial equations in the form of F representing these instances, see Definition 20.

Such a solver can be based on different methods for solving systems of polynomial equations. Since for many systems of polynomial equations the most efficient Gröbner basis method is the Gröbner basis eigenvalue method presented in Section 4.3.3, we have decided to create a solver based on a modified version of this method. In Section 4.4.6 we will show that sometimes the

remaining two Gröbner basis methods presented in Sections 4.3.1 and 4.3.2 may be useful, and that the presented specialized method can be easily modified to produce solvers based also on these methods.

The standard Gröbner basis eigenvalue method presented in Section 4.3.3, i.e. Algorithm 5, requires to compute a complete Gröbner basis G w.r.t. some monomial ordering, a standard monomial basis B (4.20) of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$, and to compute $T_f\left(\left[\mathbf{x}^{\alpha(i)}\right]\right) = \left[\mathbf{x}^{\alpha(i)} f\right] = \left[\overline{f\mathbf{x}^{\alpha(i)}}^{G}\right]$, for all $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ and some polynomial f, i.e. to perform polynomial division by a Gröbner basis G. In this way the multiplication matrix $\mathtt{M}_f$ (4.44) is created.

This method for creating multiplication matrices uses the fact that the remainder of a polynomial f on division by a Gröbner basis G, i.e. $\overline{f}^{G}$, can be used as a representative of the coset $[f] \in A$. However, next we will show that the construction of multiplication matrices does not necessarily need Gröbner bases. It is sufficient to have some representatives of cosets $[fb_i]$, for all basis cosets $[b_i] \in B$ and some polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$. These representatives do not necessarily need to be constructed using a Gröbner basis.

Next we propose a method for constructing multiplication matrices that requires only the basis B of the quotient ring A, or only the number of solutions of the system, and does not explicitly construct a complete Gröbner basis nor perform polynomial division. This method needs to generate polynomials from the ideal I with leading monomials from the set $\left(\mathbf{x}^{\beta} \cdot B\right) \setminus B$ and the remaining monomials from B, where $\mathbf{x}^{\beta}$ is the monomial for which we are creating the multiplication matrix $\mathtt{M}_{\mathbf{x}^{\beta}}$, $\setminus$ stands for the set difference, and by monomials from B we mean monomials $\mathbf{x}^{\alpha}$ such that cosets $[\mathbf{x}^{\alpha}] \in B$. Construction of these polynomials may be in general as difficult as generating a Gröbner basis, however for some systems it needs to generate less polynomials than necessary for constructing a complete Gröbner basis, see Example 23.

Note that we are considering a multiplication matrix $\mathtt{M}_{\mathbf{x}^{\beta}}$ for a monomial $\mathbf{x}^{\beta}$ and not $\mathtt{M}_f$ for a general polynomial f. It is because construction of such a multiplication matrix is for most problems sufficient and is usually simpler than construction of a multiplication matrix for a general polynomial f. Usually it is even sufficient to construct a multiplication matrix $\mathtt{M}_{x_k}$ for a variable x_k, since such a matrix is quite sparse and therefore simpler to compute than a general $\mathtt{M}_f$.

Let us assume that a basis B of the quotient ring A is known. As we have already mentioned in the previous section, in Theorem 6, this basis is the same for all systems in the form of one "representing system" F and can be computed in the preprocessing phase.

Let this basis B consists of N monomial cosets

$$B = \left\{\left[\mathbf{x}^{\alpha(1)}\right], \ldots, \left[\mathbf{x}^{\alpha(N)}\right]\right\} \tag{4.113}$$

and let the monomial for which we are creating the multiplication matrix be $\mathbf{x}^{\beta}$. Then for constructing the multiplication matrix $\mathtt{M}_{\mathbf{x}^{\beta}}$ we need to compute $T_{\mathbf{x}^{\beta}}\left(\left[\mathbf{x}^{\alpha(i)}\right]\right) = \left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right]$ for all $\left[\mathbf{x}^{\alpha(i)}\right] \in B$. This means that we need to find the coefficients $c_k^i \in \mathbb{C}$ of linear combinations

$$\left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] = \sum_{k=1}^{N} c_k^i \left[\mathbf{x}^{\alpha(k)}\right] \tag{4.114}$$

for all $\left[\mathbf{x}^{\alpha(i)}\right] \in B$.

If $\left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] \in B$, i.e. $\left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] = \left[\mathbf{x}^{\alpha(j)}\right]$ for some $j \in \{1, \ldots, N\}$, then all coefficients c_k^i in (4.114) are zero except for c_j^i, which is equal to one. This means that the i^{th} column of the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ (4.44) consists of zeros and the only nonzero element is in the j^{th} row and it is equal to one.

If $\left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] \notin B$, then obtaining coefficients c_k^i from (4.114) is more complicated. From (4.114) and from the definition of cosets (4.13) we have

$$q_i = \mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)} - \sum_{k=1}^{N} c_k^i \mathbf{x}^{\alpha(k)} \in I. \tag{4.115}$$

This means that having polynomial $q_i \in I$, i.e. a polynomial with leading monomial $\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}$ and the remaining monomials corresponding to monomial cosets from B, then the i^{th} column of the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ can be created from its coefficients. To be more precise, in this case the i^{th} column of the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ is equal to the vector $\left(c_1^i, c_2^i, \ldots, c_N^i\right)^\top$, where $-c_k^i$ is the coefficient of the polynomial $q_i \in I$ (4.115) corresponding to the monomial $\mathbf{x}^{\alpha(k)}$ such that $\left[\mathbf{x}^{\alpha(k)}\right] \in B$.

This way of obtaining multiplication matrices for $\mathbf{x}^{\beta} = x_k$ was presented already in [109]. In this case, i.e. when a multiplication matrix is created for a variable x_k, polynomials q_i (4.115) are polynomials from the reduced Gröbner basis, or from the border basis introduced in [79].

It can be easily proved that using the standard Gröbner basis eigenvalue method presented in Section 4.3.3, i.e. using the remainders of $\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}$ on division by a Gröbner basis as representatives of cosets, we obtain the same result, i.e. the column of the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ consisting of coefficients c_k^i. It follows from the following equality

$$\begin{aligned} T_{\mathbf{x}^\beta}\left(\left[\mathbf{x}^{\alpha(i)}\right]\right) = \left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] = \left[\overline{\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}}^{G}\right] = \left[\overline{q_i + \sum_{k=1}^{N} c_k^i \mathbf{x}^{\alpha(k)}}^{G}\right] = \\ = \left[\overline{\overline{q_i}^{G} + \sum_{k=1}^{N} c_k^i \mathbf{x}^{\alpha(k)}}^{G}\right] = \left[\overline{0 + \sum_{k=1}^{N} c_k^i \mathbf{x}^{\alpha(k)}}^{G}\right] = \sum_{k=1}^{N} c_k^i \left[\mathbf{x}^{\alpha(k)}\right]. \end{aligned} \tag{4.116}$$

Note that $\overline{q_i}^{G} = 0$ in this equality holds because $q_i \in I$.

This means that the action matrix $\mathtt{M}_{\mathbf{x}^\beta}$ can be constructed without explicitly constructing a Gröbner basis and performing polynomial division. All we need is to construct polynomials $q_i \in I$ of the form (4.115) for each $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ for which $\left[\mathbf{x}^{\beta}\mathbf{x}^{\alpha(i)}\right] \notin B$.

Example 23. (Construction of the multiplication matrix) *To show where this method for constructing multiplication matrices needs to generate less polynomials than necessary for constructing a complete Gröbner basis, let's first consider the following very simple system of two*

equations in two unknowns

$$a_1xy + a_2x + a_3y + a_4 = 0, \tag{4.117}$$

$$b_1x + b_2y + b_3 = 0, \tag{4.118}$$

where $a_1, \ldots, a_4, b_1, b_2, b_3 \in \mathbb{C}$ *are some "non-degenerate" coefficients.*

This system can be easily solved using standard methods, e.g. substitution method, and yields two solutions. However, it is instructive to present the proposed method for constructing multiplication matrices.

The Gröbner basis with respect to the lexicographic as well as the graded reverse lexicographic ordering has the form

$$G = \left\{c_1x + c_2y + c_3, d_1y^2 + d_2y + d_3\right\}, \tag{4.119}$$

for some coefficients $c_1, c_2, c_3, d_1, d_2, d_3 \in \mathbb{C}$*. In this case the monomial basis* B *of the quotient ring* $A = \mathbb{C}[x, y]/I$ *is* $B = \{[y], [1]\}$.

Assume that we want to construct the multiplication matrix $\mathtt{M}_x$ *for multiplication by* $[x]$ *in* $A = \mathbb{C}[x, y]/I$*. Then the presented method for constructing multiplication matrices calls for generating polynomials* $q_i = x\mathbf{x}^{\alpha(i)} - \sum_{k=1}^{2} c_k^i \mathbf{x}^{\alpha(k)} \in I$ *for each* $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ *for which* $\left[x\mathbf{x}^{\alpha(i)}\right] \notin B$*. In this case the polynomials* q_i *have the form*

$$q_1 = xy - c_1^1y - c_2^1, \tag{4.120}$$

$$q_2 = x - c_1^2y - c_2^2, \tag{4.121}$$

where $c_k^i \in \mathbb{C}$ *are some coefficients.*

Polynomials (4.120) and (4.121) can be obtained from the two input polynomials $f_1 = a_1xy + a_2x + a_3y + a_4$ *and* $f_2 = b_1x + b_2y + b_3$ *as*

$$q_1 = \frac{1}{a_1}f_1 - \frac{a_2}{a_1b_1}f_2 = xy + \left(\frac{a_3}{a_1} - \frac{a_2b_2}{a_1b_1}\right)y + \left(\frac{a_4}{a_1} - \frac{a_2b_3}{a_1b_1}\right), \tag{4.122}$$

$$q_2 = \frac{1}{b_1}f_2 = x + \frac{b_2}{b_1}y + \frac{b_3}{b_1}. \tag{4.123}$$

This means that $c_1^1 = -\left(\frac{a_3}{a_1} - \frac{a_2b_2}{a_1b_1}\right), c_2^1 = -\left(\frac{a_4}{a_1} - \frac{a_2b_3}{a_1b_1}\right), c_1^2 = -\frac{b_2}{b_1}$ *and* $c_2^2 = -\frac{b_3}{b_1}$ *and the multiplication matrix* $\mathtt{M}_x$ *has the form*

$$\mathtt{M}_x = \begin{pmatrix} c_1^1 & c_1^2 \\ c_2^1 & c_2^2 \end{pmatrix}. \tag{4.124}$$

As it was shown in Section 4.3.3, the solutions of the input polynomial equations can be obtained from eigenvalues and eigenvectors of this matrix. In this case the eigenvalues give us solutions for x*. Solutions for* y *can be obtained by dividing the coordinate of the eigenvector that corresponds to* $[y]$ *by the coordinate that corresponds to* $[1]$.

This example shows that creating the multiplication matrix using the method presented in this section is sometimes easier than creating the multiplication matrix using the standard method presented in Section 4.3.3 that needs to compute a complete Gröbner basis.

In this case the Gröbner basis (4.119) contains polynomial $d_1y^2 + d_2y + d_3$, which can be obtained from the initial polynomials (4.117) and (4.118) by multiplying polynomial f_2 by y and performing G-J elimination on the matrix $\mathtt{H} = \Psi_{T_\succ(H)}(H)$ (4.24), where $H = \{f_1, f_2, yf_2\}$ and $\succ$ is the graded reverse lexicographic ordering. On the other hand generating the multiplication matrix $\mathtt{M}_x$ using the presented method needs to perform G-J elimination on the matrix $\mathtt{F} = \Psi_{T_\succ(F)}(F)$ (4.24) with just the initial two polynomials $F = \{f_1, f_2\}$.

Example 24. (Form of polynomials q_i) *Consider again the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola, i.e. the problem of finding solutions of all "non-degenerate" instances of the system of polynomial equations (4.102).*

In Example 22 we have shown that for each "non-degenerate" instance of this problem the standard monomial basis B (4.20) of the quotient ring $A = \mathbb{C}[x, y]$ has the form

$$B = \{[xy], [x], [y], [1]\}. \tag{4.125}$$

Assume that we want to construct the multiplication matrix $\mathtt{M}_x$ for multiplication by $[x]$ in $A = \mathbb{C}[x, y]/I$. Then the presented method for constructing multiplication matrices calls for generating polynomials $q_i = x\mathbf{x}^{\alpha(i)} - \sum_{k=1}^{4} c_k^i \mathbf{x}^{\alpha(k)} \in I$ for each $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ for which $\left[x\mathbf{x}^{\alpha(i)}\right] \notin B$. In this case this results in two polynomials

$$q_1 = x^2y - c_1^1xy - c_2^1x - c_3^1y - c_4^1, \tag{4.126}$$

$$q_2 = x^2 - c_1^2xy - c_2^2x - c_3^2y - c_4^2, \tag{4.127}$$

where $c_k^i \in \mathbb{C}$ are some coefficients. Then the multiplication matrix $\mathtt{M}_x$ has the form

$$\mathtt{M}_x = \begin{pmatrix} c_1^1 & c_1^2 & 1 & 0 \\ c_2^1 & c_2^2 & 0 & 1 \\ c_3^1 & c_3^2 & 0 & 0 \\ c_4^1 & c_4^2 & 0 & 0 \end{pmatrix}. \tag{4.128}$$

Now the only issue is to find the coefficients of the polynomials q_1 (4.126) and q_2 (4.127).

Since polynomials q_i (4.115) are from the ideal I, we can generate them as polynomial combinations of the initial generators $F = \{f_1, \ldots, f_m\}$ of the ideal I. This means that to create the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$, we need to find the polynomial combinations of the initial polynomials F resulting in polynomials q_i (4.115), or, in other words, we need to find an Elimination trace leading to polynomials q_i (4.115).

One way how to find such an Elimination trace is to use existing strategies [20, 36, 53, 55, 57, 108] for selecting and reducing S-polynomials (4.10) that are used for example in Buchberger based algorithms [39] or the F4 [49] and the F5 [37] algorithms for computing Gröbner bases. In our method we have decided not to use these strategies, but to generate polynomials from the

ideal systematically by multiplying already generated polynomials by all variables or monomials up to some degree and then reducing these polynomials using G-J elimination.

In this way we also generate some "unnecessary" polynomials that wouldn't be usually generated using the strategies [20, 36, 53, 55, 57, 108, 49, 37] and that are not necessary for obtaining polynomials q_i (4.115). However, we can use these polynomials and a larger Elimination trace to efficiently optimize this trace w.r.t. to different criteria, e.g. the size or the numerical stability, by removing different subsets of "unnecessary" polynomials from it. In [81] it was shown that having a larger solver and a larger Elimination trace at the beginning usually leads to a numerically more stable solver after an optimization performed by removing different subsets of "unnecessary" polynomials.

Generation of polynomials from the ideal can be realized using several strategies, for example:

Strategy of multiple eliminations: One possible way is to perform multiple G-J eliminations, e.g., start with the initial polynomials F and then systematically generate new polynomials from I by multiplying already generated polynomials by all individual variables and reducing them each time by G-J elimination. This strategy was used in [89].

Strategy of one elimination: Another possible way is to generate all new polynomials up to a necessary degree in one step by multiplying polynomials from F by selected monomials and then reduce all generated polynomials at once using one G-J elimination. This strategy for generating polynomials from the ideal was used in [33].

Both these strategies can be used to generate all necessary polynomials q_i (4.115), and therefore, to create an Elimination trace T we are looking for. This Elimination trace says which polynomials from the ideal, i.e. which monomial multiples of initial polynomials F, should be added to F and the way how these new polynomials should be eliminated to obtain all polynomials q_i needed for constructing a multiplication matrix. The Elimination trace T is not necessarily correct according to Definition 19, i.e. it doesn't necessarily result in a complete Gröbner basis; however, it is sufficient for finding solutions of F. We will call such an Elimination trace an *admissible Elimination trace*.

Definition 21. (Admissible Elimination trace) *We say that the Elimination trace T is* admissible *for $F = \{f_1, \ldots, f_r\}$ if the Elimination trace reconstruction algorithm for F, T and some monomial ordering $\succ$ doesn't report a failure, i.e. computations based on the Elimination trace T can be carried out for F and $\succ$, and the resulting system contains all polynomials q_i necessary for constructing a multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ of multiplying by some $\left[\mathbf{x}^\beta\right]$ in $\mathbb{C}\left[x_1, \ldots, x_n\right]/I$.*

In our case we consider a set S of instances of a particular problem, which are all in the form of one "representing system" of polynomial equations F, see Definition 20. We therefore know that an Elimination trace that is correct for F w.r.t. some monomial ordering $\succ$ is correct for all instances from S. In other words, all instances from S have the same correct Elimination trace for a fixed monomial ordering $\succ$ and a fixed strategy for generating polynomials from the ideal.

This holds true also for Elimination traces leading to polynomials q_i, i.e. for admissible Elimination traces. An Elimination trace that is admissible for F w.r.t. a monomial ordering $\succ$ and the multiplication by $\left[\mathbf{x}^\beta\right]$ in $A = \mathbb{C}\left[x_1, \ldots, x_n\right]/I$ is admissible for all instances from S.

This means that for systems in the form of one "representing system" F we can find this admissible Elimination trace leading to polynomials q_i (4.115) in the preprocessing phase. Moreover, like in the case of the basis B of $A = \mathbb{C}[x_1, \ldots, x_n]/I$, we can do this in some finite prime field $\mathbb{Z}_p$ and for some "non-degenerate" coefficients (usually random non-zero) of the initial polynomial equations F. Then we can use the same Elimination trace for all systems in the form of F, i.e. all instances from S, also with coefficients from $\mathbb{C}$.

Here we present two methods of finding an admissible Elimination trace: one is based on the strategy of multiple G-J eliminations and one on the strategy of single G-J elimination. Note that it is important to distinguish between the process of finding an Elimination trace and the resulting Elimination trace.

Finding an admissible Elimination trace based on multiple eliminations

Let's start with the strategy of multiple G-J eliminations. In this strategy we generate in each step only polynomials up to some degree d and eliminate them each time by G-J elimination. We increase d when it is not possible to generate more polynomials of degree d by multiplying already generated polynomials by some monomials, and all polynomials q_i (4.115) have not yet been generated. This is important especially when the number of variables is high because in such a case increasing of the degree of generated polynomials results each time in generating large number of new polynomials in many monomials.

To be precise, let F be the input set of polynomials. This could be the set of initial polynomials, i.e. $F = \{f_1, \ldots, f_m\}$, or the set of polynomials that we obtain after G-J elimination of $\mathtt{F} = \Psi_{T_\succ(F)}(F)$ (4.24). The second choice is usually better and leads to simpler solvers, i.e. simpler Elimination traces. Let d be the total degree of the lowest degree polynomial from F. If we do not already have all necessary polynomials q_i we can find a path to polynomials q_i, i.e. an admissible Elimination trace, as described in Algorithm 8.

Function $\mathtt{Included}(Q, H_i)$ used in Algorithm 8 returns true if for every polynomial $q_j \in Q$ there exists a polynomial $h_k \in H_i$ such that $LM(q_j) = LM(h_k)$ and all monomials that have non-zero coefficient in h_k have non-zero coefficient in q_j.

Using Algorithm 8 we can find an admissible Elimination trace resulting in all necessary polynomials q_i. It is because each polynomial from the ideal can be generated as a polynomial combination of initial polynomials F. The proposed algorithm systematically generates all monomial multiples of initial polynomials F. The necessary polynomials q_i can be obtained as linear combinations of these monomial multiples of initial polynomials. In Algorithm 8, we use G-J elimination to find the coefficients of these linear combinations and in this way we generate polynomials q_i.

Finding an admissible Elimination trace based on a single elimination

Another strategy for generating polynomials q_i (4.115) is to generate all polynomials from $I = \langle F \rangle$ up to some necessary degree d in one step by multiplying each polynomial $f_i \in F$ of degree d_i by all monomials up to degree $d - d_i$ and then reduce all generated polynomials at once using one G-J elimination.

Let F be the input system of polynomial equations. Again in this method it is better to start

Algorithm 8 Multiple eliminations trace

Input: $F = \{f_1, \ldots, f_m\}$, a monomial ordering $\succ$, a set Q of all polynomials q_j (4.115) with non-zero coefficients c_k^j necessary for constructing a multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$, the degree d
Output: An admissible Elimination trace T for $F = \{f_1, \ldots, f_m\}$ and $\left[\mathbf{x}^\beta\right]$

1: In the Elimination trace T set $n_1 := m$
2: Set $H_1 := F$ and $i := 1$
3: In the Elimination trace T set $\mathbf{x}^{\alpha(1,j)} := LM\,(h_j)$, $h_j \in H_1$, $j = 1, \ldots, n_1$
4: **while** $\neg\mathtt{Included}(Q, H_i)$ **do**
5: $\quad d := d + 1$
6: $\quad$ **for** $j = 1, \ldots, n_i$ **do**
7: $\quad\quad$ Set $M_j := \{\mathbf{x}^\gamma \mid \deg(\mathbf{x}^\gamma h_j) \leq d,\ h_j \in H_i\}$
8: $\quad\quad$ In the Elimination trace T set $\mathbf{x}^{\beta(k,i,j)} = \mathbf{x}^{\gamma(k)}$ for all $\mathbf{x}^{\gamma(k)} \in M_j \setminus \{1\}$
9: $\quad$ **end for**
10: $\quad$ Set $\overline{H}_i := \{\mathbf{x}^\gamma h_j | h_j \in H_i, \mathbf{x}^\gamma \in M_j\}$, i.e. $\overline{H}_i$ contains all monomial multiples $\mathbf{x}^\gamma h_j$ of all polynomials $h_j \in H_i$ such that the total degree $\deg(\mathbf{x}^\gamma h_j) \leq d$
11: $\quad$ Perform G-J elimination on the matrix $\overline{\mathtt{H}}_i = \Psi_{T_\succ(\overline{H}_i)}(\overline{H}_i)$ (4.24). Store the result in $\widetilde{\overline{\mathtt{H}}}_i$.
12: $\quad$ Let H_{i+1} be the set of non-zero polynomials with matrix representation $\mathtt{H}_{i+1} = \widetilde{\overline{\mathtt{H}}}_i$, ordered according to their leading monomials
13: $\quad$ In the Elimination trace T set $n_{i+1} := |H_{i+1}|$
14: $\quad$ In the Elimination trace T set $\mathbf{x}^{\alpha(i+1,j)} := LM\,(h_j)$, $h_j \in H_{i+1}$, $j = 1, \ldots, n_{i+1}$
15: $\quad$ **if** a new polynomial with degree $< d$ was generated by the G-J elimination, i.e. there exists $\mathbf{x}^{\alpha(i+1,k)}$, $k \in \{1, \ldots, n_{i+1}\}$, such that $\deg\left(\mathbf{x}^{\alpha(i+1,k)}\right) < d$ and $\mathbf{x}^{\alpha(i+1,k)} \notin \left\{\mathbf{x}^{\alpha(i,1)}, \ldots, \mathbf{x}^{\alpha(i,n_i)}\right\}$, **then**
16: $\quad\quad d := d - 1$
17: $\quad$ **end if**
18: $\quad i := i + 1$
19: **end while**

with eliminated initial polynomials F and perform two G-J eliminations, one at the beginning and one at the end. Let d be the total degree of the lowest degree polynomial from F.

The process of finding an admissible Elimination trace based on a single elimination strategy is described in Algorithm 9.

In this case the final Elimination trace represents the process of generating all monomial multiples $\mathbf{x}^\gamma h_j$ of initial polynomials $h_j \in H_1$ up to the degree d found in Algorithm 9 and reducing the extended set of polynomial $\overline{\mathtt{H}}_1$ by G-J elimination to obtain polynomials H_2 containing all necessary polynomials q_i.

This means that if all necessary polynomials q_i are not already included in the input polynomials F, then the Elimination trace will contain the number n_1 of non-zero polynomials in $H_1 := F$, the number n_2 of non-zero polynomials in H_2 where $\mathtt{H}_2 = \widetilde{\overline{\mathtt{H}}}_1$, the sequence of leading monomials of polynomials in H_1 and H_2, and finally the sequence of monomials which are used to obtain the extended set of polynomials $\overline{\mathtt{H}}_1$ from H_1.

Algorithm 9 Single elimination trace

Input: $F = \{f_1, \ldots, f_m\}$, a monomial ordering $\succ$, a set Q of all polynomials q_j (4.115) with non-zero coefficients c_k^j necessary for constructing a multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$, the degree d
Output: An admissible Elimination trace T for $F = \{f_1, \ldots, f_m\}$ and $\left[\mathbf{x}^\beta\right]$

1: In the Elimination trace T set $n_1 := m$
2: Set $H_1 := F$
3: In the Elimination trace T set $\mathbf{x}^{\alpha(1,j)} := LM\,(h_j)$, $h_j \in H_1$, $j = 1, \ldots, n_1$
4: Set $H_2 := H_1$
5: **while** $\neg\mathtt{Included}(Q, H_2)$ **do**
6: Set $d := d + 1$
7: **for** $j = 1, \ldots, n_1$ **do**
8: Set $M_j := \{\mathbf{x}^\gamma \mid \deg(\mathbf{x}^\gamma h_j) \leq d,\, h_j \in H_1\}$
9: **end for**
10: Set $\overline{H}_1 := \{\mathbf{x}^\gamma h_j | h_j \in H_1, \mathbf{x}^\gamma \in M_j\}$, i.e. $\overline{H}_1$ contains all monomial multiples $\mathbf{x}^\gamma h_j$ of all polynomials $h_j \in H_1$ such that the total degree $\deg(\mathbf{x}^\gamma h_j) \leq d$
11: Perform G-J elimination on the matrix $\overline{\mathtt{H}}_1 = \Psi_{T_\succ(\overline{H}_1)}(\overline{H}_1)$ (4.24). Store the result in $\widetilde{\overline{\mathtt{H}}}_1$.
12: Let H_2 be the set of non-zero polynomials with matrix representation $\mathtt{H}_2 = \widetilde{\overline{\mathtt{H}}}_1$, ordered according to their leading monomials
13: **end while**
14: **if** $H_2 \neq H_1$ **then**
15: In the Elimination trace T set $\mathbf{x}^{\beta(k,1,j)} := \mathbf{x}^{\gamma(k)}$ for all $\mathbf{x}^{\gamma(k)} \in M_j \setminus \{1\}$ and all $j = 1, \ldots, n_1$
16: In the Elimination trace T set $n_2 := |H_2|$
17: In the Elimination trace T set $\mathbf{x}^{\alpha(2,j)} := LM\,(h_j)$, $h_j \in H_2$, $j = 1, \ldots, n_2$
18: **end if**

We have observed that the first method with multiple G-J eliminations is better for systems with a large number of variables because by using this multiple elimination method the number of generated monomials grows slower than in the case of a single G-J elimination. On the other hand, the method with a single G-J elimination is better for systems with fewer variables since in this case using one G-J elimination is numerically slightly more stable than using multiple G-J eliminations.

Removing unnecessary polynomials

Since both presented methods generate polynomials from the ideal systematically, they also generate many "unnecessary" polynomials, i.e. polynomials that are not necessary for creating polynomials q_i (4.115). Therefore, it is good to reduce the number of generated polynomials by removing these "unnecessary" ones.

There are many different ways how such polynomials can be removed. It is because removing one "unnecessary" polynomial may affect removing other "unnecessary" polynomials, and therefore, the result depends on the ordering in which these polynomials are removed. Moreover,

these "unnecessary" polynomials may be removed w.r.t. different criteria, like the size of the final Elimination trace or the numerical stability of the final solver created from the Elimination trace.

Here we present a method for removing "unnecessary" polynomials that tries to create as small admissible Elimination trace as possible and that works for the second presented strategy, i.e. the single G-J elimination strategy. In this case, we have a set of monomials that should be used for multiplying the initial polynomials H_1 to generate new polynomials $\overline{H}_1$. Since these monomials were generated systematically, this set of monomials contains all monomials up to the degree $d - d_i$ for each initial polynomial $h_i \in H_1$ of degree d_i.

We know that after G-J elimination of the coefficient $\overline{\mathtt{H}}_1 = \Psi_{T_\succ(\overline{H}_1)}(\overline{H}_1)$ we obtain all polynomials q_i (4.115) necessary for constructing the multiplication matrix.

Let the polynomials in $\overline{H}_1$ be partially ordered according to their multidegree, starting from the polynomial with the smallest leading monomial to the polynomial with the largest leading monomial. We can now systematically reduce the number of generated polynomials in the following way:

Algorithm 10 Removing polynomials for a single admissible Elimination trace

1. For all polynomials h_j from $\overline{H}_1$ starting with the last polynomial (with the polynomial with the largest LM w.r.t. $\succ$) do
   a) Perform G-J elimination on the matrix $\mathtt{M} = \Psi_{T_\succ(\overline{H}_1)}\left(\overline{H}_1 \setminus \{h_j\}\right)$ in the matrix representation (4.24) of the polynomials from $\overline{H}_1 \setminus h_j$, i.e. of the polynomials from $\overline{H}_1$ without the polynomial h_j.
   b) If the eliminated set of polynomials contains all necessary polynomials q_i, then $\overline{H}_1 := \overline{H}_1 \setminus h_j$.
   c) Go to step 1.

A method for removing "unnecessary" polynomials for multiple eliminations strategy can be constructed in a similar way.

The final admissible Elimination trace consists only of those polynomials that remain after performing this removing procedure.

Removing unnecessary monomials

After removing "unnecessary polynomials", we can remove also "unnecessary monomials" from the polynomials contained in the last step of the Elimination trace. By "unnecessary monomials" we mean monomials that do not affect the resulting multiplication matrix, and therefore, also the result of the final solver.

In this case the "unnecessary monomial" is each monomial $\mathbf{x}^\alpha$ such that:

1. $\mathbf{x}^\alpha$ is not the leading monomial of any polynomial in the Elimination trace after the final G-J elimination, i.e. $\mathbf{x}^\alpha \notin \left\{\mathbf{x}^{\alpha(l,1)}, \ldots, \mathbf{x}^{\alpha(l,n_l)}\right\}$,

2. $\mathbf{x}^{\alpha}$ doesn't belong to monomials from the monomial basis B of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$, i.e. $\mathbf{x}^{\alpha} \neq \mathbf{x}^{\alpha(i)}$, $\left[\mathbf{x}^{\alpha(i)}\right] \in B$.

Let $\overline{H}_{l-1}$ be the last extended set of polynomials in the final admissible Elimination trace that was obtained using one of the previously described methods and after removing unnecessary polynomials. This means that $\overline{H}_{l-1}$ is the set of polynomials that after G-J elimination results in the final set of polynomials H_l containing all necessary polynomials q_i (4.115).

Let $\overline{\mathtt{H}}_{l-1} = \Psi_{T_{\succ}(\overline{H}_{l-1})}(\overline{H}_{l-1})$ be the matrix representation (4.24) of $\overline{H}_{l-1}$. If we remove columns of $\overline{\mathtt{H}}_{l-1}$ that correspond to a set of all "unnecessary monomials" U, then after G-J elimination of the new matrix $\mathtt{M} = \Psi_{T_{\succ}\left(\overline{H}_{l-1} \setminus U\right)}\left(\overline{H}_{l-1}\right)$, the polynomials q_i contained in the eliminated matrix $\widetilde{\mathtt{M}}$ will remain unchanged. This means that we can create the multiplication matrix from its coefficients.

In fact, if n_l is the number of polynomials in the last set of polynomials H_l and N is the number of solutions of our system, then the matrix $\mathtt{M}$ that needs to be eliminated in the final step of the Elimination trace has the dimension $n_l \times (n_l + N)$.

4.4.4 Online phase

In the "online phase", the specific solver, created using the admissible Elimination trace found in the "offline phase", is used to efficiently solve concrete instances of the particular problem. This efficient specific solver is not general and solves only systems of polynomial equations in the form of the "representing system" F; however, it is faster than a general solver. Moreover, for problems for which all "interesting instances" are from one set S of "consistent" instances represented by F, this specific solver can be used to "solve" the "whole" problem. We will call this solver the online solver.

The online solver consists of three parts:

1. The solver starts with some input polynomial equations $f_1 = f_2 = \cdots = f_m = 0$ belonging to the considered set S with concrete coefficients from $\mathbb{C}$. Then, according to the found Elimination trace T, the solver multiplies these input polynomial equations with found monomials, eliminates them with one or multiple G-J eliminations as defined in T and, in this way, generates all necessary polynomials q_i (4.115). This process is similar to the Elimination trace reconstruction algorithm, see Algorithm 7. However, in this case, this part of the online solver works in the final step of the Elimination trace with the matrix representation of polynomials without all "unnecessary monomials". Moreover, if it is clear that the input to the online solver is a system belonging to the set S for which the admissible Elimination trace was created it is not necessary to test whether the generated polynomials have given leading monomials.

   The final algorithm for generating all necessary polynomials q_i (4.115) from the input polynomials F, given an admissible Elimination trace T, is described in Algorithm 11.

2. In the next step of the online solver the multiplication matrix $\mathtt{M}_{\mathbf{x}^{\beta}}$ is created from the coefficients of the polynomials q_i.

3. Finally, the solutions of the input system of polynomial equations F are extracted from the eigenvalues and eigenvectors of this multiplication matrix.

This means that the final online solver only performs one or more G-J eliminations, and the eigenvalue computations to solve the input polynomial equations. Usually such solver is significantly faster than a general Gröbner basis solver for solving systems of polynomial equations.

Algorithm 11 Creation of polynomials q_i from an admissible Elimination trace

Input: $F = \{f_1, \ldots, f_m\}$, a monomial ordering $\succ$, an Elimination trace T that is admissible for F and multiplication by $\left[\mathbf{x}^\beta\right]$, a set of "unnecessary monomials" U

Output: All polynomials q_i (4.115) necessary for constructing the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$.

1: Set $H_1 := F$
2: **for** $i = 1, \ldots, l-2$ **do**
3: $\quad \overline{H}_i := H_i \cup \left\{\mathbf{x}^{\beta(1,i,j)} h_j, \ldots, \mathbf{x}^{\beta(m_{i,j},i,j)} h_j \mid h_j \in H_i, j = 1, \ldots, n_i\right\}$
4: $\quad$ Perform G-J elimination on the matrix $\overline{\mathtt{H}}_i = \Psi_{T_\succ(\overline{H}_i)}(\overline{H}_i)$ (4.24). Store the result in $\widetilde{\overline{\mathtt{H}}}_i$.
5: $\quad$ Let H_{i+1} be the set of non-zero polynomials with matrix representation $\mathtt{H}_{i+1} = \widetilde{\overline{\mathtt{H}}}_i$, ordered according to their leading monomials
6: **end for**
7: $\overline{H}_{l-1} := H_{l-1} \cup \left\{\mathbf{x}^{\beta(1,l-1,j)} h_j, \ldots, \mathbf{x}^{\beta\left(m_{l-1,j},l-1,j\right)} h_j \mid h_j \in H_{l-1}, j = 1, \ldots, n_{l-1}\right\}$
8: Perform G-J elimination on the matrix $\mathtt{M} = \Psi_{T_\succ(\overline{H}_{l-1} \setminus U)}(\overline{H}_{l-1})$ in the matrix representation (4.24) of the polynomials from $\overline{H}_{l-1}$. Here $T_\succ(\overline{H}_{l-1} \setminus U)$ is the ordered set of monomials of $\overline{H}_{l-1}$ w.r.t. the monomial ordering $\succ$ without the "unnecessary monomials" U. Store the result in $\widetilde{\mathtt{M}}$.
9: Let H_l be the set of non-zero polynomials with matrix representation $\mathtt{H}_l = \widetilde{\mathtt{M}}$.
10: Extract polynomials q_i (4.115) from H_l

4.4.5 Final specialized Gröbner basis method based on eigenvalue computations

In this section we summarize the presented specialized Gröbner basis method designed for solving systems of polynomial equations that are in the form of a "representing system" F, see Definition 20.

Assume a set S of "consistent" instances of a particular problem that are all in the form of one "representing system" of polynomial equations $F = \{f_1 = f_2 = \cdots = f_m = 0\}$. A solver solving for solving all instances belonging to the set S can be created in the following way:

Offline phase

1. Fix a monomial ordering $\succ$. (The graded reverse lexicographical ordering is often good.)
2. Find the basis B of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$ as the basis that repeatedly appears for several different (usually random) choices of coefficients of input equations

that all belong to the set S. Do computations in a suitably chosen finite prime field to speed them up and to avoid numerical problems.

3. Choose a strategy for generating polynomials from the ideal $I = \langle f_1, \ldots, f_m \rangle$.
4. For a suitably chosen monomial $\mathbf{x}^\beta$ and the selected strategy for generating polynomials from the ideal I, find a "path" from the initial polynomials $f_1, \ldots, f_m$ to polynomials q_i (4.115), i.e. an admissible Elimination trace T. Do this by systematically generating higher order polynomials from the initial polynomials using the selected strategy, e.g. using Algorithm 8 or Algorithm 9. Again, do computations in a finite prime field and with some "non-degenerate" coefficients from S (usually random).
5. Remove "unnecessary" polynomials, i.e. polynomials that are not necessary for generating polynomials q_i, from all generated polynomials.
6. Detect "unnecessary monomials", i.e. monomials that do not affect the polynomials q_i.
7. The remaining polynomials, together with the way of how they should be eliminated, form the resulting admissible Elimination trace for the "online" solver.

The final "online" solver for solving all systems of polynomial equations in the form of the "representing system" F consists of these steps:

Online phase

1. Using the Elimination trace found in the "offline phase", generate all necessary polynomials q_i from the initial polynomials with concrete coefficients from $\mathbb{C}$, see Algorithm 11.
2. Construct the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ from coefficients of polynomials q_i.
3. Find the solutions of the input system of polynomial equations by finding eigenvectors of the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ or by computing roots of its characteristic polynomial [41, 30, 24] of $\mathtt{M}_{\mathbf{x}^\beta}$.

The following example shows how this specialized Gröbner basis method can be used to construct an efficient specific solver for solving all "non-degenerate" instances of the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola and how this specific solver looks like.

Example 25. (Specialized Gröbner basis method) *The problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola results in the system of polynomial equations*

$$\begin{aligned} a_0x^2 + a_1x + a_2y^2 + a_3y + a_4 = 0, \\ b_0x^2 + b_1x - b_2y^2 + b_3y + b_4 = 0, \end{aligned} \tag{4.129}$$

where $a_0, \ldots, a_4, b_0, \ldots, b_4 \in \mathbb{C}$, such that a_0, a_2 are both non-zero and with the same sign and the same holds true also for b_0 and b_2.

To create an efficient specific solver for solving all "non-degenerate" instances of the system (4.129), using the presented specialized Gröbner basis method, we first need to find the

standard monomial basis B of the quotient ring $A = \mathbb{C}[x,y]/I$. To do this, we use a standard Gröbner basis method and do computations in some finite prime field and with random "non-degenerate" coefficients of input equations (4.129). In Example 22 it was shown that the monomial basis B has the form

$$B = \{[xy], [x], [y], [1]\}, \tag{4.130}$$

w.r.t. the graded reverse lexicographic ordering and $x > y$.

Having this basis, we know the form of polynomials q_i (4.115) necessary for constructing the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ for multiplication by $\left[\mathbf{x}^\beta\right]$ in $A = \mathbb{C}[x,y]/I$. In Example 24 we have shown how these polynomials q_i look for $\mathbf{x}^\beta = x$.

To construct the multiplication matrix $\mathtt{M}_x$, we need to generate the following two polynomials

$$q_1 = x^2y - c_1^1xy - c_2^1x - c_3^1y - c_4^1, \tag{4.131}$$

$$q_2 = x^2 - c_1^2xy - c_2^2x - c_3^2y - c_4^2, \tag{4.132}$$

where $c_k^i \in \mathbb{C}$ are some coefficients. Then, the multiplication matrix $\mathtt{M}_x$ has the form

$$\mathtt{M}_x = \begin{pmatrix} c_1^1 & c_1^2 & 1 & 0 \\ c_2^1 & c_2^2 & 0 & 1 \\ c_3^1 & c_3^2 & 0 & 0 \\ c_4^1 & c_4^2 & 0 & 0 \end{pmatrix}. \tag{4.133}$$

To generate the two polynomials (4.131) and (4.132) from the initial polynomials (4.129), we first need to select the strategy of generating polynomials from the ideal I. Since, in this case, both presented strategies, i.e. the multiple eliminations strategy and the single elimination strategy, result in similar (in fact for eliminated initial polynomials the same) solvers, we will present only the results of the second strategy, which is described in Algorithm 9.

Let's start with the eliminated initial polynomials, i.e. the polynomials that we obtain from (4.129) by performing G-J elimination on the coefficient matrix $\mathtt{F} = \Psi_{T_\succ(F)}(F)$ in the matrix representation (4.24) of the two initial polynomials. Here, $T_\succ(F)$ is the ordered set of monomials of (4.129) w.r.t. the graded reverse lexicographic ordering. These eliminated polynomials have the form

$$h_1 = x^2 + c_1x + c_2y + c_3, \tag{4.134}$$

$$h_2 = y^2 + c_4x + c_5y + c_6. \tag{4.135}$$

This means that the initial set of polynomials in Algorithm 9 is $H_1 = \{h_1, h_2\}$, and this algorithm sets $n_1 := 2$ and $\left(\mathbf{x}^{\alpha(1,1)}, \mathbf{x}^{\alpha(1,2)}\right) := \left(x^2, y^2\right)$ in the Elimination trace T.

Polynomial h_1 (4.134) already has the form of the searched polynomial q_2 (4.132). Therefore the only missing polynomial is the polynomial q_1 (4.131).

To generate this polynomial from the ideal, we need to generate new polynomials, monomial multiples of the two eliminated input polynomials, using the selected strategy. Since both input polynomials have degree two, we set $d = 2$ and generate all monomial multiples of the eliminated initial polynomials h_1 (4.134) and h_2 (4.134) up to the total degree three. This means

that the extended set $\overline{H}_1$ contains six polynomials $\overline{H}_1 = \{h_1, h_2, xh_1, yh_1, xh_2, yh_2\}$. We can rewrite these polynomials in the matrix form

$$\begin{array}{l} h_1 \to \\ h_2 \to \\ xh_1 \to \\ yh_1 \to \\ xh_2 \to \\ yh_2 \to \end{array} \begin{pmatrix} & & & \bullet & & \bullet & \bullet & \bullet \\ & & & & \bullet & \bullet & \bullet & \bullet \\ \bullet & & & \bullet & \bullet & & \bullet & \\ & \bullet & & & \bullet & \bullet & & \bullet \\ & & \bullet & & \bullet & \bullet & & \bullet \\ & & & \bullet & & \bullet & \bullet & \end{pmatrix} \begin{pmatrix} x^3 \\ x^2y \\ xy^2 \\ y^3 \\ x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix}, \tag{4.136}$$

where $\bullet$ stands for some nonzero coefficient. The coefficient matrix in (4.136) is in fact the matrix representation $\overline{\mathtt{H}}_1 = \Psi_{T_\succ(\overline{H}_1)}\left(\overline{H}_1\right)$ of polynomials $\overline{H}_1$, where $T_\succ\left(\overline{H}_1\right) = \left(x^3, x^2y, xy^2, y^3, x^2, xy, y^2, x, y, 1\right)$. After the G-J elimination of this coefficient matrix $\overline{\mathtt{H}}_1$ in (4.136) with some random "non-degenerate" coefficients from some finite prime field, we obtain

$$\begin{array}{l} \\ q_1 \to \\ \\ \\ q_2 \to \\ \end{array} \begin{pmatrix} \bullet & & & & \bullet & & \bullet & \bullet & \bullet \\ & \bullet & & & \bullet & & \bullet & \bullet & \bullet \\ & & \bullet & & \bullet & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet & & \bullet & \bullet & \bullet \\ & & & & \bullet & & \bullet & \bullet & \bullet \\ & & & & & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x^3 \\ x^2y \\ xy^2 \\ y^3 \\ x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix}, \tag{4.137}$$

where again $\bullet$ stands for a non-zero coefficient.

As it can be seen eliminated matrix $\widetilde{\overline{\mathtt{H}}}_1$ (4.137) already contains both necessary polynomials q_1 (4.131) and q_2 (4.132). In this case, the polynomial corresponding to the second row of the matrix (4.137) has the form of the polynomial q_1 (4.131), and the polynomial corresponding to the fifth row has the form of the polynomial q_2 (4.132). Therefore, we do not need to generate other polynomials.

This means that Algorithm 9 sets $\left(\mathbf{x}^{\beta(1,1,1)}, \mathbf{x}^{\beta(2,1,1)}\right) := (x, y)$, $\left(\mathbf{x}^{\beta(1,1,2)}, \mathbf{x}^{\beta(2,1,2)}\right) := (x, y)$, $n_2 := 6$ and $\left(\mathbf{x}^{\alpha(2,1)}, \mathbf{x}^{\alpha(2,2)}, \mathbf{x}^{\alpha(2,3)}, \mathbf{x}^{\alpha(2,4)}, \mathbf{x}^{\alpha(2,5)}, \mathbf{x}^{\alpha(2,6)}\right) := \left(x^3, x^2y, xy^2, y^3, x^2, y^2\right)$ in the Elimination trace T, and it stops.

Now we can remove unnecessary polynomials using Algorithm 10. We found that we can remove polynomials xh_1, xh_2 and yh_2 in this case.

Therefore, the final admissible Elimination trace T for eliminated input polynomials (4.134) and (4.135) consists of the numbers of rows of generated matrices $(n_1, n_2) = (2, 3)$; the

leading monomial sequences $\left(\mathbf{x}^{\alpha(1,1)}, \mathbf{x}^{\alpha(1,2)}\right) = \left(x^2, y^2\right)$ *and* $\left(\mathbf{x}^{\alpha(2,1)}, \mathbf{x}^{\alpha(2,2)}, \mathbf{x}^{\alpha(2,3)}\right) = \left(x^2y, x^2, y^2\right)$*; and the multiplier sequence* $\left(\mathbf{x}^{\beta(1,1,1)}\right) = (y)$.

In the next step, we can remove "unnecessary monomials". However, in this case, all monomials contained in the polynomials h_1, h_2, yh_1*, i.e. the monomials* $x^2y, x^2, xy, y^2, x, y, 1$*, are necessary for constructing the polynomials* q_1 *and* q_2.

The final online solver for solving all systems of polynomial equations in the form of the "representing system" (4.129) with "non-degenerate" coefficients consists of the following steps:

1. *Take two input polynomial equations* F *with concrete coefficients from* $\mathbb{C}$ *and perform G-J elimination of the matrix* $\mathtt{F} = \Psi_{T_\succ(F)}(F)$ *(4.24) Denote the two eliminated polynomials as* $H_1 = \{h_1, h_2\}$*, i.e.* $\mathtt{H}_1 = \widetilde{\mathtt{F}}$*, where* h_1 *is the polynomial with the leading monomial* x^2 *and* h_2 *the polynomial with the leading monomial* y^2.

2. *Use the admissible Elimination trace found for* H_1 *to generate polynomials* q_1 *(4.131) and* q_2 *(4.132), i.e. polynomials necessary for constructing the multiplication matrix* $\mathtt{M}_x$.

   a) *Add the polynomial* yh_1 *to the two polynomials* h_1 *and* h_2 *and set* $\overline{H}_1 := \{h_1, h_2, yh_1\}$.

   b) *Perform G-J elimination on the coefficient matrix* $\overline{\mathtt{H}}_1 = \Psi_{T_\succ\left(\overline{H}_1\right)}\left(\overline{H}_1\right)$ *in the matrix representation of these three polynomials from* $\overline{H}_1$.

   c) *After the G-J elimination of the matrix* $\overline{\mathtt{H}}_1$*, extract the coefficients from the* 1^{st} *and the* 2^{nd} *row of the matrix* $\widetilde{\overline{\mathtt{H}}}_1$ *corresponding to the polynomials* q_1 *(4.131) and* q_2 *(4.132).*

3. *Use extracted coefficients to create the multiplication matrix* $\mathtt{M}_x$ *of the form (4.133).*

4. *Compute the four left eigenvectors* $\mathbf{v}_i$ *of the multiplication matrix* $\mathtt{M}_x$ *or, equivalently, the four right eigenvectors of* $\mathtt{M}_x^\top$.

5. *Find four solutions* (x_i, y_i) *of the input system of polynomial equations as* $x_i = \frac{\mathbf{v}_i(2)}{\mathbf{v}_i(4)}$ *and* $y_i = \frac{\mathbf{v}_i(3)}{\mathbf{v}_i(4)}$*, where* $\mathbf{v}_i(j)$ *stands for the* j^{th} *coordinate of the* i^{th} *eigenvector* $\mathbf{v}_i$.

This means that the final online solver consists only of two G-J eliminations and computations of eigenvectors of a 4×4 *matrix.*

For the concrete instance of the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola described in Example 4

$$\begin{aligned} 2x^2 + y^2 + 3y - 12 &= 0, \\ x^2 - y^2 + x + 3y - 4 &= 0, \end{aligned} \tag{4.138}$$

the presented online solver performs these steps:

1. *Perform G-J elimination on the matrix representation (4.24) of the initial two polynomials* $F = \left\{2x^2 + y^2 + 3y - 12, x^2 - y^2 + x + 3y - 4\right\}$*, i.e. on the coefficient matrix* $\mathtt{F} =$

$\Psi_{T(F)}(F)$, *where* $T(F) = \left\{x^2, y^2, x, y, 1\right\}$ *is the ordered set of monomials using the graded reverse monomial ordering. In this case,* $\mathtt{F}$ *has the form*

$$\mathtt{F} = \begin{pmatrix} 2 & 1 & 0 & 3 & -12 \\ 1 & -1 & 1 & 3 & -4 \end{pmatrix}. \tag{4.139}$$

After the G-J elimination of this matrix $\mathtt{F}$*, we obtain*

$$\mathtt{H}_1 = \widetilde{\mathtt{F}} = \begin{pmatrix} 1 & 0 & \frac{1}{3} & 2 & -\frac{16}{3} \\ 0 & 1 & -\frac{2}{3} & -1 & -\frac{4}{3} \end{pmatrix}. \tag{4.140}$$

2. *Let's denote these two eliminated polynomials* $h_1 = x^2 + \frac{1}{3}x + 2y - \frac{16}{3}$ *and* $h_2 = y^2 - \frac{2}{3}x - y - \frac{4}{3}$*. According to the precomputed admissible Elimination trace, in the second step, the online solver performs the following steps:*

   a) *Adds to these two polynomials the polynomial* $yh_1 = x^2y + \frac{1}{3}xy + 2y^2 - \frac{16}{3}y$*.*

   b) *Performs G-J elimination on the matrix representation of these three polynomials* $\overline{H}_1 = \{h_1, h_2, yh_1\}$*, i.e. on the matrix*

$$\overline{\mathtt{H}}_1 = \Psi_{T\left(\overline{H}_1\right)}\left(\overline{H}_1\right) = \begin{pmatrix} 0 & 1 & 0 & 0 & \frac{1}{3} & 2 & -\frac{16}{3} \\ 0 & 0 & 0 & 1 & -\frac{2}{3} & -1 & -\frac{4}{3} \\ 1 & 0 & \frac{1}{3} & 2 & 0 & -\frac{16}{3} & 0 \end{pmatrix}, \tag{4.141}$$

   where $T\left(\overline{H}_1\right) = \left\{x^2y, x^2, xy, y^2, x, y, 1\right\}$*.*

   c) *After the G-J elimination of the matrix* $\overline{\mathtt{H}}_1$ *(4.141), we obtain the matrix*

$$\widetilde{\overline{\mathtt{H}}}_1 = \begin{pmatrix} 1 & 0 & \frac{1}{3} & 0 & \frac{4}{3} & -\frac{10}{3} & \frac{8}{3} \\ 0 & 1 & 0 & 0 & \frac{1}{3} & 2 & -\frac{16}{3} \\ 0 & 0 & 0 & 1 & -\frac{2}{3} & -1 & -\frac{4}{3} \end{pmatrix}. \tag{4.142}$$

   The first row of this matrix corresponds to the searched polynomial $q_1 = x^2y + \frac{1}{3}xy + \frac{4}{3}x - \frac{10}{3}y + \frac{8}{3}$ *and the second row to the searched polynomial* $q_2 = x^2 + \frac{1}{3}x + 2y - \frac{16}{3}$*. This means that in (4.131)* $c_1^1 = -\frac{1}{3}, c_2^1 = -\frac{4}{3}, c_3^1 = \frac{10}{3}, c_4^1 = -\frac{8}{3}$*; and in (4.132)* $c_1^2 = 0, c_2^2 = -\frac{1}{3}, c_3^2 = -2$ *and* $c_4^4 = \frac{16}{3}$*.*

3. *Then the online solver creates the multiplication matrix* $\mathtt{M}_x$*, which in this case has the form*

$$\mathtt{M}_x = \begin{pmatrix} -\frac{1}{3} & 0 & 1 & 0 \\ -\frac{4}{3} & -\frac{1}{3} & 0 & 1 \\ \frac{10}{3} & -2 & 0 & 0 \\ -\frac{8}{3} & \frac{16}{3} & 0 & 0 \end{pmatrix}. \tag{4.143}$$

4. *In the next step, the online solver computes the left eigenvectors of* $\mathtt{M}_x$ *(4.143) or right eigenvectors of* $\mathtt{M}_x^\top$*. In this case, these eigenvectors are*

$$\begin{pmatrix} -2 \\ -2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -\frac{7}{9} \\ -\frac{7}{3} \\ \frac{1}{3} \\ 1 \end{pmatrix} \begin{pmatrix} -\frac{32}{9} \\ \frac{8}{3} \\ -\frac{4}{3} \\ 1 \end{pmatrix}. \tag{4.144}$$

5. *Finally, the online solver finds solutions of the input system (4.138) using the coordinates of the eigenvectors (4.144) corresponding to* $[x]$ *and* $[y]$*. The second and the third coordinates give us the well known solutions*

$$(-2,1)\,,(1,2)\,,\left(-\frac{7}{3},\frac{1}{3}\right),\left(\frac{8}{3},-\frac{4}{3}\right). \tag{4.145}$$

4.4.6 Modifications of specialized Gröbner basis method

The presented specialized Gröbner basis method for solving systems of polynomial equations in the form of some system F, is based on the modified eigenvalue method presented in Section 4.3.3. However, this specialized Gröbner basis method can also be easily modified to the remaining two standard Gröbner basis methods for solving systems of polynomial equations presented in Sections 4.3.1 and 4.3.2, i.e. the Elimination lexicographic Gröbner basis method and the FGLM conversion method.

These modifications need to be performed in the process of finding an Elimination trace. In this case, this process will not end when all necessary polynomials q_i (4.115) for constructing a multiplication matrix are generated, but it will end when a complete Gröbner basis is generated. This means that Algorithm 8 and Algorithm 9 for finding an admissible Elimination trace presented in the previous section will stop when all polynomials from a Gröbner basis w.r.t. some monomial ordering are generated. In this way an Elimination trace that is correct for F will be found. Again this correct Elimination trace can be found only once, in the offline phase, and for some "non-degenerate" coefficients of F from $\mathbb{Z}_p$. Then it can be used to solve all "consistent" instances of the particular problem represented by F.

Using the Elimination Gröbner basis method, or the FGLM conversion method, may be useful in the case when the construction of a lexicographic Gröbner basis for a particular problem is not extremely complicated, or we know some constraints on some of the variables, e.g., if we know that a variable belongs to some interval, and we need to find solutions only for that variable.

In such a case it may be sometimes more efficient to generate more polynomials resulting in a complete grevlex Gröbner basis and then perform the polynomial division in the FGLM [48] algorithm, or even to generate a lexicographic Gröbner basis, than doing the eigenvalue computations as in the presented method. It is because the lexicographic Gröbner basis will provide a single-variable polynomial in the selected variable, and such a polynomial can be efficiently solved using Sturm sequences [66] w.r.t. the known constraints.

Moreover, in [30, 24] we have shown that the time-consuming polynomial division performed in the standard FGLM algorithm [48] can be replaced with efficient matrix-vector multiplication using a constructed multiplication matrix (4.44). This method is very efficient and also numerically stable for smaller problems with up to 20 solutions, and it significantly speeds up final online solvers. This "matrix FGLM method" even doesn't require a compete grevlex Gröbner basis, i.e. a correct Elimination trace. Therefore, this method can be easily incorporated in the presented specialized Gröbner basis method.

Another class of modifications that can be performed on the proposed specialized Gröbner basis method is an improvement of the numerical stability of the final online solver. Methods for improving numerical stability of Gröbner basis solvers have been studied in [33, 34, 31]. These methods are based on LU or QR decompositions of matrices in the matrix representation (4.24) of polynomials; changing the ordering of monomials, i.e. reordering columns in these matrices; or "extending" a basis of the quotient ring $\mathbb{C}[x_1, \ldots, x_n]/I$. Such improvements can be incorporated into the proposed specialized Gröbner basis method. However, for almost all problems presented and solved in this thesis, the methods for improving the numerical stability [33, 34, 31] were not necessary or resulted only in a small improvement w.r.t a larger computation effort, so we have decided not to use them.

5 Resultant based methods for solving systems of polynomial equations

"Pure mathematics is, in its way, the poetry of logical ideas."

Albert Einstein

An alternative algebraic approach for solving systems of polynomial equations is a method based on multipolynomial resultants. The resultant of two polynomials in one variable described in Section 3.1.2 is well known and is implemented in many computer algebra systems.

In this chapter we first review its generalization to several polynomials in several variables known as multipolynomial resultant and show one method of its computation. Then we review two classical methods for solving systems of polynomial equations based on multipolynomial resultants. Finally, we present a specialized resultant based method based on hidden variable resultants suitable for solving problems arising in computer vision.

5.1 Basic Concepts

Multipolynomial resultants are defined for $n+1$ polynomials $f_0, \ldots, f_n$ in n variables $x_1, \ldots, x_n$ or equivalently $n+1$ homogeneous polynomials $F_0, \ldots, F_n$ in $n+1$ variables $x_0, \ldots, x_n$.

One can note that we have one more equation than variables. It is because resultants were initially developed to determine whether a system of polynomial equations has a common root or not. However, resultants can be also used to solve a system of $m \geq n$ (4.1) equations in n variables as we will show in Section 5.3.

Suppose we are given $n+1$ homogeneous polynomials $F_0, \ldots, F_n \in \mathbb{C}[x_0, \ldots, x_n]$ in $n+1$ variables $x_0, \ldots, x_n$, and assume that the total degree of F_i is $d_i > 0$. This means that each F_i can be written as

$$F_i = \sum_{|\alpha(j)|=d_i} c_{ij} \mathbf{x}^{\alpha(j)}, \tag{5.1}$$

where $\mathbf{x}^{\alpha(j)} = x_0^{\alpha_0(j)} x_1^{\alpha_1(j)} \ldots x_n^{\alpha_n(j)}$ is a monomial (4.2) of degree d_i, i.e. $\sum_{k=0}^{n} \alpha_k(j) = d_i$ and $c_{ij} \in \mathbb{C}$ is a coefficient of this monomial.

Such homogeneous polynomials can by obtained from polynomials $f_0, \ldots, f_n \in \mathbb{C}[x_1, \ldots, x_n]$ using a new variable x_0 and setting $F_i = x_0^{d_i} f_i(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}),\ i = 0, \ldots, n$.

Because all F_i are homogeneous of positive total degree, the system of polynomial equations

$$F_0(x_0, \ldots, x_n) = \cdots = F_n(x_0, \ldots, x_n) = 0 \tag{5.2}$$

always has the trivial solution $x_0 = \cdots = x_n = 0$. Hence, the question is whether there is a nontrivial solution of this system of polynomial equations (5.2). The answer to this question can be given using resultants.

To define the resultant we first introduce a variable u_{ij} for each possible pair of indices i, j in equation (5.1), i.e. for each coefficient c_{ij}. Then, given a polynomial $P \in \mathbb{C}[u_{ij}]$, i.e. a polynomial in variables u_{ij}, we let $P(F_0, \ldots, F_n)$ denote the number that we obtain by replacing each variable u_{ij} in P with the corresponding coefficient c_{ij}. The polynomial $P(F_0, \ldots, F_n)$ is known as the polynomial in the coefficients of the F_i. Then there holds the following theorem.

Theorem 7. *For fixed positive degrees $d_0, \ldots, d_n$ there is a unique polynomial $Res \in \mathbb{Z}[u_{ij}]$ which has the following properties:*

1. *if $F_0, \ldots, F_n \in \mathbb{C}[x_0, \ldots, x_n]$ are homogeneous of degrees $d_0, \ldots, d_n$, then the system of equations (5.2) has a nontrivial solution over $\mathbb{C}$ if and only if $Res(F_0, \ldots, F_n) = 0$,*
2. *$Res\left(x_0^{d_0}, \ldots, x_n^{d_n}\right) = 1$,*
3. *Res is irreducible, even when regarded as a polynomial in $\mathbb{C}[u_{ij}]$.*

Proof. The proof of this theorem can be found in [54]. □

$Res(F_0, \ldots, F_n)$ is called the resultant of $F_0, \ldots, F_n$. Sometimes $Res_{d_0,\ldots,d_n}$ is used to emphasize the dependence of the resultant on the degrees. In addition to the properties from Theorem 7 the resultant has several other interesting properties which are described for example in [39].

One simple example where it is easy to see how the resultant looks like is the system of linear polynomial equations.

Example 26. (Resultant of a linear system) *System of $n+1$ homogeneous linear polynomial equations in $n+1$ unknowns can be written in the form*

$$\begin{aligned} F_0 = c_{00}x_0 + \cdots + c_{0n}x_n &= 0, \\ &\cdots \\ F_n = c_{n0}x_0 + \cdots + c_{nn}x_n &= 0. \end{aligned} \tag{5.3}$$

It is known that system (5.3) has a nontrivial solution if and only if the determinant of the coefficient matrix $\mathtt{M} = (c_{ij})_{i,j=0}^{n}$ vanishes, i.e. $\det(\mathtt{M}) = 0$. This determinant is a polynomial in the coefficients c_{ij} and satisfies all three conditions from Theorem 7. Therefore we have $Res_{1,\ldots,1}(F_0, \ldots, F_n) = \det(\mathtt{M})$.

Next we will show the basic idea of the Macaulay's method of computing resultants for general systems of polynomial equations.

5.2 Computation of resultants

Suppose we have $n+1$ homogeneous polynomials $F_0, \ldots, F_n \in \mathbb{C}[x_0, \ldots, x_n]$ (5.1) of total degrees $d_0, \ldots, d_n$. Set

$$d = \sum_{i=0}^{n} (d_i - 1) + 1 = \sum_{i=0}^{n} d_i - n + 1. \tag{5.4}$$

For instance, when $(d_0, d_1, d_2) = (1, 2, 2)$ then $d = 3$.

Now take the set of all monomials $\mathbf{x}^\alpha = x_0^{\alpha_0} x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ of total degree d, i.e. $\sum_{i=0}^{n} \alpha_i = d$, and partition it into $n+1$ subsets

$$\begin{aligned} S_0 &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_0^{d_0} \mid \mathbf{x}^\alpha \right\}, \\ S_1 &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_0^{d_0} \nmid \mathbf{x}^\alpha \text{ but } x_1^{d_1} \mid \mathbf{x}^\alpha \right\}, \\ &\ldots \\ S_n &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_0^{d_0}, \ldots, x_{n-1}^{d_{n-1}} \nmid \mathbf{x}^\alpha \text{ but } x_n^{d_n} \mid \mathbf{x}^\alpha \right\}, \end{aligned} \tag{5.5}$$

where $x_i^{d_i} \mid \mathbf{x}^\alpha$ means that $x_i^{d_i}$ divides the monomial $\mathbf{x}^\alpha$ and $x_i^{d_i} \nmid \mathbf{x}^\alpha$ means that it does not.

Note that every monomial of total degree d in variables $x_0, \ldots, x_n$ lies in one of these sets and these sets are mutually disjoint. Moreover, if $\mathbf{x}^\alpha \in S_i$, then $x_i^{d_i} \mid \mathbf{x}^\alpha$ and $\mathbf{x}^\alpha / x_i^{d_i}$ is a monomial of total degree $d - d_i$.

Now we can create a system of equations that generalizes system (5.2)

$$\begin{aligned} \mathbf{x}^\alpha / x_0^{d_0}\, F_0 &= 0 \text{ for all } \mathbf{x}^\alpha \in S_0, \\ &\ldots \\ \mathbf{x}^\alpha / x_n^{d_n}\, F_n &= 0 \text{ for all } \mathbf{x}^\alpha \in S_n. \end{aligned} \tag{5.6}$$

This system has some special properties. Since F_i is a homogeneous polynomial of total degree d_i, the polynomial $\mathbf{x}^\alpha / x_i^{d_i}\, F_i$ is a homogeneous polynomial of total degree d for all $i = 0, \ldots, n$ and therefore can be written as a linear combination of monomials of degree d in $n+1$ variables $x_0, \ldots, x_n$. There exist $\binom{n+d}{d}$ monomials of degree d in $n+1$ variables.

The total number of equations in the system (5.6) is equal to the number of elements in sets $S_0, \ldots S_n$, which is also $\binom{n+d}{d}$ because these sets contain all monomials of degree d in variables $x_0, \ldots x_n$. Thus the system of polynomial equations (5.6) consists of $\binom{n+d}{d}$ homogeneous equations in $\binom{n+d}{d}$ monomials (all of degree d) in variables $x_0, \ldots, x_n$.

We can consider each monomial of total degree d in system (5.6) as an unknown and in this way get a system of $\binom{n+d}{d}$ linear equations in $\binom{n+d}{d}$ unknowns. Such system can be written in the form

$$\mathtt{M}\mathbf{X} = \mathbf{0}, \tag{5.7}$$

where $\mathtt{M}$ is a $\binom{n+d}{d} \times \binom{n+d}{d}$ coefficient matrix and $\mathbf{X}$ is a vector of $\binom{n+d}{d}$ monomials. We denote the determinant of this coefficient matrix $\mathtt{M}$ as D_n. This determinant has some interesting properties (see for example [39]) and is very close to the resultant of the initial system (5.2).

Macaulay [100] showed that the resultant is equal to this determinant D_n divided by a minor, i.e. the determinant of a submatrix of the $\binom{n+d}{d} \times \binom{n+d}{d}$ matrix $\mathtt{M}$. To obtain this submatrix we first need the following definition

Definition 1. *A monomial $\mathbf{x}^\alpha = x_0^{\alpha_0} \dots x_n^{\alpha_n}$ of total degree d is reduced if $x_i^{d_i}$ divides $\mathbf{x}^\alpha$ for exactly one i.*

We denote by $\mathtt{M}'$ the submatrix of the coefficient matrix $\mathtt{M}$ (5.7) of the extended system (5.6) that we obtain by deleting all rows and columns corresponding to reduced monomials $\mathbf{x}^\alpha$, and by D'_n its determinant. Then there holds the following theorem.

Theorem 8. *The resultant of the system of homogeneous polynomial equations (5.2) is given by*

$$Res\,(F_0, \dots, F_n) = \pm \frac{D_n}{D'_n} \tag{5.8}$$

whenever $D'_n \neq 0$.

Proof. The proof for this theorem can be found in Macaulay's paper [100] or in [54]. □

Note that the sets $S_0, \dots, S_n$ from (5.5) and the determinants D_n and D'_n are dependent on the ordering of the variables $x_0, \dots, x_n$. In this context, the index n in the notation D_n means that the variable x_n comes last when creating sets $S_0, \dots, S_n$. For different ordering of the variables with the last variable x_i for $i \in \{0, \dots, n-1\}$ we get different sets $S_0, \dots, S_n$ and slightly different extended system (5.6). For example, taking x_0 last means that S_0 consists of the monomials of the form

$$S_0 = \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_1^{d_1}, \dots, x_n^{d_n} \nmid \mathbf{x}^\alpha \text{ but } x_0^{d_0} \mid \mathbf{x}^\alpha \right\} = \tag{5.9}$$

$$= \left\{ \mathbf{x}^\alpha = x_0^{\alpha_0} x_1^{\alpha_1} \dots x_n^{\alpha_n} : 0 \leq a_i \leq d_i - 1 \text{ for } i > 0, |\alpha| = \sum_{i=0}^{n} a_i = d \right\}. \tag{5.10}$$

We will denote D_i the determinant of the coefficient matrix $\mathtt{M}$ of the extended system where x_i come last when creating sets $S_0, \dots, S_n$. Theorem 8 holds also for this determinant D_i and its minor D'_i.

Theorem 8 is universal and can be used to obtain the resultant for any system of $n+1$ homogeneous polynomial equations in $n+1$ unknowns (5.2). However, this theorem has some disadvantages. In general case, when coefficients of the polynomials in the system are not numerical, the computation of the resultant using formula (5.8) requires to divide two very large polynomials, which can be very time-consuming. On the other hand for numerical coefficients problems with vanishing of D'_n appear. The problem when $D'_n = 0$ can be solved using Canny's method which introduces a new variable u and considers the resultant $Res\left(F_0 - ux_0^{d_0}, \dots, F_n - ux_n^{d_n}\right)$. This method is described in [39].

In some cases, the resultant can be expressed as a single determinant like for the system of $n+1$ linear homogeneous equations in $n+1$ unknowns from Example 26 or in the case of Sylvester resultant (3.2) for two homogeneous equations in two unknowns. In these cases previously mentioned problems disappear. Therefore, it would be nice if this could be done for all resultants, i.e. all resultants could be expressed as a single determinant. Even though in some cases such formulas exist (e.g. for the resultant of three ternary quadrics which can be expressed by the 6×6 determinant [121] or for the resultant of three polynomials of degree r), most of the time we can't express the resultant in the form of a single determinant. Moreover, it is not known if this is possible in general.

Besides the Macaulay's formula from Theorem 8, there are other ways how to represent multivariate resultants as quotients. The best known are *Bezoutians* or *Bezout's resultants* [44] and *Dixon's resultants* [76]. The Macaulay formulation has however the advantage that the entries of the coefficient matrices are actually the coefficients of the initial polynomial equations, whereas in the case of Bezout or Dixon formulations the entries are polynomial functions of these coefficients [101].

Now we show the Macaulay's method for computing resultants on a simple example.

Example 27. (Macaulay's method for computing resultants) *Consider a system of three homogeneous polynomial equations in three unknowns x_0, x_1, x_2 and symbolic coefficients*

$$\begin{aligned} F_0 &= a_0x_0 + a_1x_1 + a_2x_2 = 0, \\ F_1 &= b_{00}x_0^2 + b_{01}x_0x_1 + b_{02}x_0x_2 + b_{11}x_1^2 + b_{12}x_1x_2 + b_{22}x_2^2 = 0, \\ F_2 &= c_{00}x_0^2 + c_{01}x_0x_1 + c_{02}x_0x_2 + c_{11}x_1^2 + c_{12}x_1x_2 + c_{22}x_2^2 = 0. \end{aligned} \tag{5.11}$$

The degrees of these equations are $d_0 = 1, d_1 = 2, d_3 = 2$. Therefore, d from Equation (5.4) is $d = \sum_{i=0}^{3}(d_i - 1) + 1 = 3$.

Now we take all monomials of degree 3 in variables x_0, x_1, x_2. There is $\binom{2+3}{3} = 10$ such monomials and these monomials are $x_0^3, x_0^2x_1, x_0^2x_2, x_0x_1^2, x_0x_2^2, x_0x_1x_2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3$. We partition these monomials into sets S_0, S_1 and S_3 as described in (5.5). In this case

$$\begin{aligned} S_0 &= \left\{x_0^3, x_0^2x_1, x_0^2x_2, x_0x_1^2, x_0x_2^2, x_0x_1x_2\right\}, \\ S_1 &= \left\{x_1^3, x_1^2x_2\right\}, \\ S_2 &= \left\{x_1x_2^2, x_2^3\right\}. \end{aligned} \tag{5.12}$$

We use these sets to create the extended system of polynomial equations in the form (5.6). In this way we obtain the following system of ten equations $x_0^2F_0 = x_0x_1F_0 = x_0x_2F_0 = x_1^2F_0 = x_2^2F_0 = x_1x_2F_0 = x_1F_1 = x_2F_1 = x_1F_2 = x_2F_2 = 0$.

These ten equations can be written in the matrix form

$$\begin{pmatrix} a_0 & a_1 & a_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_0 & 0 & a_1 & 0 & a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_0 & 0 & a_2 & a_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_0 & 0 & 0 & a_1 & a_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_0 & 0 & 0 & 0 & a_1 & a_2 \\ 0 & 0 & 0 & 0 & 0 & a_0 & 0 & a_1 & a_2 & 0 \\ 0 & b_{00} & 0 & b_{01} & 0 & b_{02} & b_{11} & b_{12} & b_{22} & 0 \\ 0 & 0 & b_{00} & 0 & b_{02} & b_{01} & 0 & b_{11} & b_{12} & b_{22} \\ 0 & c_{00} & 0 & c_{01} & 0 & c_{02} & c_{11} & c_{12} & c_{22} & 0 \\ 0 & 0 & c_{00} & 0 & c_{02} & c_{01} & 0 & c_{11} & c_{12} & c_{22} \end{pmatrix} \begin{pmatrix} x_0^3 \\ x_0^2 x_1 \\ x_0^2 x_2 \\ x_0 x_1^2 \\ x_0 x_2^2 \\ x_0 x_1 x_2 \\ x_1^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (5.13)$$

We denote the 10×10 *coefficient matrix in (5.13) as* $\mathtt{M}$ *and its determinant* D_2.

To compute the resultant using Macaulay's formula from Theorem 8 we need to compute the determinant of a submatrix of this coefficient matrix. We obtain this submatrix by deleting all rows and columns corresponding to reduced monomials, i.e. monomials $\mathbf{x}^\alpha = x_0^{\alpha_0} x_1^{\alpha_1} x_2^{\alpha_2}$ *of total degree* 3 *which are divisible by exactly one* $x_i^{d_i}$, $i \in \{0, 1, 2\}$ *and* $d_0 = 1, d_1 = 2, d_2 = 2$.

In this case reduced monomials are monomials $x_0^3, x_0^2 x_1, x_0^2 x_2, x_0 x_1 x_2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3$. *The only two remaining monomials are monomials* $x_0 x_1^2$ *and* $x_0 x_2^2$, *which correspond to the fourth and fifth column and row of the coefficient matrix in (5.13). Therefore, after deleting all rows and columns corresponding to the reduced monomials we obtain the* 2×2 *matrix*

$$\mathtt{M}' = \begin{pmatrix} a_0 & 0 \\ 0 & a_0 \end{pmatrix}. \quad (5.14)$$

We denote the determinant of this matrix $\mathtt{M}'$ *as* D_2'.

Then the resultant of the system of three homogeneous polynomial equations in three unknowns (5.11) is according to Theorem 8

$$Res_{1,2,2}(F_0, F_1, F_2) = \pm \frac{det(\mathtt{M})}{det(\mathtt{M}')} = \pm \frac{D_2}{D_2'}, \quad (5.15)$$

whenever $D_2 \neq 0$.

5.3 Standard resultant based methods for solving systems of polynomial equations

In this section, we will show how resultants can be used to solve systems of polynomial equations. We will briefly discuss two standard resultant based methods, the u-resultant method and the hidden variable method, and we will also show how these methods are connected to multiplication matrices described in Section 4.3.3 and to polynomial eigenvalue problems [9].

Let us assume that we have n polynomials $f_1, \ldots, f_n$ in n variables $x_1, \ldots, x_n$ or equivalently n homogeneous polynomials $F_1, \ldots, F_n$ of degree $d_1, \ldots, d_n$ in $n + 1$ variables $x_0, \ldots, x_n$.

Polynomials $f_1, \ldots, f_n$ can be obtained from homogeneous polynomials $F_1, \ldots, F_n$ by setting $x_0 = 1$

$$f_i(x_1, \ldots, x_n) = F_i(1, x_1, \ldots, x_n). \tag{5.16}$$

Note that we have one polynomial less than in the previous section. It is because our goal in this section is not to create the resultant of the system of polynomial equations and in this way determine if the system has a common solution or not but our goal is to find all nontrivial solutions of the system of polynomial equations

$$F_1(x_0, \ldots, x_n) = \cdots = F_n(x_0, \ldots, x_n) = 0 \tag{5.17}$$

or equivalently to find all solutions of the dehomogenized system

$$f_1(x_1, \ldots, x_n) = \cdots = f_n(x_1, \ldots, x_n) = 0. \tag{5.18}$$

Since we have n polynomials in n variables we have a slightly different problem than Problem 1 from the previous Chapter 4 about Gröbner basis methods in which we were assuming m polynomials in n variables with $m \geq n$. However, in Section 5.5 we will show that resultant based methods can be extended to systems of $m \geq n$ polynomial equations in n variables, i.e. to solve Problem 1.

5.3.1 Solving systems of equations using the u-resultant

The basic idea of the first resultant based method for solving systems of polynomial equations is to add a new polynomial equation $f_0 = 0$ to the initial system of polynomial equations (5.18). The standard form of the added polynomial f_0 is

$$f_0 = u_0 + u_1 x_1 + \cdots + u_n x_n. \tag{5.19}$$

where $u_0, \ldots, u_n$ are independent variables, which will be regarded as constants.

Note that in the case of homogeneous polynomial equations $F_1, \ldots, F_n$ the new polynomial has the form

$$F_0 = u_0 x_0 + u_1 x_1 + \cdots + u_n x_n. \tag{5.20}$$

After adding the new polynomial f_0 to the system (5.18) we obtain $n+1$ equations in n variables. In the case of the homogeneous system (5.17) after adding polynomial F_0 (5.20), the number of homogeneous equations in this system equals the number of variables. Therefore we have enough equations to create the resultant $Res_{1,d_1,\ldots,d_n}(f_0, f_1, \ldots, f_n)$, respectively $Res_{1,d_1,\ldots,d_n}(F_0, F_1, \ldots, F_n)$. This resultant is called the u-resultant and it is a polynomial in $u_0, \ldots, u_n$. The u-resultant can be then used to find all solutions of the system of polynomial equations (5.18), respectively (5.17) using the following proposition.

Proposition 1. *Assume that $f_1 = \cdots = f_n = 0$ have total degrees bounded by $d_1, \ldots, d_n$, no solutions at infinity, and all solutions of multiplicity one. If $f_0 = u_0 + u_1 x_1 + \cdots + u_n x_n$, where $u_0, \ldots, u_n$ are independent variables, then there is a nonzero constant c such that*

$$Res_{1,d_1,\ldots,d_n}(f_0, \ldots, f_n) = c \prod_{p \in V(f_1,\ldots,f_n)} f_0(p), \tag{5.21}$$

where $V(f_1, \ldots, f_n)$ is the affine variety defined by $f_1, \ldots, f_n$, i.e. the set of all solutions of the system $f_1 = \cdots = f_n = 0$.

Proof. The proof of this proposition can be found in [39]. □

The "no solutions at infinity" in this proposition means that the system of polynomial equations $F_1(0, x_1, \ldots, x_n) = \cdots = F_n(0, x_1, \ldots, x_n) = 0$ has no nontrivial solutions.

To see in more detail what Proposition 1 says, let the points of the affine variety $V(f_1, \ldots, f_n)$, i.e. the solutions of (5.18), be $\mathbf{p}_i = (a_{i1}, \ldots, a_{in})$, $i = 1, \ldots, N$. From Bézout's theorem [39] it follows that $N \leq d_1 d_2 \ldots d_n$. Then the u-resultant (5.21) from Proposition 1 is given by

$$Res_{1,d_1,\ldots,d_n}(f_0, \ldots, f_n) = c \prod_{i=1}^{N} (u_0 + a_{i1} u_1 + \cdots + a_{in} u_n). \tag{5.22}$$

Therefore, to find the solutions of the system of polynomial equations (5.18), we first need to compute $Res_{1,d_1,\ldots,d_n}(f_0, \ldots, f_n)$, factor it into linear factors, and then read off the solutions from the coefficients of these linear factors.

Unfortunately, this has several limitations. First, for some systems it is necessary to compute symbolic determinants of large size which is not easy. Second, multivariable factorization is very hard especially in floating point arithmetic. Possible solutions to these problems are discussed in [102, 39].

Example 28. (U-resultant) *To show how the presented u-resultant method works, let us consider our system of polynomials equations (4.30) form Example 4, i.e. the system*

$$\begin{aligned} 2x_1^2 + x_2^2 + 3x_2 - 12 &= 0, \\ x_1^2 - x_2^2 + x_1 + 3x_2 - 4 &= 0. \end{aligned} \tag{5.23}$$

Note that we have substituted x_1 for x and x_2 for y in (4.30), to be consistent with the notation used in this section.

We know from Bézout's Theorem that this system of polynomial equations has at most four solutions. Now we show how we can find these solutions using the u-resultant method.

In this method we add the equation $f_0 = u_0 + u_1 x_1 + u_2 x_2 = 0$, where u_0, u_1 and u_2 are independent variables, to equations (5.23).

After homogenizing all three equations using x_0, we obtain the following system of three homogeneous equations in three unknowns x_0, x_1 and x_2

$$\begin{aligned} F_0 &= u_0 x_0 + u_1 x_2 + u_2 x_2 = 0, \\ F_1 &= 2x_1^2 + x_2^2 + 3x_0 x_2 - 12x_0^2 = 0, \\ F_2 &= x_1^2 - x_2^2 + x_0 x_1 + 3x_0 x_2 - 4x_0^2 = 0. \end{aligned} \tag{5.24}$$

This system has the form of the system (5.11) from previous Example 27. In this example we have shown that the Macaulay's method for computing resultants presented in Section 5.2 with ordering of variables x_0, x_1 and x_2 results in the sets S_i (5.5) of the form

$$\begin{aligned} S_0 &= \left\{x_0^3, x_0^2x_1, x_0^2x_2, x_0x_1^2, x_0x_2^2, x_0x_1x_2\right\}, \\ S_1 &= \left\{x_1^3, x_1^2x_2\right\}, \\ S_2 &= \left\{x_1x_2^2, x_2^3\right\}. \end{aligned} \tag{5.25}$$

Using these sets we obtain extended system (5.6) of ten polynomial equations $x_0^2F_0 = x_0x_1F_0 = x_0x_2F_0 = x_1^2F_0 = x_2^2F_0 = x_1x_2F_0 = x_1F_1 = x_2F_1 = x_1F_2 = x_2F_2 = 0$.

In this case this extended system can be written in the matrix form

$$\begin{pmatrix} u_0 & u_1 & u_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_0 & 0 & u_1 & 0 & u_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_0 & 0 & u_2 & u_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_0 & 0 & 0 & u_1 & u_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_0 & 0 & 0 & 0 & u_1 & u_2 \\ 0 & 0 & 0 & 0 & 0 & u_0 & 0 & u_1 & u_2 & 0 \\ 0 & -12 & 0 & 0 & 0 & 3 & 2 & 0 & 1 & 0 \\ 0 & 0 & -12 & 0 & 3 & 0 & 0 & 2 & 0 & 1 \\ 0 & -4 & 0 & 1 & 0 & 3 & 1 & 0 & -1 & 0 \\ 0 & 0 & -4 & 0 & 3 & 1 & 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_0^3 \\ x_0^2x_1 \\ x_0^2x_2 \\ x_0x_1^2 \\ x_0x_2^2 \\ x_0x_1x_2 \\ x_1^3 \\ x_1^2x_2 \\ x_1x_2^2 \\ x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5.26}$$

Note that this matrix equation corresponds to the matrix equation (5.13) from Example 27 with coefficients from our input equations (5.24).

The determinant of the 10×10 coefficient matrix $\mathtt{M}$ from (5.26) is denoted by D_2 and has the form

$$\begin{aligned} D_2 &= u_0^2(9u_0^4 + 18u_0^3u_2 - 6u_0^3u_1 + 27u_0^2u_2u_1 - 77u_0^2u_1^2 - 13u_0^2u_2^2 + 145u_0u_2^2u_1 \\ &\quad -195u_0u_2u_1^2 + 50u_0u_1^3 - 30u_0u_2^3 - 212u_1^2u_2^2 + 112u_1^4 + 96u_2u_1^3 + 84u_2^3u_1 - 8u_2^4). \end{aligned} \tag{5.27}$$

As it was shown in Example 27, the submatrix $\mathtt{M}'$ obtained after deleting all rows and columns of $\mathtt{M}$ corresponding to the reduced monomials has the form

$$\mathtt{M}' = \begin{pmatrix} u_0 & 0 \\ 0 & u_0 \end{pmatrix} \tag{5.28}$$

and its determinant is $D_2' = u_0^2$.

Then the resultant of the system of three homogeneous polynomial equations in three unknowns (5.24) is according to Theorem 8

$$Res\,(F_0, F_1, F_2) = \pm\frac{D_2}{D_2'}, \tag{5.29}$$

which is in this case

$$\begin{aligned} Res\left(F_0,F_1,F_2\right) &= \pm(9u_0^4+18u_0^3u_2-6u_0^3u_1+27u_0^2u_2u_1-77u_0^2u_1^2 \\ &\quad -13u_0^2u_2^2+145u_0u_2^2u_1-195u_0u_2u_1^2+50u_0u_1^3 \\ &\quad -30u_0u_2^3-212u_1^2u_2^2+112u_1^4+96u_2u_1^3+84u_2^3u_1-8u_2^4). \end{aligned} \tag{5.30}$$

This polynomial $Res\left(F_0,F_1,F_2\right)$ *(5.30) can be factorized as*

$$\pm\left(u_0-2u_1+u_2\right)\left(3u_0-7u_1+u_2\right)\left(3u_0+8u_1-4u_2\right)\left(u_0+u_1+2u_2\right). \tag{5.31}$$

From Proposition 1 we know that this u-resultant $Res\left(F_0,F_1,F_2\right)$ *can be written as*

$$Res\left(F_0,F_1,F_2\right)=c\prod_{i=1}^{4}\left(u_0+a_{i1}u_1++a_{i2}u_2\right), \tag{5.32}$$

where $c\in\mathbb{C}$ *is some non-zero constant and* $\mathbf{p}_i=\left(a_{i1},a_{i2}\right),\ i=1,\ldots,4$ *are solutions of the initial system of polynomial equation (5.23).*

After rewriting the factorized u-resultant (5.31) to the form (5.32) we obtain

$$\pm 9\left(u_0-2u_1+u_2\right)\left(u_0-\frac{7}{3}u_1+\frac{1}{3}u_2\right)\left(u_0+\frac{8}{3}u_1-\frac{4}{3}u_2\right)\left(u_0+u_1+2u_2\right). \tag{5.33}$$

Coefficients of u_1 *and* u_2 *in each factor give us the four solutions of the initial system of polynomial equations (5.23)*

$$\left(-2,1\right),\left(-\frac{7}{3},\frac{1}{3}\right),\left(\frac{8}{3},-\frac{4}{3}\right),\left(1,2\right). \tag{5.34}$$

Note that in this case it is slightly easier to compute the u-resultant $Res\left(F_0,F_1,F_2\right)$ *as* $Res\left(F_0,F_1,F_2\right)=\pm\frac{D_0}{D_0'}$*, i.e. with the ordering of variables* x_2,x_1 *and* x_0*. For such an ordering the sets* S_i *have the form*

$$\begin{aligned} S_0 &= \left\{x_0^3,x_0^2x_1,x_0^2x_2,x_0x_1x_2\right\}, \\ S_1 &= \left\{x_0x_1^2,x_1^3,x_1^2x_2\right\}, \\ S_2 &= \left\{x_0x_2^2,x_1x_2^2,x_2^3\right\}, \end{aligned} \tag{5.35}$$

and the determinant D_0' *is numerical, i.e. it doesn't contain variables* u_i*. Therefore, to find solutions of the initial system of polynomial equations it is sufficient to compute only the determinant* D_0*. This holds for all systems of polynomial equations and orderings of variables where* x_0 *is the last variable. However, to obtain solutions we will still need to compute quite a large symbolic determinant and to perform complicated multivariable factorization.*

5.3.2 Solving systems of equations using hidden variables

The second, usually more practical resultant based method for solving systems of polynomial equations is the hidden variable method. The main idea of this method is to consider one of variables as a constant and then compute the resultant.

Consider the problem of finding all solutions of the system (5.18) of n polynomial equations $f_1(x_1, \ldots, x_n) = \cdots = f_n(x_1, \ldots, x_n) = 0$ in n variables $x_1, \ldots, x_n$.

From Section 5.1 we know that to compute the resultant we need to have one more polynomial equation than variables. This means that we cannot compute the resultant for (5.18) directly.

One way how to solve this is to add a new polynomial equation to our n equations as it was shown in the previous u-resultant method. An alternative approach is to consider one of the variables, e.g. x_n, as a constant and in this way obtain n polynomials in $n-1$ variables $x_1, \ldots, x_{n-1}$

$$f_1, \ldots, f_n \in (\mathbb{C}[x_n])[x_1, \ldots, x_{n-1}]. \tag{5.36}$$

We sometimes say that we "hide" the variable x_n, which gives also the name of this method.

Since now we have one more polynomial than variables we can compute the resultant of these polynomials (5.36). This resultant is sometimes called the hidden variable resultant and is usually denoted as

$$Res^{x_n}_{d_1,\ldots,d_n}(f_1, \ldots, f_n), \tag{5.37}$$

where the superscript x_n means that we are regarding x_n as a constant.

From the resultant definition we know that the resultant is a polynomial in the coefficients of the input polynomials $f_1, \ldots, f_n$. Since in our case these input polynomials have coefficients from $\mathbb{C}[x_n]$, the hidden variable resultant (5.37) is a polynomial in x_n. Moreover, this polynomial has some special property which can be used to solve the initial system of polynomial equations (5.18) and which is stated in the following proposition.

Proposition 2. *The hidden variable resultant $Res^{x_n}_{d_1,\ldots,d_n}(f_1, \ldots, f_n)$, is generically a polynomial in x_n whose roots are the x_n-coordinates of the solutions of the system of polynomial equations (5.18), i.e. of the system $f_1(x_1, \ldots, x_n) = \cdots = f_n(x_1, \ldots, x_n) = 0$.*

Proof. The proof of this proposition can be found in [39]. □

Here by generically we mean that this property holds for "most" polynomials $f_1, \ldots, f_n$ [39].

Note that in this case, like in the case of the u-resultant, when computing the hidden variable resultant as $Res^{x_n}_{d_1,\ldots,d_n}(f_1, \ldots, f_n) = \pm\frac{D_0}{D_0'}$, the denominator D_0' doesn't involve x_n [39] and we can therefore ignore it.

Example 29. (Hidden variable resultant) *Let's return to our system of two polynomial equations form the previous example*

$$\begin{aligned} 2x_1^2 + x_2^2 + 3x_2 - 12 &= 0, \\ x_1^2 - x_2^2 + x_1 + 3x_2 - 4 &= 0. \end{aligned} \tag{5.38}$$

Here we show how this system can be solved using the hidden variable method.

First consider x_2 as a constant, i.e. "hide" x_2 in the coefficient field. This gives us two polynomial equations in one variable x_1 and coefficients from $\mathbb{C}[x_2]$

$$\begin{aligned} 2x_1^2 + \left(x_2^2 + 3x_2 - 12\right) &= 0, \\ x_1^2 + x_1 + \left(-x_2^2 + 3x_2 - 4\right) &= 0. \end{aligned} \tag{5.39}$$

For two polynomials in one variable we can easily compute the resultant using the well known Sylvester matrix (3.2). This results in

$$Res^{x_2}(f_1, f_2) = \det \begin{pmatrix} 2 & 0 & x_2^2 + 3x_2 - 12 & 0 \\ 0 & 2 & 0 & x_2^2 + 3x_2 - 12 \\ 1 & 1 & -x_2^2 + 3x_2 - 4 & 0 \\ 0 & 1 & 1 & -x_2^2 + 3x_2 - 4 \end{pmatrix} \tag{5.40}$$

from which we obtain

$$Res^{x_2}(f_1, f_2) = 9x_2^4 - 18x_2^3 - 13x_2^2 + 30x_2 - 8. \tag{5.41}$$

From Proposition 2 we know that the roots of this polynomial $Res^{x_2}(f_1, f_2)$ give us x_2-coordinates of the solutions of the initial system (5.38). This results to four solutions $x_2 = 1,\ 2,\ \frac{1}{3},\ -\frac{4}{3}$.

To obtain solutions for x_1 we can compute the hidden variable resultant $Res^{x_1}(f_1, f_2)$, or, in this case, simply backsubstitute all obtained solutions for x_2 into the input polynomial equations (5.38). In this way we obtain four solutions

$$(-2, 1), (1, 2), \left(-\frac{7}{3}, \frac{1}{3}\right), \left(\frac{8}{3}, -\frac{4}{3}\right). \tag{5.42}$$

The hidden variable method results in simpler computations than the u-resultant method presented in the previous section. It is because the hidden variable method involves resultants with fewer equations and variables than the u-resultant. Moreover, the hidden variable method doesn't require multivariable factorization and results only in finding roots of a single variable polynomial, e.g., by finding the eigenvalues of the companion matrix or using Sturm sequences [66].

One disadvantage of the hidden variable method is that it only results in solutions for individual variables and to compute complete solutions it is usually necessary to compute the hidden variable resultants $Res^{x_i}_{d_1,\ldots,d_n}(f_1, \ldots, f_n)$ for all variables x_i, or to use solutions for one variable to obtain solutions for the remaining variables in a different way. Moreover, the solutions for individual variables needs to be combined to correct solutions of the input system of polynomial equations.

One way how to avoid this problem is to consider the extended system of polynomial equations (5.6) used to compute the hidden variable resultant as $Res^{x_n}_{d_1,\ldots,d_n}(f_1, \ldots, f_n) = \pm\frac{D_i}{D_i'}$, i.e. the system from which D_i is computed as the determinant of its coefficient matrix.

Such an extended dehomogenized system can be written as

$$\mathtt{C}(x_n)\,\mathbf{X} = \mathbf{0}, \tag{5.43}$$

where $\mathtt{C}(x_n)$ is a square coefficient matrix with elements from $\mathbb{C}[x_n]$, for which it holds $D_i = \det(\mathtt{C}(x_n))$, and $\mathbf{X}$ is a vector of monomials in variables $x_1, \ldots, x_{n-1}$.

Generically, (5.43) is a polynomial eigenvalue problem [9], which can be transformed to the generalized eigenvalue problem and solved using standard eigenvalue methods [9]. By solving this polynomial eigenvalue problem, we can obtain solutions for x_n and for the monomial vector $\mathbf{X}$, from which we can extract solutions for the remaining variables $x_1, \ldots, x_{n-1}$. In this way we can find solutions of the whole system.

Next we will discuss polynomial eigenvalue problems (PEP) in more detail which we will use to solve various computer vision problems.

5.4 Polynomial eigenvalue problems

Polynomial eigenvalue problems (PEP) are problems of the form

$$\mathtt{C}(\alpha)\,\mathbf{v} = \mathbf{0}, \tag{5.44}$$

where $\mathbf{v}$ is a vector of monomials in all variables except for α and $\mathtt{C}(\alpha)$ is a matrix polynomial in variable α defined as

$$\mathtt{C}(\alpha) \equiv \alpha^l \mathtt{C}_l + \alpha^{l-1}\mathtt{C}_{l-1} + \cdots + \alpha\mathtt{C}_1 + \mathtt{C}_0, \tag{5.45}$$

where $\mathtt{C}_j$'s are $n \times n$ coefficient matrices [9].

We next describe how these problems can be solved by transforming them to the generalized eigenvalue problems.

5.4.1 Transformation to the standard generalized eigenvalue problem

Polynomial eigenvalue problems (5.44) can be transformed to the standard generalized eigenvalue problems (GEP)

$$\mathtt{A}\,\mathbf{y} = \alpha\,\mathtt{B}\,\mathbf{y}. \tag{5.46}$$

GEPs (5.46) are well studied problems and there are many efficient numerical algorithms for solving them [9]. A very useful method for dense and small or moderate-sized GEPs is the QZ algorithm [9], with time complexity $O\left(n^3\right)$ and memory complexity $O\left(n^2\right)$. Different algorithms are usually used for large scale GEPs. A common approach for large scale GEPs is to reduce them to standard eigenvalue problems and then apply iterative methods.

Algorithms for solving GEPs and standard eigenvalue problems are available in almost all mathematical software and libraries, for example in LAPACK, ARPACK, or MATLAB which provides `polyeig` function for solving polynomial eigenvalue problems (5.44) of arbitrary degree (including their transformation to the generalized eigenvalue problems (5.46)).

Quadratic eigenvalue problems

To see how a PEP (5.44) can be transformed to a GEP (5.46) let us first consider an important class of polynomial eigenvalue problems, the quadratic eigenvalue problems (QEP) of the form

$$\left(\alpha^2 \mathtt{C}_2 + \alpha \mathtt{C}_1 + \mathtt{C}_0\right) \mathbf{v} = \mathbf{0}, \tag{5.47}$$

where $\mathtt{C}_2, \mathtt{C}_1$ and $\mathtt{C}_0$ are coefficient matrices of size $n \times n$, α is a variable called an eigenvalue and $\mathbf{v}$ is a vector of monomials in all variables with the exception of α, called an eigenvector.

QEP (5.47) can be transformed to the generalized eigenvalue problem (5.46) with

$$\mathtt{A} = \begin{pmatrix} \mathtt{0} & \mathtt{I} \\ -\mathtt{C}_0 & -\mathtt{C}_1 \end{pmatrix}, \mathtt{B} = \begin{pmatrix} \mathtt{I} & \mathtt{0} \\ \mathtt{0} & \mathtt{C}_2 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} \mathbf{v} \\ \alpha\mathbf{v} \end{pmatrix}. \tag{5.48}$$

Here, 0 and $\mathtt{I}$ are $n \times n$ null and identity matrices, respectively. GEP (5.46) with matrices (5.48) gives equations $\alpha\mathbf{v} = \alpha\mathbf{v}$ and $-\mathtt{C}_0\mathbf{v} - \alpha\mathtt{C}_1\mathbf{v} = \alpha^2\mathtt{C}_2\mathbf{v}$ which is equivalent to (5.47). Note, that this GEP (5.48) has $2\,n$ eigenvalues and therefore by solving it we obtain $2\,n$ solutions to the QEP (5.47).

Higher order eigenvalue problems

Higher order PEPs of degree l,

$$\left(\alpha^l \mathtt{C}_l + \alpha^{l-1}\mathtt{C}_{l-1} + \cdots + \alpha\mathtt{C}_1 + \mathtt{C}_0\right) \mathbf{v} = \mathbf{0}, \tag{5.49}$$

can be also transformed to the generalized eigenvalue problem (5.46). Here

$$\mathtt{A} = \begin{pmatrix} \mathtt{0} & \mathtt{I} & \mathtt{0} & \dots & \mathtt{0} \\ \mathtt{0} & \mathtt{0} & \mathtt{I} & \dots & \mathtt{0} \\ \dots & \dots & \dots & \dots & \dots \\ -\mathtt{C}_0 & -\mathtt{C}_1 & -\mathtt{C}_2 & \dots & -\mathtt{C}_{l-1} \end{pmatrix}, \tag{5.50}$$

$$\mathtt{B} = \begin{pmatrix} \mathtt{I} & & & \\ & \dots & & \\ & & \mathtt{I} & \\ & & & \mathtt{C}_l \end{pmatrix}, \mathbf{y} = \begin{pmatrix} \mathbf{v} \\ \alpha\mathbf{v} \\ \dots \\ \alpha^{l-1}\mathbf{v} \end{pmatrix}.$$

For higher order PEPs, one has to work with larger matrices with $n\,l$ eigenvalues. Therefore, for larger values of n and l convergence problems when solving these problems may appear [9].

Note that if the leading matrix $\mathtt{C}_l$ is nonsingular and well conditioned, then we can consider a monic matrix polynomial

$$\overline{\mathtt{C}\,(\alpha)} = \mathtt{C}_l^{-1}\mathtt{C}\,(\alpha)\,, \tag{5.51}$$

with coefficient matrices $\overline{\mathtt{C}}_i = \mathtt{C}_l^{-1}\mathtt{C}_i$, $i = 0 \dots l-1$. Then, the PEP (5.49) can be transformed directly to the eigenvalue problem

$$\mathtt{A}\mathbf{y} = \alpha\mathbf{y}, \tag{5.52}$$

where

$$\mathtt{A} = \begin{pmatrix} \mathtt{0} & \mathtt{I} & \mathtt{0} & \dots & \mathtt{0} \\ \mathtt{0} & \mathtt{0} & \mathtt{I} & \dots & \mathtt{0} \\ \dots & \dots & \dots & \dots & \dots \\ -\overline{\mathtt{C}}_0 & -\overline{\mathtt{C}}_1 & -\overline{\mathtt{C}}_2 & \dots & -\overline{\mathtt{C}}_{l-1} \end{pmatrix}. \tag{5.53}$$

In this case, the matrix $\mathtt{A}$ (5.53) is sometimes called a block companion matrix or generalized companion matrix.

It happens quite often that the leading matrix $\mathtt{C}_l$ is singular, but the last matrix $\mathtt{C}_0$ is regular and well conditioned. Then, either the described method which transforms PEP (5.49) to GEP (5.50) or the transformation $\beta = 1/\alpha$ can be used. The transformation $\beta = 1/\alpha$ reduces the problem to finding eigenvalues of the matrix

$$\mathtt{A} = \begin{pmatrix} \mathtt{0} & \mathtt{I} & \mathtt{0} & \dots & \mathtt{0} \\ \mathtt{0} & \mathtt{0} & \mathtt{I} & \dots & \mathtt{0} \\ \dots & \dots & \dots & \dots & \dots \\ -\mathtt{C}_0^{-1}\mathtt{C}_l & -\mathtt{C}_0^{-1}\mathtt{C}_{l-1} & -\mathtt{C}_0^{-1}\mathtt{C}_{l-2} & \dots & -\mathtt{C}_0^{-1}\mathtt{C}_1 \end{pmatrix}. \tag{5.54}$$

In some cases other linear rational transformations which improve the conditioning of the leading matrix can be used [9].

5.5 Specialized resultant based methods for solving systems of polynomial equations

In Section 5.3 we have presented two standard resultant based methods for solving systems of polynomial equations. As in the case of standard Gröbner basis methods presented in Section 4.3 these methods were developed for general systems of polynomial equations. Moreover, in this case, the resultant based methods assume dense systems of n polynomial equations in n unknowns with generic coefficients.

However, our goal is not to solve general systems of dense polynomial equations, but to solve systems that arise from a particular problem. Therefore, in this section we will present a specialized resultant based method for solving systems of polynomial equations in the form of one "representing system" F, see Definition 20. Like in the case of specialized Gröbner basis method presented in Section 4.4 this specialized resultant based method is suitable especially for problems where all "interesting instances" of the problem can be solved by following one or only a very small number of "solutions templates".

The proposed specialized resultant based method is based on the hidden variable resultants and the polynomial eigenvalue problems presented in Section 5.4. The main idea of this method is the transformation of the initial system of polynomial equations to the polynomial eigenvalue problem (5.44) by hiding a variable and using the modified Macaulay's method of computing resultants. Therefore, we will call this specialized resultant based method also the polynomial eigenvalue method.

Like in the case of the specialized Gröbner basis method the resultant based method can be divided into two phases, into the "offline" and the "online" phase.

1. **Offline phase:** In this phase we are trying to find a formulation of a particular problem as a polynomial eigenvalue problem (5.44) and to design an efficient specific solver for this problem. For a particular problem and its set S of "consistent" instances, which are all in the form of one "representing system" F, see Definition 20, this phase needs to be performed only once.

2. **Online phase:** In the "online phase", the efficient specific solver is used to solve concrete instances of the particular problem represented by F.

Next we will describe these two phases in more detail.

5.5.1 Offline phase

The offline phase of the proposed specialized resultant based method consists of two main steps. In the first step we transform the system of polynomial equations F, representing considered instances of the particular problem, to PEP (5.44). In the second step we reduce the size of this PEP.

Now we describe the first step of transforming a system of polynomial equations to a PEP (5.44).

Transformation of systems of polynomial equations to a PEP

Let's consider systems of polynomial equations in the form of one "representing system" of m polynomial equations in n variables

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0, \\ &\ldots, \\ f_m(x_1,\ldots,x_n) &= 0, \end{aligned} \tag{5.55}$$

with coefficients from $\mathbb{C}$ and a finite number of solutions.

Our goal in this step is to transform this system of polynomial equations to a polynomial eigenvalue problem (5.44).

If we are lucky, like in the case of some of the computer vision problems presented in Chapter 7, then for some x_j, let's say x_1, equations (5.55) can be directly rewritten to a polynomial eigenvalue problem

$$\mathtt{C}(x_1)\,\mathbf{v} = \mathbf{0}, \tag{5.56}$$

where $\mathtt{C}(x_1)$ is a matrix polynomial (5.45) with square $m \times m$ coefficient matrices $\mathtt{C}_i$ and $\mathbf{v}$ is a vector of s monomials in variables $x_2,\ldots,x_n$, i.e. monomials of the form $\mathbf{x}^\alpha = x_2^{\alpha_2} x_3^{\alpha_3} \ldots x_n^{\alpha_n}$. In this case the number of monomials s is equal to the number of equations m, i.e. $s = m$.

Unfortunately, not all systems of polynomial equations (5.55) can be directly transformed to a PEP (5.56) for some x_j. This happens when, after rewriting the system (5.55) to the form (5.56), we remain with fewer equations than monomials in these equations, i.e. $s > m$, and therefore we do not have square coefficient matrices $\mathtt{C}_i$. In such a case, we have to generate new equations as polynomial combinations of initial equations (5.55). This has to be done in a special way

to get as many linearly independent equations as monomials, after rewriting the system to the form (5.56). Then, we can treat each monomial as a new variable and look at this system as at a linear one. Therefore, we sometimes say that we "linearize" our system of equations.

In Section 5.3.2 we use Macaulay's method for computing resultants presented in Section 5.2 to generate an extended system that has the same number of equations as monomials in these equations after "hiding" one variable in the coefficient field. This "Macaulay based method" for generating as many polynomials as monomials in these polynomials looks trivial, but unfortunately it doesn't work for all systems of polynomial equations. Macaulay's method for computing resultants was designed for dense systems of polynomial equations with generic coefficients. For sparse systems of polynomial equations or systems with some special coefficients it may generate linearly dependent equations. For such linearly depended polynomial equations the formulation (5.56) will result in singular coefficient matrices $\mathtt{C}_i$ and couldn't be solved as a polynomial eigenvalue problem.

Therefore, for our purpose we will modify this "Macaulay based method" for generating polynomials. Before describing this modification we first once more review the standard Macaulay's method for computing resultants presented in Section 5.2. It is because in our modified "Macaulay based method" we "hide" one variable in the coefficient field, and therefore we use different notation comparing to the notation used in Section 5.2.

Macaulay based method

Let us now review the standard hidden variable resultant method, which is based on Macauly's formulation of resultants and which can be in some cases used to transform a system of polynomial equations to a PEP (5.56). This method was designed for systems of n polynomial equations in n unknowns; however, it can be applied also to $m \geq n$ polynomial equations (5.55) in n unknowns. Since we will discuss the extension to $m > n$ equations later, we will first consider a system of n polynomial equations in n unknowns, i.e. the system

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0, \\ &\ldots \\ f_n(x_1, \ldots, x_n) &= 0. \end{aligned} \tag{5.57}$$

Assume that we want to formulate these equations as a PEP (5.56) for x_1. This means that we "hide" x_1 in the coefficient field and consider these equations as n equations in $n-1$ variables $x_2, \ldots, x_n$ and coefficients from $\mathbb{C}[x_1]$, i.e.

$$f_1, \ldots, f_n \in (\mathbb{C}[x_1])[x_2, \ldots, x_n]. \tag{5.58}$$

Let the degrees of these equations in variables $x_2, \ldots, x_n$ be $d_1, d_2, \ldots, d_n$, respectively.

Now consider the system of n homogeneous polynomial equations in n unknowns

$$\begin{aligned} F_1(x_2,\ldots,x_{n+1}) &= 0, \\ F_2(x_2,\ldots,x_{n+1}) &= 0, \\ &\cdots \\ F_n(x_2,\ldots,x_{n+1}) &= 0, \\ F_1,\ldots,F_n &\in (\mathbb{C}[x_1])[x_2,\ldots,x_n,x_{n+1}], \end{aligned} \tag{5.59}$$

which we obtain from the system (5.58) by homogenizing it using a new variable x_{n+1}, i.e. $F_i = x_{n+1}^{d_i} f_i(\frac{x_2}{x_{n+1}},\ldots,\frac{x_n}{x_{n+1}})$. Like in (5.4) we set

$$d = \sum_{i=1}^{n}(d_i - 1) + 1 = \sum_{i=1}^{n} d_i - n + 1. \tag{5.60}$$

Consider the set of all monomials $\mathbf{x}^\alpha = x_2^{\alpha_2} x_3^{\alpha_3} \ldots x_n^{\alpha_n} x_{n+1}^{\alpha_{n+1}}$ in variables $x_2,\ldots,x_{n+1}$ of total degree d, i.e. $|\alpha| = \sum_{i=2}^{n+1}\alpha_i = d$, and partition it into n subsets:

$$\begin{aligned} S_1 &= \left\{\mathbf{x}^\alpha : |\alpha| = d,\ x_2^{d_1} \mid \mathbf{x}^\alpha\right\}, \\ S_2 &= \left\{\mathbf{x}^\alpha : |\alpha| = d,\ x_2^{d_1} \nmid \mathbf{x}^\alpha \text{ but } x_3^{d_2} \mid \mathbf{x}^\alpha\right\}, \\ &\cdots \\ S_n &= \left\{\mathbf{x}^\alpha : |\alpha| = d,\ x_2^{d_1},\ldots,x_n^{d_{n-1}} \nmid \mathbf{x}^\alpha \text{ but } x_{n+1}^{d_n} \mid \mathbf{x}^\alpha\right\}. \end{aligned} \tag{5.61}$$

Note that these sets are created like in the case (5.5); however, here the variable used as the last is the variable x_{n+1}, used in this case for the homogenization of equations (5.57).

Using sets S_i (5.61) we can create the extended system of polynomial equations

$$\begin{aligned} \mathbf{x}^\alpha / x_2^{d_1} F_1 &= 0 \text{ for all } \mathbf{x}^\alpha \in S_1, \\ &\cdots \\ \mathbf{x}^\alpha / x_{n+1}^{d_n} F_n &= 0 \text{ for all } \mathbf{x}^\alpha \in S_n. \end{aligned} \tag{5.62}$$

All polynomials $\mathbf{x}^\alpha / x_{i+1}^{d_i} F_i$ in this system are homogeneous polynomials of total degree d and therefore can be written as a linear combination of monomials of degree d in n variables $x_2,\ldots,x_{n+1}$. There exist $\binom{n+d-1}{d}$ monomials of degree d in n variables.

The total number of equations in the extended system (5.62) is equal to the number of elements in sets the $S_1,\ldots S_n$ (5.61), which is also $\binom{n+d-1}{d}$ because these sets contain all monomials of degree d in variables $x_2,\ldots x_{n+1}$. Thus the system of polynomial equations (5.62) consists of $\binom{n+d-1}{d}$ homogeneous equations in $s \leq \binom{n+d-1}{d}$ monomials (all of degree d) in variables $x_2,\ldots,x_{n+1}$. Note that the coefficients of these equations are polynomials in x_1 and that $s \leq \binom{n+d-1}{d}$ is because some coefficients may be zero and therefore some monomials may not appear in (5.62).

We can next dehomogenize equations (5.62) by setting $x_{n+1} = 1$. This dehomogenization doesn't reduce the number of monomials in these equations. This is because there are no monomials of the form $\mathbf{x}^\alpha = x_2^{\alpha_2} x_3^{\alpha_3} ... x_n^{\alpha_n} x_{n+1}^{\alpha_{n+1}}$ among monomials of total degree d which differ only in the power α_{n+1}.

Therefore, we obtain a system of $\binom{n+d-1}{d}$ equations in $s \leq \binom{n+d-1}{d}$ monomials up to degree d in $n-1$ variables $x_2, \ldots, x_n$. Note that the number of monomials up to degree d in $n-1$ variables is $\sum_{k=0}^{d} \binom{n+k-2}{k} = \binom{n+d-1}{d}$, which is exactly the number of monomials of degree d in n variables.

The system obtained after the dehomogenization is equivalent to the initial system (5.57), i.e. has the same solutions, and can be written as

$$\mathtt{C}\,(x_1)\,\mathbf{v} = 0, \tag{5.63}$$

where $\mathtt{C}\,(x_1)$ is a matrix polynomial and $\mathbf{v}$ is a vector of $s \leq \binom{n+d-1}{d}$ monomials in variables $x_2, \ldots, x_n$.

For many systems of polynomial equations we are able to choose $s \leq \binom{n+d-1}{d}$ linearly independent polynomials (including all initial polynomials (5.57)) from the generated $\binom{n+d-1}{d}$ polynomials (5.62). In that case the coefficient matrices $\mathtt{C}_j$ in the matrix polynomial $\mathtt{C}(x_1)$ are square, at least one is regular, and therefore the formulation (5.63) is directly a polynomial eigenvalue formulation of our system of polynomial equations (5.57).

Unfortunately, it may sometimes happen that between $\binom{n+d-1}{d}$ polynomials (5.62) there are less than s linearly independent polynomials or that, after rewriting these polynomials into the form (5.63), all coefficient matrices $\mathtt{C}_j$ are close to singular. This may happen because the presented Macaulay based method is not designed for general polynomials but for dense homogeneous ones, i.e. for polynomials whose coefficients are nonzero and generic, and it constructs a minimal set of equations that is necessary to obtain square matrices for these specific polynomials [39].

Therefore, we will use a small modification of this Macaulay based method. This modified method differs from the standard method in the form of the sets S_i and leads to a higher number of polynomial equations.

Modified Macaulay based method

We again consider all monomials $\mathbf{x}^\alpha = x_2^{\alpha_2} \ldots x_{n+1}^{\alpha_{n+1}}$ in variables $x_2, \ldots, x_{n+1}$ of total degree d; however, here the sets S_i have the form

$$\begin{aligned}
\overline{S}_1 &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_2^{d_1} \mid \mathbf{x}^\alpha \right\}, \\
\overline{S}_2 &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_3^{d_2} \mid \mathbf{x}^\alpha \right\}, \\
&\ldots \\
\overline{S}_n &= \left\{ \mathbf{x}^\alpha : |\alpha| = d,\, x_{n+1}^{d_n} \mid \mathbf{x}^\alpha \right\}.
\end{aligned} \tag{5.64}$$

This means that in the extended set of polynomial equations

$$\begin{aligned} \mathbf{x}^{\alpha}/x_2^{d_1}\, F_1 &= 0 \text{ for all } \mathbf{x}^{\alpha} \in \overline{S}_1, \\ &\dots \\ \mathbf{x}^{\alpha}/x_{n+1}^{d_n}\, F_n &= 0 \text{ for all } \mathbf{x}^{\alpha} \in \overline{S}_n, \end{aligned} \tag{5.65}$$

we multiply each homogeneous polynomial F_i of degree d_i, $i = 1, \dots, n$ by all monomials of degree $d - d_i$. This is because the set of monomials

$$\left\{ \mathbf{x}^{\alpha}/x_{i+1}^{d_i},\ x^{\alpha} \in \overline{S}_i \right\} \tag{5.66}$$

contains all monomials of degree $d - d_i$ in variables $x_2, \dots, x_{n+1}$. Therefore, the extended system of polynomial equations (5.65) consists of all possible homogeneous polynomials of degree d which can be obtained from the initial homogeneous polynomials F_i (5.59) by multiplying them by some monomials.

After creating this extended system of polynomial equations, the method continues as the previously described standard Macaulay based method, i.e. we dehomogenize all equations from (5.65) and rewrite them to the form

$$\overline{\mathtt{C}\,(x_1)}\overline{\mathbf{v}} = \mathbf{0}, \tag{5.67}$$

where $\overline{\mathtt{C}\,(x_1)}$ is a matrix polynomial and $\overline{\mathbf{v}}$ is a vector of $\overline{s} \leq \binom{n+d-1}{d}$ monomials up to degree d in variables $x_2, \dots, x_n$.

Since in this case we generate more polynomial equations, in fact all possible polynomial equations of degree d that can be obtained by only multiplication by monomials, it is more frequent that we are able to choose $\overline{s} \leq \binom{n+d-1}{d}$ linearly independent polynomials from them. Moreover, when we have more than $\overline{s}$ linearly independent polynomials, we can select a subset of polynomials to obtain well conditioned coefficient matrices $\mathtt{C}_j$ and we can in this way improve numerical stability of the polynomial eigenvalue formulation (5.67).

In the presented approach we were considering system (5.57) of n equations in n unknowns. However, this method can be easily extended to a system (5.55) of $m \geq n$ equations in n unknowns.

In the case of the Modified Macaulay based method it is sufficient to multiply each homogeneous polynomial $F_i,\ i = 1, \dots, m$ of degree d_i, by all monomials of degree $d - d_i$, which is independent on the number of polynomials.

For the standard Macaulay based method all what we need to do is to select n equations from the initial m equations with largest degrees d_i. Then, we can apply the presented method to these n equations in n unknowns. In this way we will generate $\binom{n+d-1}{d}$ equations in $s \leq \binom{n+d-1}{d}$ monomials up to degree d in $n-1$ variables $x_2, \dots, x_n$ and with polynomial coefficients in x_1. Adding the remaining $m - n$ original equations may increase the number of monomials. It is because some monomials contained in these equation do not have be contained in the generated and the selected equations. However, this number will not be greater then $\binom{n+d-1}{d}$ because the degrees of these $m - n$ equations are smaller or equal to the degrees of the selected equations

and therefore also smaller than d. Moreover, we can also add all multiples of the remaining $m - n$ equations up to degree d. In both cases, the resulting system of equations will contain only monomials up to degree d in variables $x_2, \ldots, x_n$ and there are at most $\binom{n+d-1}{d}$ such monomials.

Problem relaxation

The presented Modified Macaulay based method results in the polynomial eigenvalue formulations (5.56) for most systems of polynomial equations. However, note that this polynomial eigenvalue formulation (5.56) is a relaxed formulation of the original problem of finding all solutions of the system of polynomial equations (5.55). This is because polynomial eigenvalue formulation (5.56) does not consider potential dependencies in the monomial vector $\mathbf{v}$ and solves for general eigenvalues x_1 and general eigenvectors $\mathbf{v}$. However, after transforming the system of polynomial equations (5.55) to PEP (5.56), coordinates of the vector $\mathbf{v}$ are in general dependent, e.g., $\mathbf{v} = \left(x_3^2, x_2x_3, x_2^2, x_3, x_2, 1\right)^\top$ and therefore $\mathbf{v}(1) = \mathbf{v}(4)^2$, $\mathbf{v}(2) = \mathbf{v}(4)\mathbf{v}(5)$, etc..

This implies that after solving PEP (5.56) one has to check which of the computed eigenpairs $(x_1, \mathbf{v})$ satisfy the original polynomial equations (5.55). This can be done either by testing all monomial dependencies in $\mathbf{v}$ or by substituting the solutions to the original equations and checking if they are satisfied.

Reducing the size of the polynomial eigenvalue problem

The second step of the offline phase of the specialized resultant based method tries to reduce the size of the found polynomial eigenvalue formulation (5.56) of the input system of polynomial equations (5.55).

Removing unnecessary polynomials

Let's assume that the proposed Modified Macaulay based method results in the polynomial eigenvalue formulation (5.67) of the input system of polynomial equations (5.55).

This Modified Macaulay based method usually does not lead to the smallest polynomial eigenvalue formulation of the initial system of polynomial equations (5.55) and, for larger systems with larger degrees, it may generate large polynomial eigenvalue problems which are not practical. Therefore, we present here a method for removing unnecessary polynomials from the generated polynomial eigenvalue formulation.

Assume that we have the polynomial eigenvalue formulation (5.67) of the system (5.55) of m polynomial equations in n unknowns $x_1, \ldots, x_n$, and that the vector $\overline{\mathbf{v}}$ in this polynomial eigenvalue formulation (5.67) consist of $\overline{s}$ monomials and the coefficient matrices $\overline{\mathtt{C}}_j$ in the matrix polynomial $\overline{\mathtt{C}(\mathtt{x}_1)}$ have size $\overline{s} \times \overline{s}$. Then, we can remove unnecessary polynomials using the following procedure:

Algorithm 12 Removing polynomials in PEP

$i := 1$
while $i \leq \overline{s} - m$ **do**
  if there exist i monomials $\mathbf{x}^{\alpha}$, among monomials of the vector $\overline{v}$, that are contained only in $k \leq i$ polynomials of (5.67) from our $\overline{s}$ polynomials, and these k polynomials doesn't contain any initial polynomial **then**
    remove these k polynomials
    $\overline{s} := \overline{s} - k$
    $i := 1$
  else
    $i := i + 1$
  end if
end while

In each cycle of Algorithm 12 we remove more or as many polynomials as monomials from (5.67), or we remove nothing. Therefore, after each cycle the coefficient matrices $\overline{\mathtt{C}}_j$ are square or contain more rows than columns and can be therefore transformed to square matrices. Moreover, there remain at least initial polynomial equations at the end of this algorithm. Therefore, using this procedure, we obtain a polynomial eigenvalue formulation of the initial system of polynomial equations (5.55) which is smaller or of the same size as the starting polynomial eigenvalue formulation (5.67).

This removing algorithm 12 may have larger impact when it is performed on an eliminated system of polynomial equations, i.e. a system of polynomial equations eliminated by G-J elimination. For example, G-J elimination may help when the system contains monomials that after "hiding" a variable in the coefficient field, e.g., x_1, have only numerical (constant) coefficients, i.e. coefficients not containing x_1. Such monomials may then be directly eliminated, not depending on in how many polynomials they appear. This is because the application of G-J elimination on all polynomials with suitable ordered monomials, before "hiding" x_1, may eliminate these monomials from all equations except for one equation. Therefore, such monomials will be contained only in one polynomial and will be eliminated using the presented algorithm.

Removing zero eigenvalues

It often happens that matrices $\mathtt{C}_j$ in the polynomial eigenvalue formulation (5.56) contain zero columns. A zero column in a matrix $\mathtt{C}_i$ means that the monomial corresponding to this column and to x_1^i does not appear in the considered system of equations. Moreover, it also happens quite often that this monomial does not appear in the system for all x_1^j, where $j > i$. Then, the column corresponding to this monomial is zero in all matrices $\mathtt{C}_j$ for $j > i$ including the highest order matrix $\mathtt{C}_l$, which is in this case singular. We will call such monomials, which produce zero columns in matrices $\mathtt{C}_j,\ j = i, \ldots, l$, "zero" monomials.

In the case of singular matrix $\mathtt{C}_l$ and regular matrix $\mathtt{C}_0$, the transformation of the PEP to the eigenvalue problem (5.52) will produce matrix $\mathtt{A}$ in the form (5.54). This matrix will contain for each such a "zero" monomial a zero column, a column in the block corresponding to $-\mathtt{C}_0^{-1}\mathtt{C}_l$.

This zero column will result in a zero eigenvalue which is, in this case, "parasitic" and is not the solution to our original problem. Therefore, we can remove this column together with the corresponding row from the matrix $\mathtt{A}$ (5.54) and in this way we can remove this zero eigenvalue.

Removing the row will remove 1 from the column corresponding to the same monomial as the "zero" monomial in the $-\mathtt{C}_0^{-1}\mathtt{C}_l$ block, but in this case in the block $-\mathtt{C}_0^{-1}\mathtt{C}_{l-1}$. This means that if this "zero" monomial multiplied by x_1^{l-1} does not appear in the considered system of polynomial equations we have again a zero column resulting in zero eigenvalue. Therefore, we can again remove this zero column and the corresponding row of the matrix $\mathtt{A}$. This can be repeated until the first matrix $\mathtt{C}_j$ which contains, for this "zero" monomial, a non-zero column.

In this way, we can often remove most of the "parasitic" zero eigenvalues and therefore solve a considerably smaller eigenvalue problem which may significantly improve the computational efficiency and the stability of the solution. This is because the eigenvalue computation is usually the most time-consuming part of final solvers.

Note that the reduction of the size of the polynomial eigenvalue problem (5.55) needs to be performed for a particular problem only once in the offline phase. The final online solver will use only polynomials which remain after this reduction step and will therefore solve the "reduced" polynomial eigenvalue problem with removed "zero columns" .

Since in this offline phase of the specialized resultant based method we do not perform any operations that can be numerically unstable we do not need to perform computations in a finite prime field, although it can be done.

The offline phase can be performed for all variables $x_j,\ j = 1, \ldots, n$, i.e. in this phase we can try to "hide" all variables, and then select the variable that results to the smallest polynomial eigenvalue formulation.

5.5.2 Online phase

In the online phase, an efficient specific polynomial eigenvalue solver for solving all systems of polynomial equations in the form of the input system F (5.55), see Definition 20, is created and used to solve all these systems.

The final online specific polynomial eigenvalue solver starts with the input polynomial equations (5.55) with concrete coefficients from $\mathbb{C}$. Then, according to the found "reduced" polynomial eigenvalue formulation, the solver multiplies the input polynomial equations with necessary monomials and in this way creates an extended system which can be formulated as PEP (5.44).

Then the solver solves this PEP (5.44) by transforming it to the eigenvalue problem

$$\mathtt{A}\mathbf{y} = \lambda\mathbf{y}. \tag{5.68}$$

This is done as described in Section 5.4.1. When it is necessary, i.e. when the coefficient matrix $\mathtt{C}_l$ in the matrix polynomial (5.45) is singular, then the linear transformation $\frac{1}{x_j}$, where x_j is the hidden variable, is used. From the final matrix $\mathtt{A}$ in the eigenvalue formulation (5.68) the solver removes "zero" columns as found in the offline phase.

Finally, the solver uses a standard eigenvalue method [9] to find eigenvalues and eigenvectors of (5.68) and in this way solves the input polynomial equations.

5.5.3 Modifications of specialized resultant based method

We conclude this section with a discussion of some modifications of the presented specialized resultant based method for solving systems of polynomial equations, especially its offline phase, where we transform a system of polynomial equations to PEP (5.44).

The specialized resultant based method, which was presented in this section and which uses hidden variable resultants and polynomial eigenvalue problems, doesn't work for all systems of polynomial equations. It is because the Modified Macaulay based method for transforming a system of polynomial equations to PEP (5.44) doesn't result for all systems in their polynomial eigenvalue formulations. However, this Modified Macaulay based method is very simple, intuitive an we found it working for most of the problems discussed later in the text.

In this subsection we present some alternative methods that can be used to transform the initial system of polynomial equations to PEP (5.44).

The first method is only a small modification of the Modified Macaulay based method presented in Section 5.5.1. As we have already mentioned, the Modified Macaulay based method from Section 5.5.1 may sometimes generate less linearly independent polynomial equations than monomials in these equations (after hiding one variable in the coefficient field). However, this may not always be a problem. It is because it may happen that between these polynomial equations there exist i monomials which are contained only in $j < i$ polynomials. Therefore after removing these j polynomials we will remove less polynomials than monomials. If we are lucky then after removing them we will obtain as many or more polynomials than monomials contained in these polynomials, and therefore we can formulate these equations as PEP (5.44).

This removing of polynomials is exactly what is done in Algorithm 12 used in the second step of the presented offline phase. Therefore, sometimes, even if we do not obtain a polynomial eigenvalue formulation in the first step of the offline phase, we can continue with the second step, i.e. with the removing of unnecessary polynomials, and we can still obtain PEP (5.44).

If we still do not obtain a polynomial eigenvalue formulation of the initial system of polynomial equations we can increase the degree d of generated polynomials and try if this helps.

We end this section with a method for transforming a system of polynomial equations to PEP (5.56) that is somewhat more complicated than the , but that is general and works for all systems of polynomial equations.

In the Modified Macaulay based method from Section 5.5.1 we generate new polynomials only by multiplying initial polynomials by monomials of some degree. In this way we are not able to generate all polynomials from the ideal.

Therefore, to generate polynomials from the ideal, we can use a method which is similar to the one used to generate polynomials in the specialized Gröbner basis method presented in Section 4.4.3. In Section 4.4.3 we have presented two different strategies for generating polynomials from the ideal, i.e. the single elimination and the multiple elimination strategy. For the purpose of the specialized resultant based method it is better to use the multiple elimination strategy. It is because using multiple elimination strategy, the number of generated monomials grows slower than in the case of a single G-J elimination and that is what we want.

The multiple elimination strategy starts with the initial polynomials (5.55) and then systematically generates new polynomials from the ideal by multiplying all new generated polynomials

of degree $< d$ by all individual variables and reducing them each time by G-J elimination. After each G-J elimination, we can check if we already have a polynomial eigenvalue formulation. If no new polynomials of degree $< d$ were generated and no polynomial eigenvalue formulation was obtained, then we increase the degree d.

By checking if we already have a polynomial eigenvalue formulation we mean checking if there exists i polynomials containing $j \leq i$ monomials between already generated polynomials. We can do this either by the removing procedure presented in the previous section 5.5.1 or directly by trying all subsets of the generated polynomials. This verification is not expensive, since it needs to be done only once in the offline phase, and we do not need to know coefficients of the generated polynomials.

6 Automatic generator of minimal problems solvers

"Everything should be made as simple as possible, but not simpler."

Albert Einstein

6.1 Introduction

Algebraic solvers based on Gröbner bases have been used recently to solve many computer vision problems [134, 132, 135, 136]. These Gröbner basis solvers were mostly designed for particular problems and in general consisted of three key steps.

In the first step of these solvers [134, 132, 135, 136], the particular problem was solved using a computer algebra system, e.g. Macaulay 2 or Maple, for several usually random coefficients from a finite field of the input system F representing considered instances of this particular problem. This was done using a general technique for solving systems of polynomial equations by finding a Gröbner basis of the ideal generated by the initial polynomials F [39]. In this phase, the basic parameters of the considered instances of the problem were also identified, such as how many solutions the problem has, and how "hard" is it to obtain the Gröbner basis, and which leading monomials it contains. This procedure was usually automatic and relied on general algorithms of algebraic geometry such as Buchberger's algorithm [18] or F4 algorithm [49].

In the second step, a special elimination procedure, a special "elimination template", similar to an admissible Elimination trace presented in Section 4.4.3, was in all these solvers designed. This elimination procedure determined which polynomials from the ideal should be added to the initial polynomial equations and how they should be eliminated to obtain a complete Gröbner basis or at least all polynomials needed for constructing a multiplication matrix and thus solving the initial equations.

The goal of this step was to obtain a computationally efficient and numerically robust procedure for solving all systems in the form of the "representing system" F. Until now, this step has been mainly manual, requiring to trace a path toward a Gröbner basis for some coefficient from a finite field, checking redundancies and possible numerical pitfalls, and writing down a program in a procedural language such as Matlab or C.

In the last step, an efficient solver for solving all considered instances of the particular problem, i.e. all systems of polynomial equations in the form of the input system, was created according to the constructed "elimination template". The template was used to create a multiplication matrix (4.44). The solutions to the original problem were obtained numerically as eigenvalues or eigenvectors of this matrix.

The first and the third step are standard and well understood and are in fact very similar in all these solvers including all our solvers presented in this thesis.

It is the second step which involves considerable amount of craft and which makes the process of solver generating rather complex and virtually impenetrable for a non-specialist. Moreover, for some problems it isn't clear how their "elimination templates" were generated and therefore non-specialists often use them as black boxes, they often are not able to reimplement them, improve them, or create similar solvers for their own new problems.

On the other hand, the method for generating admissible Elimination traces presented in Section 4.4.3 can be easily automated and used also by non-specialists and without any knowledge of algebraic geometry. Therefore, in this chapter we present an automatic generator of admissible Elimination traces for Gröbner basis solvers, or in fact a generator of efficient specific Gröbner basis solvers for systems of polynomial equations in the form of some input system F.

We have to accept that there is no hope to obtain an efficient and robust solver for completely general systems of polynomial equations since the general problem is known to be EXPSPACE complete [82]. On the other hand, many camera geometry problems share the property that the Elimination trace associated with their solution is the same for all interesting configurations of coefficients of the particular problem, i.e. all interesting instances of the problem are in the form of one "representing system" F, see Definition 20. Moreover, this Elimination trace is usually quite simple and can be efficiently implemented.

Consider, for instance, the classical 5-point relative pose problem, i.e. the problem of estimating the essential matrix from five image point correspondences [112, 132]. In general, an admissible Elimination trace used to solve this 5-point problem depends on actual image coordinates measured. Fortunately, for "non-degenerate" image correspondences, i.e. for those which are not collinear or coincide and which lead to a finite number of essential matrices, the Elimination trace is always the same, see Example 21. Therefore, it is enough to find an admissible Elimination trace for one particular "non-degenerate" configuration of coefficients and then use the same trace for all "non-degenerate" configurations of coefficients, i.e. all "non-degenerate" instances of the 5-point problem. Elimination traces for "degenerate" configurations of coefficients may be very different and there may be very many of them. However, in practice we do not need to consider them.

In this chapter we propose an automatic generator that is based on the specialized Gröbner basis method presented in Section 4.4 and that searches for an admissible Elimination trace for a given system of polynomial equations F. Using this admissible Elimination trace the automatic generator produces an efficient solver for all instances of the input problem that would lead to the same Elimination trace, i.e. all systems of polynomial equations in the form of the input system F. There is a number of valid traces, i.e. valid paths, the generator can find. The choice of the particular trace, is determined by the particular coefficients of F.

The input to our automatic generator is a system of polynomial equations F with concrete coefficients from $\mathbb{Z}_p$. A particular choice of coefficients determines the Elimination trace and the set of "consistent" instances of F for which this trace is admissible, see Definition 21. For many problems, the interesting "regular" solutions can be obtained with almost any, i.e. random, choice of coefficients of the "representing system" F. Therefore, we use random values from $\mathbb{Z}_p$ as default coefficients.

The output of the automatic generator is the Matlab or Maple code that returns solutions of all systems of polynomial equations in the form of the input system F, with concrete coefficients from $\mathbb{Q}$. During the online computations, only the resulting solver is called.

Next we describe all steps of the automatic generator in more detail and in Chapter 7 we demonstrate that this automatic generator constructs efficient and numerically stable solvers that are comparable or better than known manually constructed solvers in terms of computational time, space requirements, and numerical stability.

6.2 The automatic procedure for generatig Gröbner basis solvers

In this section we describe our approach to automatic generation of specific Gröbner basis solvers for systems of polynomial equations in the form of a given system F. These solvers follow the specialized method for solving systems of polynomial equations based on Gröbner bases presented in Section 4.4 and summarized in Section 4.4.5.

The input to the automatic generator is a system of polynomial equations in n unknowns and with symbolic coefficients. These symbolic coefficients can be products of several symbolic variables. The list of these "known" symbolic variables and the list of n unknowns are also inputs to the automatic generator.

The output of the automatic generator is the solver, i.e. the MATLAB or Maple code, which returns solutions of all system of polynomial equations in the form of the input system F with concrete coefficients from $\mathbb{Q}$.

An example of the input to our automatic generator for the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola in $\mathbb{C}^2$, presented in Example 19, can be found in Figure 6.1.

In this case we have two equations $eq(1)$ and $eq(2)$ representing an ellipse and a hyperbola, the list of known variables is the list of coefficients $a0, \ldots, a4, b0, \ldots, b4$ and the list of unknown variables contains x and y. These are inputs to the automatic generator, which is called using the function $[\texttt{res export}] = \texttt{gbs\_CreateCode}(\texttt{'example'}, \texttt{eq}, \texttt{known}, \texttt{unknown})$. Here the first parameter $\texttt{'example'}$ is the name of the final online solver.

The automatic generator consists of several modules, see Figure 6.2. Since all these modules are independent, they can be further improved or replaced by more efficient implementations. Next we briefly describe each of these modules.

6.2.1 Polynomial equations parser

First, the input equations are split into coefficients and monomials. If no constraints on known variables occurring in coefficients are given, the generator will instantiate each known variable with a random number from $\mathbb{Z}_p$ by default.

For the problem of the intersection of axis-aligned ellipse and an axis-aligned hyperbola from Figure 6.1, this means, that the known variables $a0, \ldots, a4, b0, \ldots, b4$, which in this case correspond directly to the coefficients of the two input equation, are replaced with some random

```
% symbolic variables
syms a0 a1 a2 a3 a4 b0 b1 b2 b3 b4;

syms x y;

% two equations representing an ellipse and a hyperbola
eq(1) = a0*x^2 + a1*x + a2*y^2 + a3*y +a4;
eq(2) = b0*x^2 + b1*x - b2*y^2 + b3*y +b4;

% known parameters
known = {'a0','a1','a2','a3','a4','b0','b1','b2','b3','b4'}

%two unknowns
unknown = {'x','y'}

% call code generator
[res export] = gbs_CreateCode('example', eq, known, unknown);
```

Figure 6.1: Input Matlab script for the automatic generator of Gröbner basis solvers generating the solver for the intersection of an axis-aligned ellipse and an axis-aligned hyperbola.

numbers from $\mathbb{Z}_p$. In the case of this problem instantiating with random numbers from $\mathbb{Z}_p$ is sufficient.

The automatic generator also assign a unique identifier to each coefficient used. This is important for the code generation module, to be able to track the Elimination trace.

In our automatic generator, it is also possible to instantiate coefficients with concrete numbers from $\mathbb{Z}_p$. This is useful when one wants to solve a special instance of the problem, for example some "degenerate" instance with some zero coefficients or some coefficients depending on other coefficients. More about exact instantiating can be found in [24].

After the instantiation, the next module which corresponds to the first step of the offline phase presented in Section 4.4.3, i.e. to the step of identification of basic parameters, starts.

6.2.2 Computation of the basis B and the number of solutions

This module starts with the system of polynomial equations F with coefficients from $\mathbb{Z}_p$. The generator first verifies if the input system of equations F has a finite number of solution, i.e. if the initial polynomial equations generate a zero-dimensional ideal, and how many solutions there are. If the system has a finite number of solutions, the automatic generator computes the Gröbner basis G w.r.t. the graded reverse lexicographic monomial ordering and the basis B (4.20) of the quotient ring A (4.12). The output of this part of the generator is the basis B of the quotient ring A.

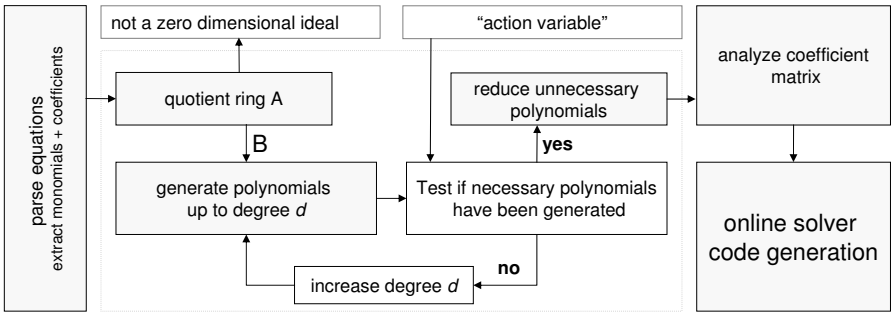

Figure 6.2: Block diagram of the off-line solver.

To obtain this information, we use either Macaulay 2 or Maple, which are able to compute in finite prime fields and which provide all functions that we need for our purpose. Moreover, these functions can be called directly from MATLAB and their output can be further used in the generator. Modularity of the generator allows replacing this part of the code by another existing module computing the Gröbner basis G and the basis B.

For the example of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola from Figure 6.1 the automatic generator returns these intermediate results:

```
Creating random instance in Z_30097 for 2 equations
...instancing equations
calculating Grobner basis using Macaulay2
...algebra B basis :
   1 x x*y y
...system is expected to have 4 solutions
analyzing quotient ring basis
extracting coefficient & monomials
...used 5 monomials :
   x^2 y^2 x^1 y^1
```

This says that our two equations (4.102) have four solutions and that the monomial basis B (4.21) of the quotient ring $A = \mathbb{C}[x, y]/I$ has the form $B = \{[xy], [x], [y], [1]\}$ for random coefficients and for the default grevlex monomial ordering.

In this case Macaulay 2 [59] was used to obtain this information. Figure 6.3 shows the input to Macaulay 2, for the intersection of an axis-aligned ellipse and an axis-aligned hyperbola, which was automatically generated using our automatic generator. With this input, Macaulay 2 [59] computes the number of solutions, the Gröbner basis and the basis B (4.20) of the quotient ring $A = \mathbb{Z}_p[x_1, x_2]/I$ for the ideal I generated by the two input equations with random coefficients from $\mathbb{Z}_p$. Here $x_1 := x$ and $x_2 := y$.

6.2.3 Single elimination trace construction

The input to this third, the most important, module of our automatic generator is the basis B of the quotient ring A and the polynomial f for which we want to create the multiplication

```
-- Macaulay2 code template for gbsPolySolver

KK = ZZ/30097

R = KK[x_1..x_2, MonomialOrder=>GRevLex]

-- generated part
f1=18242*x_1^2+23211*x_1+9870*x_2^2+3306*x_2+19092;
f2=19732*x_1^2+28665*x_1+28311*x_2^2+27305*x_2+9428;

f = (f1 || f2 );
-- generated part end

I1 = ideal(f);
gbTrace 3;
dm = dim I1;
dg = degree I1;

-- Do not modify following lines
print "dim:"
print dm;
print "deg:"
print dg;

A = R/I1;
Ab = basis A;

print "@@@@@@@@@@@@@@@"
print Ab;
print "@@@@@@@@@@@@@@@"
-- end

quit
```

Figure 6.3: Input to Macaulay 2 [59] automatically generated using our automatic generator of Gröbner basis solvers for the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola.

matrix. In our automatic generator we by default try all individual variables, i.e. $f = x_k$ for $k = 1, \ldots, n$, called "action variables", to create the multiplication matrices $\mathtt{M}_{x_k}$. For the final solver we select the action variable x_k that results in the "simplest" multiplication matrix $\mathtt{M}_{x_k}$.

The goal of this module of the automatic generator is to generate all necessary polynomials for constructing the multiplication matrix $\mathtt{M}_{x_k}$, i.e. to find an admissible Elimination trace for constructing this matrix. This module corresponds to the second step of the offline phase of the specialized Gröbner basis method presented in Section 4.4.3, i.e. to the step of the construction of the multiplication matrix.

The method described in this Section 4.4.3 calls for generating polynomials from the ideal I with the leading monomials $x_k\mathbf{x}^{\alpha(i)}$ and the remaining monomials corresponding to the monomial cosets from B, i.e. polynomials q_i (4.115) of the form $q_i = x_k\mathbf{x}^{\alpha(i)} - \sum_{k=1}^{N} c_k^i \mathbf{x}^{\alpha(k)} \in I$, where $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ and $c_k^i \in \mathbb{C}$.

As explained in Section 4.4.3, there are several ways how to generate these polynomials from the initial polynomials F. In our automatic generator we have decided to implement the strategy of single G-J elimination presented in Section 4.4.3, i.e. Algorithm 9. This means that we generate necessary polynomials in one step by multiplying polynomials from F by selected monomials and reducing all generated polynomials at once using a single G-J elimination of a single coefficient matrix.

The monomial multiples of polynomials F, which should be added to the initial polynomials to obtain all necessary polynomials q_i, are found in this part of the automatic generator. This is done by systematically generating polynomials from the ideal I and testing them by checking if all polynomials q_i are already generated. We stop when all necessary polynomials q_i are obtained. The generator tries to add polynomials starting with the polynomials with as low degree as possible. Thus, it first multiplies input polynomials with the lowest degree monomials and then moves to the higher degree monomials.

This procedure of finding a "path" from F to necessary polynomials q_i, i.e. finding an admissible Elimination trace for F is exactly the procedure presented in Section 4.4.3 in Algorithm 9. Alternatively, the second strategy for constructing an admissible Elimination trace based on multiple eliminations, i.e. Algorithm 8, or another different strategy for generating polynomials from the ideal can be used here.

In our example of the intersection of an ellipse and a hyperbola from Figure 6.1, the automatic generator generates monomial multiples of the two initial polynomial equations up to degree three and returns the following:

```
adding polynomials
...initializing matrix size 2,6
...reducing matrix with size 2x6
...removing equation 2 failed +
...coefficient matrix reallocaction, adding 4(3 degree) monomials
...reducing matrix with size 6x10
all required polynomials for action variable 'x' generated,
```

This means that the process of generating polynomials ends with 6 polynomials in 10 mono-

mials and that after the G-J elimination of their coefficient matrix (4.24) all polynomials q_i necessary for constructing the multiplication matrix $\mathtt{M}_x$ are obtained.

6.2.4 Reducing the size of the trace

This part of the automatic generator starts with the polynomials generated in the previous step. We know that after the G-J elimination of these polynomials, i.e. after the G-J elimination of the corresponding matrix in the matrix representation (4.24) of these polynomials, we obtain all polynomials that we need for construction of the multiplication matrix $\mathtt{M}_{x_k}$.

However, since the method used for generating polynomials from the ideal generates polynomials systematically, it generates also many unnecessary polynomials, i.e. polynomials which are not necessary for creating polynomials q_i (4.115). Therefore, the automatic generator reduces these unnecessary polynomials in this step. This is done using Algorithm 10 described in Section 4.4.3.

The simple removal Algorithm 10 removes polynomials one by one and each time calls expensive G-J elimination. Its complexity is $O(n^4)$ in the number n of polynomials used. This algorithm is not efficient but it is not crucial since it is called only once for the particular problem in the offline phase of creating the solver.

In the automatic generator we enhanced this removal algorithm in several ways: (1) we use sparse G-J elimination since an Elimination trace usually results in a quite sparse matrix, (2) we remove more than one polynomial at once with the heuristic - if we succeeded to remove k polynomials, then we remove $2\,k$ polynomials in the next step. If we failed to remove k polynomials, then we try to remove only $\frac{1}{4}\,k$ polynomials in the next step. These two improvements considerably speed up the reduction process. Moreover, we can employ the fact that the polynomials in the Elimination trace are ordered according to their leading monomials. In G-J elimination of such ordered polynomials, we can exploit results from previous G-J eliminations.

In the next step the automatic generator removes also "unnecessary monomials", i.e. monomials which do not affect the necessary polynomials q_i and therefore also the resulting multiplication matrix.

Polynomials which remain after this removing procedure form the final Elimination trace. Let's denote these polynomials by H and the matrix in their matrix representation (4.24) by $\mathtt{H} = \Psi\,(H)$.

Note, that in this matrix representation there are only columns that correspond to monomials that remained after removing all "unnecessary monomials". This means that if s is the number of polynomials in $\mathtt{H}$ and N is the number of solutions of the system, then the matrix $\mathtt{H}$, which needs to be eliminated in the final step of the Elimination trace has the dimension $s \times (s + N)$.

The automatic generator removes two polynomials from the six generated polynomials for our example from Figure 6.1 in the following way:

```
removing unnecessary
...removing equation 3 - 6 failed *
...removing equation 6 succeeded
...removing equation 4 - 5 failed *
...removing equation 5 failed *
```

```
...removing equation 4 succeeded
...removing equation 2 - 3 failed *
...removing equation 3 failed *
...removing equation 2 failed *
...removing equation 1 failed *
generated 4 polynomials in 10 monomials (8 nonzero)
extracting the action matrix for variable 'x'
(used coef. matrix size 4x8)
--- generating matlab solver ---
```

Note that in this case the final Elimination trace contains one more polynomial in comparison with Example 25. It is because in Example 25 we start with already eliminated initial polynomials (4.134) and (4.135) and we in fact perform two G-J eliminations.

The automatic generator will return for the eliminated polynomials (4.134) and (4.135) the same Elimination trace as described in Example 25.

6.2.5 Construction of the multiplication matrix

To create a "template" for the multiplication matrix $\mathtt{M}_{x_k}$, we identify those rows of the matrix $\widetilde{\mathtt{H}}$ that correspond to the polynomials q_i (4.115). Here $\widetilde{\mathtt{H}}$ is the matrix representing polynomials from the final Elimination trace $\mathtt{H}$ after the G-J elimination. The multiplication matrix $\mathtt{M}_{x_k}$ will then contain coefficients from the identified rows and from the columns of $\widetilde{\mathtt{H}}$ that correspond to the monomials corresponding to the monomial cosets from the basis B.

This means that for the online solver we just need to know which rows and columns we need to extract. Note that the structure of the eliminated matrix $\widetilde{\mathtt{H}}$ is always the same for all systems of polynomial equations which are in the form of the input system F. Therefore, the rows and the columns that need to be extracted are always the same.

6.2.6 Generating efficient online solver

The online solver consists of the following steps:

1. Construction of the coefficient matrix $\mathtt{H} = \Psi(H)$, i.e. the matrix representation (4.24) of the polynomials H from the final Elimination trace;
2. G-J elimination of the coefficient matrix $\mathtt{H}$;
3. Construction of the multiplication matrix $\mathtt{M}_{x_k}$ from the matrix $\widetilde{\mathtt{H}}$, i.e. from the eliminated matrix $\mathtt{H}$;
4. Extraction of the solutions of the input system of polynomial equations from eigenvectors of the multiplication matrix $\mathtt{M}_{x_k}$.

In order to be able to create a source code of the online solver, we use unique identifiers associated with coefficients of the input polynomials. Hence, besides the coefficient matrix $\mathtt{H} = \Psi(H)$ in $\mathbb{Z}_p$, we also maintain a matrix with coefficient identifiers, i.e. an index matrix

$\mathtt{H}_{index}$. Then we apply all operations performed on actual coefficients of $\mathtt{H}$ in $\mathbb{Z}_p$, i.e. adding and removing rows and linear combination of rows, also on the index matrix $\mathtt{H}_{index}$.

Recall that in the construction of the Elimination trace we use the input equations and multiply them by monomials. This is nothing else but shifting identifiers associated to the input polynomials in the index matrix $\mathtt{H}_{index}$.

Removing unnecessary polynomials and unnecessary monomials from H corresponds to removing rows and columns from the index matrix $\mathtt{H}_{index}$. Hence, the code generator creates a code that simply puts coefficients of the input polynomials to the correct places using the identifiers of coefficients. Then, after G-J elimination, it reads values form the precomupted rows and columns and builds the final multiplication matrix.

Figure 6.4 shows an example of the final online solver generated using the presented automatic generator for the input from Figure 6.1. This online solver solves all "non-degenerate" instances of the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola.

```
function [x y] = example(a0, a1, a2, a3, a4, b0, b1, b2, b3, b4)

% precalculate polynomial equations coefficients
M = zeros(4, 8);

M([3, 9]) = a0;
M([7, 17]) = a2;
M([15, 21]) = a1;
M([19, 25]) = a3;
M([27, 29]) = a4;
M([4, 10]) = b0;
M([8, 18]) = -b2;
M([16, 22]) = b1;
M([20, 26]) = b3;
M([28, 30]) = b4;

Mr = rref(M);  % replace me with a MEX

A = zeros(4);
amcols = [8 7 6 4];
A(1, 3) = 1;
A(2, 4) = 1;
A(3, :) = -Mr(3, amcols);
A(4, :) = -Mr(1, amcols);

[V D] = eig(A);
sol =  V([3, 2],:)./(ones(2, 1)*V(1,:));

if (find(isnan(sol(:))) > 0)
x = [];
y = [];
else
I = find(not(imag( sol(1,:) )));
x = sol(1,I);
y = sol(2,I);
end
end
```

Figure 6.4: Final online Matlab solver, generated using the presented automatic generator, for the problem of the intersection of an axis-aligned ellipse and an axis-aligned hyperbola from Figure 6.1.

7 Minimal problems

"The mere formulation of a problem is often far more essential than its solution, which may be merely a matter of mathematical or experimental skills. To raise new questions, new possibilities, to regard old problems from a new angle requires creative imagination and marks real advances in science."

Albert Einstein

Many problems in computer vision can be formulated using systems of polynomial equations. For these problems the number of equations in the system depends on the number of geometric constraints and the number of input data (usually point or line correspondences) used to formulate the problem. The computer vision problems solved from a minimal number of input data and using all possible geometric constrains are often called "minimal problems" or "minimal cases".

It is well known that a smaller number of correspondences less likely contains incorrect matches and therefore considerably reduces the number of samples in RANSAC [50] algorithms, which are widely used in many applications. This is the motivation for studying the minimal number of correspondences necessary to solve a given problem and for developing algorithms solving these minimal cases. Another motivation for studying minimal cases is the fact that the minimal case solutions give us full information about the given problem, since all available constraints are used.

In this section we will discuss minimal cases for several problems in computer vision and we will give solutions to them using the methods for solving systems of polynomial equations presented in Chapters 4 and 5.

7.1 Relative pose problems

Relative pose problems are problems of estimating relative camera position and orientation and sometimes also internal calibration parameters of the camera. Usually image point correspondences, i.e. image points $\mathbf{x}$ and $\mathbf{x}'$ corresponding to the same point $\mathbf{X}$ in the space, as shown in Figure 7.1, are used for this purpose.

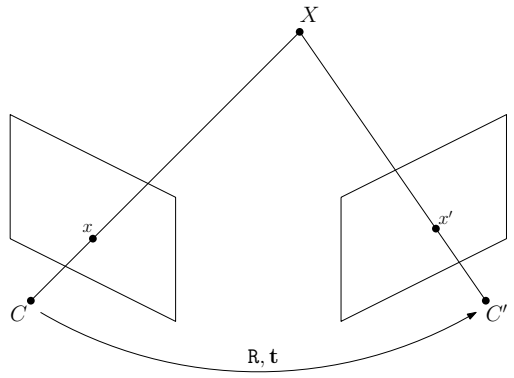

Figure 7.1: Epipolar geometry.

7.1.1 Introduction

Estimating camera motion and internal calibration parameters from sequences of images is an important problem with a broad range of applications [61]. It is one of the oldest computer vision problems and even though much has already been solved some questions remain still open.

It is known that for estimating relative pose of two fully-calibrated cameras, the minimal number of image points is five and for two cameras with unknown equal focal length it is six. Therefore, these two minimal problems are often called the *five-point relative pose problem* (5-pt problem) and the *six-point focal length problem* (6-pt problem).

Efficient algorithms for solving these two important minimal problems appeared only recently [112, 132, 134]. It was caused by the fact that these problems result in relatively complex systems of non-linear polynomial equations. These solutions are based on Gröbner basis computations [132, 134] and on special manipulations with equations [112] and are widely used in many applications such as 3D reconstruction [123, 68] and structure-from-motion [124, 125]. This is because they are very effective as hypothesis generators in popular RANSAC paradigm [50] or can be used for initializing the bundle adjustment [61].

Even though these solutions [112, 132, 134] are practical and efficient they do not cover all problems which appear in real situations. Nowadays different omnidirectional cameras and wide angle lenses are often used in many applications. In the case of such cameras the projection is often modeled as the perspective projection followed by a radial "distortion" in the image plane. In fact, not only wide angle lenses but almost all consumer camera lenses suffer from some radial distortion.

It was observed that ignoring the distortion, even for standard cameras, may lead to significant errors in 3D reconstruction [51], metric measurements from images or when calibrating cameras. Therefore, in such applications, other algorithms than the algorithms for standard pinhole cameras usually need to be used.

Whereas for problems of estimating epipolar geometry of perspective cameras there exist several minimal solutions, such as the mentioned solutions to the 5-point [112, 132] or the 6-point focal length problem [134], solutions for similar problems of estimating epipolar geometry and

radial distortion parameters for cameras with radial distortion were missing. This was caused by the fact that these "minimal radial distortion problems" result in more difficult systems of polynomial equations. Therefore, the existing methods mainly focused on proposing new models for cameras with radial distortion and on choosing simpler systems which could be solved easier from a higher number of correspondences.

In this section we therefore extend the family of minimal relative pose solvers by new efficient solutions to three problems of estimating epipolar geometry and radial distortion parameters from point correspondences. The first one is the problem of simultaneous estimation of the fundamental matrix and a common radial distortion parameter for two uncalibrated cameras from eight image point correspondences, the second one is the estimation of the essential matrix and a common radial distortion parameter for two partially calibrated cameras and six image point correspondences and the third one is the problem of simultaneous estimation of the fundamental matrix and radial distortion parameters for two uncalibrated cameras with different distortions from nine image point correspondences.

Our solutions to all these three new minimal problems are based on the specialized Gröbner basis method presented in Section 4.4 and are generated using our automatic generator of Gröbner basis solvers presented in Chapter 6. Moreover, we also show the importance of the problem formulation and the selection of the strategy for generating polynomials from the ideal on the "8-point uncalibrated radial distortion problem". For this problem we also propose the "polynomial eigenvalue solution" based on the specialized resultant based method presented in Section 5.5.

Beside these new radial distortion problems, we show that two well known relative pose problems for cameras without radial distortion, i.e. the 5-point problem and the 6-point equal focal length problem lead to polynomial equations that can be solved robustly and efficiently as cubic and quadratic, eigenvalue problems. These new solution are very fast and in the case of the 6-pt solver also slightly more stable than the existing solution [134]. They are in some sense also more straightforward and easier to implement than previous solutions [112, 132, 134] since polynomial eigenvalue problems are well studied and their efficient numerical solvers are available. In this section we also propose a new Gröbner basis solution to the 6-point equal focal length problem which is generated using our automatic generator and we show that this solution is slightly more stable and also faster then the previous Gröbner basis solutions to this problem [134, 33].

We further extend the family of minimal relative pose solvers for cameras without radial distortion by two new efficient solutions to the 6-point minimal problem for one fully calibrated and one up to focal length calibrated camera. This problem was previously solved only from non-minimal number of seven point correspondences [142]. Our two solutions are again based on the specialized Gröbner basis method from Section 4.4 and on the specialized resultant based method and generalized eigenvalues from Section 5.5. These solvers work well even in some situations when the classical 6-point equal focal length algorithm [134] fails, e.g. when optical axes of the cameras are parallel or intersecting, and can be effectively used to reconstruct 3D scenes from collections of images with very few (in principle single) images with known focal lengths.

We finalize this section by proposing a new solution to the minimal problem of estimating epipolar geometry and unknown focal length from images of four points lying on a plane and one

off-the-plane point. This problem is known as the "plane + parallax" problem for cameras with unknown focal length and has not been solved before. Such problem is important in situations where some dominant plane is present in the scene, like in 3D reconstructions of cities or other man-made objects.

This section summarize our work presented in [89, 83, 84, 35, 87, 92] and is organized as follows: First we review basic geometric constraints for two cameras observing a common scene, i.e. we formulate relative pose problems as systems of polynomial equations. Then we, one-by-one, discuss all seven previously mentioned relative pose problems. In each subsection devoted to a one from these seven minimal problems we first overview previous solutions to this problem, then we formulate this problem as a system of polynomial equations and describe our Gröbner basis and in some cases also polynomial eigenvalue solutions to this problem and finally we study the stability of proposed solvers and compare them with previous solutions.

7.1.2 Geometric constraints

Consider a pair of cameras and the standard pinhole camera model [61]. In this model the image projection $\mathbf{u}_i$ of a 3D point $\mathbf{X}_i$ can be written as

$$\lambda_i \, \mathbf{u}_i = \mathtt{P} \, \mathbf{X}_i, \tag{7.1}$$

where $\mathtt{P}$ is a 3×4 projection matrix, λ_i is an unknown scalar value and points $\mathbf{u}_i = (u_i, v_i, 1)^\top$ and $\mathbf{X}_i = (x_i, y_i, z_i, 1)^\top$ are represented by their homogeneous coordinates.

The projection matrix $\mathtt{P}$ can be written as

$$\mathtt{P} = \mathtt{K} \, [\mathtt{R} \,|\, \mathbf{t}], \tag{7.2}$$

where $\mathtt{R} = [r_{ij}]_{i,j=1}^{3}$ is a 3×3 rotation matrix, which holds the information in which direction the camera is pointing , $\mathbf{t} = [t_x, t_y, t_z]^\top$ contains the information about the camera position and $\mathtt{K}$ is the 3×3 calibration matrix of the camera holding its intrinsic parameters.

Generally, this calibration matrix can be written as

$$\mathtt{K} = \begin{pmatrix} f & s & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{pmatrix}, \tag{7.3}$$

where f represents the camera focal length, s is the skew, α the aspect ratio of the pixels and (u, v) are the coordinates of the principal point [61].

Modern digital cameras have square pixels and the principal point close to the center of the image [61]. Therefore, for most of the applications this prior knowledge can be used and four out of the five internal calibration parameters can be safely set to some known fixed value (the skew s to 0, the pixel aspect ratio α to 1 and the principal point to the center of the image).

Adopting these calibration constraints has several advantages. First, the minimal number of points needed to solve relative pose problems is reduced. Second, since fewer parameters are estimated, the results are more stable.

We consider these three situations:

1. the camera is fully calibrated, which means that the calibration matrix $\mathtt{K}$ (7.3) is known and we can multiply the image coordinates $\mathbf{u}_i$ in (7.1) by $\mathtt{K}^{-1}$ and assume that $\mathtt{K}$ is the identity matrix in (7.2),

2. the camera is calibrated up to an unknown focal length f, which means that we can assume that the calibration matrix $\mathtt{K}$ (7.3) has the form

$$\mathtt{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{7.4}$$

3. the camera is completely uncalibrated.

It is known [61] that in the case of two fully calibrated cameras $\mathtt{P}$ and $\mathtt{P}'$, points $\mathbf{x}_i = \mathtt{K}^{-1}\mathbf{u}_i$ and $\mathbf{x}'_i = \mathtt{K}'^{-1}\mathbf{u}'_i$, which are projections of a 3D point $\mathbf{X}_i$, are geometrically constrained by the epipolar geometry coplanarity constraint

$$\mathbf{x}_i'^T \mathtt{E}\, \mathbf{x}_i = 0, \tag{7.5}$$

where $\mathtt{E}$ is a 3×3 rank-2 essential matrix with two equal singular values. These constraints on $\mathtt{E}$ can be written as

$$\det(\mathtt{E}) = 0, \tag{7.6}$$

$$2\,\mathtt{E}\,\mathtt{E}^\top\mathtt{E} - \mathrm{trace}(\mathtt{E}\,\mathtt{E}^\top)\mathtt{E} = \mathtt{0}. \tag{7.7}$$

Equation (7.6) is called the rank constraint and equation (7.7) the trace constraint [61].

The essential matrix encodes the relative pose of two cameras $\mathtt{P}$ and $\mathtt{P}'$. Without loss of generality, we can transfer 3D space and the camera pair so that the first camera will be of the form $\mathtt{P} = [\mathtt{I}_{3\times3}|0]$ and second of the general form $\mathtt{P}' = [\mathtt{R}|\mathbf{t}]$. In such a case the essential matrix $\mathtt{E}$ from (7.5) will have the form

$$\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}, \tag{7.8}$$

where $[\mathbf{t}]_\times$ is the skew symmetric matrix

$$[\mathbf{t}]_\times = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}. \tag{7.9}$$

This means that from the essential matrix $\mathtt{E}$ we can extract the relative position and orientation of two cameras.

The usual way [112, 132] for computing the essential matrix is to linearize relation (7.5) into the form $\mathtt{M}\,\mathbf{X} = 0$, where the vector $\mathbf{X}$ contains nine elements of the matrix $\mathtt{E}$ and $\mathtt{M}$ contains products of image measurements. The essential matrix $\mathtt{E}$ is then constructed as a linear combination of the conveniently reshaped null space vectors of the matrix $\mathtt{M}$. The dimension of the

null space depends on the number of point correspondences used. Additional constraints (7.6) and (7.7) are used to determine the coefficients in the linear combination of the null space vectors or to project an approximate solution to the space of correct essential matrices.

The fundamental matrix $\mathtt{F}$ describes uncalibrated cameras similarly as the essential matrix $\mathtt{E}$ describes calibrated cameras (7.5) , i.e.

$$\mathbf{u}_i'^T \mathtt{F}\, \mathbf{u}_i = 0 \tag{7.10}$$

and can be computed in analogous way. The relationship between the essential and the fundamental matrix can be written as

$$\mathtt{E} = \mathtt{K}'^\top \mathtt{F} \mathtt{K}. \tag{7.11}$$

For the fundamental matrix $\mathtt{F}$ we have similar constraint

$$\det(\mathtt{F}) = 0, \tag{7.12}$$

i.e. the constraint that the fundamental matrix is singular.

As we have described before in this section we will consider also a camera pair with unknown but equal focal length f and a camera pair with one fully-calibrated and one up to focal length calibrated camera. In both cases other calibration parameters are considered to be known. In such a case the calibration matrix $\mathtt{K}$ (7.3) is a diagonal matrix of the form (7.4).

Therefore, for two cameras with unknown but equal focal length, the essential matrix

$$\mathtt{E} = \mathtt{K}^\top\, \mathtt{F}\, \mathtt{K} = \mathtt{K}\, \mathtt{F}\, \mathtt{K}, \tag{7.13}$$

since $\mathtt{K}' = \mathtt{K}$ is diagonal. Since $\mathtt{K}$ is regular we have

$$\begin{aligned} \det(\mathtt{F}) &= 0, & (7.14)\\ 2\,\mathtt{F}\,\mathtt{Q}\mathtt{F}^\top \mathtt{Q}\mathtt{F} - \operatorname{trace}(\mathtt{F}\,\mathtt{Q}\mathtt{F}^\top \mathtt{Q})\,\mathtt{F} &= \mathtt{0}. & (7.15) \end{aligned}$$

Equation (7.15) is obtained by substituting the expression for the essential matrix (7.13) into the trace constraint (7.7), applying the substitution $\mathtt{Q} = \mathtt{K}\,\mathtt{K}$, and multiplying (7.7) by $\mathtt{K}^{-1}$ from left and right. Note that the calibration matrix can be written as

$$\mathtt{K} \simeq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix}. \tag{7.16}$$

to simplify equations.

For the case with one fully-calibrated and one up to focal length calibrated camera we have $\mathtt{E} = \mathtt{K}\,\mathtt{F}$ and we obtain constraints

$$\begin{aligned} \det(\mathtt{F}) &= 0, & (7.17)\\ 2\,\mathtt{F}\,\mathtt{F}^\top \mathtt{Q}\mathtt{F} - \operatorname{trace}(\mathtt{F}\mathtt{F}^\top \mathtt{Q})\,\mathtt{F} &= \mathtt{0}. & (7.18) \end{aligned}$$

Equation (7.18) is obtained similarly as equation (7.15).

Since almost all consumer camera lenses, including wide angle lenses, suffer from some radial distortion we consider here also problems of estimating relative pose of cameras with radial distortion.

Here we assume the one-parameter radial distortion division model proposed by Fitzgibon [51], which is given by the formula

$$\mathbf{x}_u \sim \mathbf{x}_d/(1 + \lambda r_d^2), \tag{7.19}$$

where λ is the distortion parameter, $\mathbf{x}_u = (x_u, y_u, 1)^\top$, and $\mathbf{x}_d = (x_d, y_d, 1)^\top$, are the corresponding undistorted and distorted image points, and r_d is the radius of $\mathbf{x}_d$ w.r.t. the distortion center. We assume that the distortion center is in the center of the image, i.e. $r_d^2 = x_d^2 + y_d^2$.

Since the epipolar constraint (7.5) and (7.10) contains undistorted image points $\mathbf{x}_i$ but we measure the distorted ones, we need to put the relation (7.19) into equation (7.10). Let $\mathbf{x}_{d_i} = (x_{d_i}, y_{d_i}, 1)^\top$ be the i^{th} measured distorted image point and let $r_{d_i}^2 = x_{d_i}^2 + y_{d_i}^2$. Then $\mathbf{x}_i = \left(x_{d_i}, y_{d_i}, 1 + \lambda r_{d_i}^2\right)^\top$ and the epipolar constraint for the uncalibrated cameras has the form

$$\left(x'_{d_i}, y'_{d_i}, 1 + \lambda' r'^2_{d_i}\right) \mathtt{F} \left(x_{d_i}, y_{d_i}, 1 + \lambda r_{d_i}^2\right)^\top = 0. \tag{7.20}$$

Similar relation holds for calibrated cameras and the essential matrix $\mathtt{E}$.

These are the basic equations which define almost all considered relative pose problems and which we will use to formulate these problems as systems of polynomial equations and solve them using the methods presented in Sections 4.4 and 5.5.

Now, we will describe all these relative pose problems in detail. We start with the problem of simultaneous estimation of the fundamental matrix and a common radial distortion parameter for two uncalibrated cameras from eight image point correspondences, also called the "8-point radial distortion problem". It is because this problem is quite practical, hasn't been solved before and on this problem we can illustratively show the importance of the problem formulation and the selection of the strategy for generating polynomials from the ideal used in the specialized Gröbner basis method 4.4.

7.1.3 8-point radial distortion problem

Previous solutions

The first non-minimal solution to the problem of simultaneous estimation of the epipolar geometry and the one parameter radial distortion division model from point correspondences was proposed by Fitzgibbon [51]. The division model proposed in this paper [51] leads to solving a system of algebraic equations, which is especially nice because the algebraic constraints of the epipolar geometry, e.g. $\det(\mathtt{F}) = 0$ (7.12), 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. However, the singularity constraint (7.12) on the fundamental matrix wasn't used in [51] and therefore nine point correspondences were necessary. Thanks to neglecting this constraint, the resulting equations could be easily formulated as a quadratic eigenvalue problem (5.47) and solved using standard solvers.

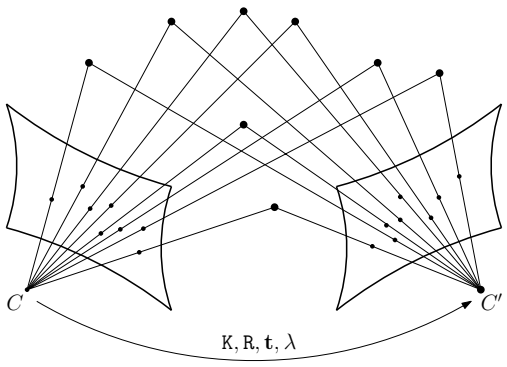

Figure 7.2: Illustration of the 8-point radial distortion problem.

Li and Hartley [97] treated the original Fitzgibbon's problem as a system of algebraic equations and used the hidden variable resultant technique [39] and symbolic determinants to solve them. Again no algebraic constraint on the fundamental matrix was used in this solution. The resulting technique solves exactly the same problem as [51] but in a different way. Our experiments have shown that the quality of the result in [97] was comparable to [51] but the technique [97] was considerably slower than the original technique [51] because in [97] Maple [126] was used to evaluate large polynomial determinants. Work [97] 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 the singularity constraint to really solve this problem.

The first minimal solution to the problem of estimating epipolar geometry and one parameter division model was proposed in our paper [89]. This solution used $\det \mathtt{F} = 0$, and therefore, the minimal number of eight point correspondences was sufficient to solve it. In this solution, we first transformed the initial nine polynomial equations (7.10) and (7.12) in nine unknowns to three polynomial equations in three unknowns. Then the equations were solved using the Gröbner basis method similar to that presented in Section 4.4. The final solver was created manually and consists of three G-J eliminations of a 8×22, 11×30, 36×50 matrices and the eigenvalue computation of a 16×16 matrix.

The improved version of this solver was proposed in our paper [83]. It was generated using the automatic generator of Gröbner basis solvers presented in Chapter 6 and consists only of one elimination of a 32×48 matrix.

Besides work presented in [89, 83], this section summarizes our work presented in two additional papers [90, 92], where we have proposed a different formulation of the 8-point radial distortion problem and its polynomial eigenvalue solution based on the method presented in Section 5.5.

We next provide the formulation of the problem of estimating the epipolar geometry and one parameter division model from eight point correspondences.

Problem formulation

In this problem we want to correct a radial lens distortion and estimate the epipolar geometry for an uncalibrated camera using the minimal number of image point correspondences in two views. We assume the one-parameter division distortion model [51], which is given by the formula

$$\mathbf{x}_u \sim \mathbf{x}_d/(1 + \lambda r_d^2), \tag{7.21}$$

where λ is the distortion parameter, $\mathbf{x}_u = (x_u, y_u, 1)^\top$ and $\mathbf{x}_d = (x_d, y_d, 1)^\top$, are the corresponding undistorted and distorted image points, and r_d is the radius of $\mathbf{x}_d$ w.r.t. the distortion center. We assume that the distortion center is in the center of the image and that pixels are square, i.e. $r_d^2 = x_d^2 + y_d^2$. We show in experiments that our approach works well even when these assumptions are not strictly satisfied. This was shown also in [51].

Undistorted image points $\mathbf{x}_u$ and $\mathbf{x}'_u$, which are projections of a 3D point $\mathbf{X}$, are geometrically constrained by the epipolar geometry constraint

$$\mathbf{x}_u'^\top \mathtt{F}\, \mathbf{x}_u = 0, \tag{7.22}$$

where $\mathtt{F}$ is a 3×3 rank-2 fundamental matrix

$$\mathtt{F} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{pmatrix}. \tag{7.23}$$

It is well known that for the standard uncalibrated case, without considering radial distortion, seven 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 eight point correspondences to estimate λ and the epipolar geometry simultaneously. To get this "8-point radial distortion 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

$$\mathbf{x}_{u_i}^\top(\lambda)\, \mathtt{F}\, \mathbf{x}'_{u_i}(\lambda) = 0, \quad i = 1, \ldots, 8, \tag{7.24}$$

and the singularity of $\mathtt{F}$

$$\det(\mathtt{F}) = 0, \tag{7.25}$$

where $\mathbf{x}'_u(\lambda), \mathbf{x}_u(\lambda)$ represent homogeneous coordinates of a pair of undistorted image points. These are the basic polynomial equations defining the "8-point radial distortion problem".

The complexity of solving polynomial equations depends mostly on the complexity of polynomials (degree, number of variables, form, etc.). It is usually 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. Unfortunately, reducing the number of variables often leads to higher degree polynomials. However, for "8-point radial distortion problem" the presented methods for solving systems of polynomial equations would result in huge and unusable solvers without performing an elimination of variables.

In the next we present two different elimination schemes. The first results in 7 polynomial equations in 7 variables and the second in 3 polynomial equations in 3 variables.

Eliminating variables - 7 equations in 7 unknowns

The epipolar constraint (7.24) gives 8 equations in 15 monomials ($f_{3,3}\lambda^2$, $f_{1,3}\lambda$, $f_{2,3}\lambda$, $f_{3,1}\lambda$, $f_{3,2}\lambda$, $f_{3,3}\lambda$, $f_{1,1}$, $f_{1,2}$, $f_{1,3}$, $f_{2,1}$, $f_{2,2}$, $f_{2,3}$, $f_{3,1}$, $f_{3,2}$, $f_{3,3}$) and 10 variables ($f_{1,1}$, $f_{1,2}$, $f_{1,3}$, $f_{2,1}$, $f_{2,2}$, $f_{2,3}$, $f_{3,1}$, $f_{3,2}$, $f_{3,3}$, λ).

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

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.26}$$

where $\mathbf{X} = (f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, f_{3,3}, f_{1,3}\lambda, f_{2,3}\lambda, f_{3,1}\lambda, f_{3,2}\lambda, f_{3,3}\lambda, f_{3,3}\lambda^2)^\top$ and $\mathtt{M}$ is a coefficient matrix.

If we denote the i^{th} row of the matrix $\mathtt{M}$ as m_i and write the i^{th} undistorted point $\mathbf{x}_{u_i}$ as

$$\mathbf{x}_{u_i} = \begin{pmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} 0 \\ 0 \\ r_{d_i}^2 \end{pmatrix}, \tag{7.27}$$

then $m_i = (x_{d_i}x'_{d_i},\, x_{d_i}y'_{d_i},\, x_{d_i},\, y_d x'_d,\, y_{d_i}y'_{d_i},\, y_{d_i},\, x'_{d_i},\, y'_{d_i},\, 1,\ x_{d_i}r'^2_{d_i},\, x_{d_i}r'^2_{d_i},\, r^2_{d_i}x'_{d_i},\, r^2_{d_i}y'_{d_i},\, r^2_{d_i} + r'^2_{d_i},\, r^2_{d_i}r'^2_{d_i})$.

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

$$\mathbf{X} = x_1\mathbf{N}_1 + x_2\mathbf{N}_2 + x_3\mathbf{N}_3 + x_4\mathbf{N}_4 + x_5\mathbf{N}_5 + x_6\mathbf{N}_6 + x_7\mathbf{N}_7, \tag{7.28}$$

where $\mathbf{N}_1, ..., \mathbf{N}_7 \in \mathbb{Q}^{15\times 1}$ are basic vectors of the null space and $x_1, \ldots, x_7$ are coefficients of the linear combination of these basic vectors. Assuming $x_7 \neq 0$, we can set $x_7 = 1$. Then we can write

$$\mathbf{X} = \sum_{i=1}^{7} x_i\mathbf{N}_i = \sum_{i=1}^{6} x_i\mathbf{N}_i + \mathbf{N}_7. \tag{7.29}$$

This means that for the j^{th} element X_j of the monomial vector $\mathbf{X}$ and the j^{th} element $N_{i,j}$ of the vectors $\mathbf{N}_i$ we have

$$X_j = \sum_{i=1}^{6} x_i N_{i,j} + N_{7,j}, \quad j = 1, .., 15. \tag{7.30}$$

Considering dependencies between monomials in $\mathbf{X}$ and $\det(\mathtt{F}) = 0$ we get 7 equations in 7

unknowns $x_1, x_2, x_3, x_4, x_5, x_6, \lambda$:

$$X_{10} = \lambda.X_3 \Rightarrow \sum_{i=1}^{6} x_i N_{i,10} + N_{7,10} = \lambda \sum_{i=1}^{6} x_i N_{i,3} + N_{7,3}, \tag{7.31}$$

$$X_{11} = \lambda.X_6 \Rightarrow \sum_{i=1}^{6} x_i N_{i,11} + N_{7,11} = \lambda \sum_{i=1}^{6} x_i N_{i,6} + N_{7,6}, \tag{7.32}$$

$$X_{12} = \lambda.X_7 \Rightarrow \sum_{i=1}^{6} x_i N_{i,12} + N_{7,12} = \lambda \sum_{i=1}^{6} x_i N_{i,7} + N_{7,7}, \tag{7.33}$$

$$X_{13} = \lambda.X_8 \Rightarrow \sum_{i=1}^{6} x_i N_{i,13} + N_{7,13} = \lambda \sum_{i=1}^{6} x_i N_{i,8} + N_{7,8}, \tag{7.34}$$

$$X_{14} = \lambda.X_9 \Rightarrow \sum_{i=1}^{6} x_i N_{i,14} + N_{7,14} = \lambda \sum_{i=1}^{6} x_i N_{i,9} + N_{7,9}, \tag{7.35}$$

$$X_{15} = \lambda.X_{14} \Rightarrow \sum_{i=1}^{6} x_i N_{i,15} + N_{7,15} = \lambda \sum_{i=1}^{6} x_i N_{i,14} + N_{7,14}, \tag{7.36}$$

$$\det(\mathtt{F}) = 0 \Rightarrow \det \begin{pmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{pmatrix} = 0. \tag{7.37}$$

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 $f_{3,3} = 1$) we have only 7 equations (six quadratic and one cubic) in 7 unknowns.

Eliminating variables - 3 equations in 3 unknowns

The second elimination procedure starts with the original set of equations (7.24) and (7.25). Assuming, e.g. $f_{3,3} \neq 0$, we can set $f_{3,3} = 1$. Then the epipolar constraint (7.24) gives 8 equations in 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 $(f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda)$.

Among these 9 variables we have four variables that appear in one monomial only $(f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2})$ and four variables that appear in two monomials $(f_{1,3}, f_{2,3}, f_{3,1}, f_{3,2})$. Since we have 8 equations from which each one contains all 15 monomials, we can eliminate 6 variables, the four variables $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}$ that appear in one monomial only, and two variables from the variables that appear in two monomials. We decided to eliminate $f_{1,3}$ 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 $\mathbf{X} = (f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda, f_{1,3}, f_{2,3}\lambda, f_{2,3}, f_{3,1}\lambda, f_{3,2}\lambda, \lambda^2, f_{3,1}, f_{3,2}, \lambda, 1)^\top$.

Then, 8 equations from the epipolar constraint can be written in a matrix form

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.38}$$

where $\mathtt{M}$ is the coefficient matrix and $\mathbf{X}$ is this reordered monomial vector. After computing G-J elimination $\widetilde{\mathtt{M}}$ of $\mathtt{M}$, we obtain 8 equations $\widetilde{\mathtt{M}}\,\mathbf{X} = \mathbf{0}$ of the form

$$\begin{pmatrix} 1 & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & 1 & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & 1 & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & 1 & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & 1 & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & 1 & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & & 1 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} f_{1,1} \\ f_{1,2} \\ f_{2,1} \\ f_{2,2} \\ f_{1,3}\lambda \\ f_{1,3} \\ f_{2,3}\lambda \\ f_{2,3} \\ f_{3,1}\lambda \\ f_{3,2}\lambda \\ \lambda^2 \\ f_{3,1} \\ f_{3,2} \\ \lambda \\ 1 \end{pmatrix} = \mathbf{0}. \tag{7.39}$$

Denoting polynomials corresponding to the rows of this matrix equation (7.39) as p_i, we can write

$$p_i = LT(p_i) + g_i(f_{3,1}, f_{3,2}, \lambda) = 0, \tag{7.40}$$

where $LT\,(p_i)$ is the leading term of the polynomial p_i which is our case $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}$, $f_{1,3}\lambda$, $f_{1,3}$, $f_{2,3}\lambda$ and $f_{2,3}$ for $i = 1, \ldots, 8$. Polynomials $g_i(f_{3,1}, f_{3,2}, \lambda)$ are 2^{nd}order polynomials in three variables $f_{3,1}, f_{3,2}, \lambda$. Next we express 6 variables, $f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}$ as the following functions of the remaining three variables $f_{3,1}, f_{3,2}, \lambda$

$$f_{1,1} = -g_1(f_{3,1}, f_{3,2}, \lambda), \tag{7.41}$$

$$f_{1,2} = -g_2(f_{3,1}, f_{3,2}, \lambda), \tag{7.42}$$

$$f_{1,3} = -g_6(f_{3,1}, f_{3,2}, \lambda), \tag{7.43}$$

$$f_{2,1} = -g_3(f_{3,1}, f_{3,2}, \lambda), \tag{7.44}$$

$$f_{2,2} = -g_4(f_{3,1}, f_{3,2}, \lambda), \tag{7.45}$$

$$f_{2,3} = -g_8(f_{3,1}, f_{3,2}, \lambda). \tag{7.46}$$

We can substitute expressions (7.41) - (7.46) into the remaining two equations p_5 and p_7 from the epipolar constraint (7.40) and also to the singularity constraint (7.25) for $\mathtt{F}$. In this way we obtain 3 polynomial equations in 3 unknowns (two 3^{rd}order polynomials and one 5^{th}order polynomial)

$$\lambda(-g_6(f_{3,1}, f_{3,2}, \lambda)) + g_5(f_{3,1}, f_{3,2}, \lambda) = 0, \tag{7.47}$$

$$\lambda(-g_8(f_{3,1}, f_{3,2}, \lambda)) + g_7(f_{3,1}, f_{3,2}, \lambda) = 0, \tag{7.48}$$

$$\det \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & -g_8 \\ f_{3,1} & f_{3,2} & 1 \end{pmatrix} = 0. \tag{7.49}$$

These three equations in three variables are simpler and more suitable for methods presented in Sections 4.4 and 5.5 than the seven polynomial equations (7.31)- (7.37) in seven unknowns resulting from the first elimination procedure. However, the disadvantage of this second elimination procedure is that it assumes that the elements $f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}$, and $f_{3,3}$ of the fundamental matrix $\mathtt{F}$ (7.23) are non-zero. This introduces some "critical motions" to this problem, e.g. pure translation, which are in fact not "critical motions" of the problem itself, but of its formulation.

Gröbner basis solution of the "7 in 7" formulation

In this subsection we describe in detail how to create an efficient Gröbner basis solver for solving all "non-degenerate" instances of the first formulation of the "8-point radial distortion problem" leading to seven equations in seven unknowns. To create this solver we use the specialized Gröbner basis method presented in Section 4.4.

Identification of basic parameters

In the first step of the offline phase of the process of creating an efficient online solver we need to identify the basic parameters of the system of seven polynomial equations (7.31)-(7.37) in seven unknowns $x_1, \ldots, x_6, \lambda$ resulting from the proposed formulation. This means that we need to compute the number of solutions and to find the standard monomial basis B (4.20) of the quotient ring A (4.12).

We use algebraic geometric software Macaulay 2 [59], which can compute in finite fields for this purpose. If the basis B remains stable for many different random coefficients from $\mathbb{Z}_p$, it is generically equivalent to the basis of the original system of polynomial equations [140].

This way we have found that our problem has 16 solutions, and for the fixed graded reverse lexicographic ordering with $x_1 > x_2 > x_3 > x_4 > x_5 > \lambda > x_6$, the basis B of the quotient ring $A = \mathbb{C}[x_1, x_2, x_3, x_4, x_5, \lambda, x_6]/I$ has the form $B = \{[x_6^3], [\lambda^2], [x_1x_6], [x_2x_6], [x_3x_6], [x_4x_6], [x_5x_6], [x_6^2], [x_1], [x_2], [x_3], [x_4], [x_5], [\lambda], [x_6], [1]\}$.

Construction of the multiplication matrix

Having the basis B, we know the form of the polynomials q_i (4.115) necessary for constructing the multiplication matrix $\mathtt{M}_{\mathbf{x}^\beta}$ for multiplication by $[\mathbf{x}^\beta]$ in A. For this problem we have decided to create the multiplication matrix $\mathtt{M}_\lambda$. This means that we need to generate polynomials from the ideal I of the form $q_i = \lambda\mathbf{x}^{\alpha(i)} - \sum_{k=1}^{16} c_k^i \mathbf{x}^{\alpha(k)} \in I$, where $[\mathbf{x}^{\alpha(k)}] \in B$ and $c_k^i \in \mathbb{C}$.

To generate these polynomials from the initial polynomials (7.31)-(7.37), we first need to select the strategy of generating polynomials from the ideal I.

Unfortunately, in this case, the single elimination strategy, which is presented in Section 4.4.3 in Algorithm 9 and that is implemented in our automatic generator of Gröbner basis solvers presented in Chapter 6, doesn't result in a feasible solver. This is caused by the fact that in this

formulation we have a quite large number of variables, and for a such large number of variables the number of generated monomials grows with the growing monomial degree very fast.

Therefore, for this formulation we have decided to use the second strategy of generating polynomials from the ideal I presented in Section 4.4.3, i.e. the multiple elimination strategy from Algorithm 8. In this strategy, we generate only polynomials up to some degree d in each step and we eliminate them each time by G-J elimination. We increase d when it is not possible to generate more polynomials of degree d by multiplying already generated polynomials by some monomials, and all polynomials q_i (4.115) have not yet been generated.

We can thus systematically generate all necessary polynomials for constructing the multiplication matrix $\mathtt{M}_\lambda$, i.e. we can find an "elimination template", or in fact an admissible Elimination trace for constructing this matrix, see Definition 21. Unfortunately, as it was already described in Section 4.4.3, we also generate many unnecessary polynomials.

Next we describe the main steps of the "elimination template" in which some of these unnecessary polynomials are not generated. This "elimination template" was created manually, using information obtained by computing the Gröbner basis in some finite prime field and using the mentioned multiple elimination strategy. A very similar "elimination template" can also be obtained automatically.

The basic steps of the "elimination template" for generating all polynomials necessary for constructing the multiplication matrix $\mathtt{M}_\lambda$ are as follows:

1. We begin with six 2^{nd} degree polynomials $f_1^{(0)}, \ldots, f_6^{(0)}$ (7.31)-(7.36) and one 3^{rd} degree polynomial $f_7^{(0)} = \det(\mathtt{F}) = 0$ (7.37). We perform G-J elimination of the matrix representing these polynomials and obtain the reduced polynomials $f_1^{(1)}, \ldots, f_7^{(1)}$.

2. Since polynomials $f_1^{(1)}, \ldots, f_6^{(1)}$ have degree two and polynomial $f_7^{(1)}$ degree three, in the next step we multiply $f_1^{(1)}, \ldots, f_6^{(1)}$ by $1, x_1, x_2, x_3, x_4, x_5, \lambda, x_6$ and add $f_7^{(1)}$ to get 49 2^{nd} and 3^{rd} degree polynomials $f_1^{(2)}, \ldots, f_{49}^{(2)}$. They can be represented by 119 monomials and a 49×119 matrix with rank 49, which we simplify by one G-J elimination.

3. We obtain 15 new 2^{nd} degree polynomials ($f_{29}^{(2)} \ldots, f_{43}^{(2)}$), six old 2^{nd} degree polynomials (reduced polynomials $f_1^{(1)}, \ldots, f_6^{(1)}$, now $f_{44}^{(2)} \ldots, f_{49}^{(2)}$) and 28 polynomials of degree three. In order to avoid adding 4^{th} degree polynomials in 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 $f_1^{(2)}, \ldots, f_{49}^{(2)}$. Thus obtaining 154 polynomials $f_1^{(3)}, \ldots, f_{154}^{(3)}$ representable by a 154×119 matrix, which has rank 99. We simplify it by G-J elimination and obtain $f_1^{(4)}, \ldots, f_{99}^{(4)}$.

4. The only 4^{th} degree polynomial that we need for constructing the multiplication matrix $\mathtt{M}_\lambda$ is a polynomial $q_1 = \lambda x_6^3 - \sum_{k=1}^{16} c_k^1 \mathbf{x}^{\alpha(k)} \in I$, where $\left[\mathbf{x}^{\alpha(k)}\right] \in B$ and $c_k^1 \in \mathbb{Q}$. To obtain this polynomial, we only need to add monomial multiples of one polynomial g from $f_1^{(4)}, ..., f_{99}^{(4)}$ with 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 106×126 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 $LM(g) = \lambda\, x_6^2$ is already 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 $c_1, ..., c_4 \in \mathbb{Q}$ and $h' = \sum_{k=1}^{16} a_k \mathbf{x}^{\alpha(k)}$, where $\left[\mathbf{x}^{\alpha(k)}\right] \in B$, instead of the desired $q_1 = \lambda\, x_6^3 + h_1$, $h_1 = -\sum_{k=1}^{16} c_k^1 \mathbf{x}^{\alpha(k)}$.

5. Among the polynomials $f_1^{(6)}, ..., f_{106}^{(6)}$, there are 12 out of the 14 polynomials that are required for constructing the multiplication matrix $\mathtt{M}_\lambda$. The first polynomial that is missing is the above mentioned polynomial $q_1 = \lambda\, x_6^3 + h_1$. To obtain this polynomial from q, we need to generate polynomials from the ideal with the 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 $q_2 = \lambda^3 + h_2$, $h_2 = -\sum_{k=1}^{16} c_k^2 \mathbf{x}^{\alpha(k)}$. Unfortunately, all these 3^{rd} degree polynomials from the ideal I can be, obtained only by eliminating some 4^{th} degree polynomials. To get these 4^{th} degree polynomials, the polynomial with the 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 λ and subtracted from the polynomial with the leading monomial λx_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 the G-J elimination, polynomials with the leading monomials $x_2 x_6^2$,$x_3 x_6^2$, $x_4 x_6^2$, $x_5 x_6^2$, λ^3 are obtained.

In this way we can obtain all polynomials necessary for constructing the multiplication matrix $\mathtt{M}_\lambda$.

Online Gröbner basis solver

Note that all the processing until now, the identification of basic parameters and finding an "elimination template" for constructing the multiplication matrix, i.e. an admissible Elimination trace, needs to be done only once when studying this problem and building its online solver. The final online solver for solving all systems of polynomial equations of the form (7.31)-(7.37) with "non-degenerate" coefficients, i.e. for solving the "8-point radial distortion problem", consists of the following steps:

1. Take seven input polynomial equations (7.31)-(7.37) with concrete coefficients from $\mathbb{Q}$ and perform all five steps of the "elimination template".
2. After the final G-J elimination of the matrix representing all polynomials from the fifth step of this "elimination template", extract from this matrix coefficients corresponding to the polynomials q_i that are necessary for constructing the multiplication matrix $\mathtt{M}_\lambda$.
3. Use extracted coefficients to create the multiplication matrix $\mathtt{M}_\lambda$.
4. Compute 16 left eigenvectors $\mathbf{v}_i$ of the multiplication matrix $\mathtt{M}_\lambda$ or equivalently 16 right eigenvectors of $\mathtt{M}_\lambda^\top$.
5. Extract 16 solutions for $(x_1, \ldots, x_6, \lambda)$ from these eigenvectors.
6. Use these 16 solutions to reconstruct solutions to the fundamental matrix $\mathtt{F}$.

This means that the final online solver for this formulation of the "8-point radial distortion problem" consists only of five G-J eliminations and computations of eigenvectors of a 16×16 matrix.

Gröbner basis solution of the "3 in 3" formulation

Next we describe how to create an efficient Gröbner basis solver for solving all "non-degenerate" instances of the second formulation of the "8-point radial distortion" problem leading to three polynomial equations in three unknowns. To create this solver we use the specialized Gröbner basis method presented in Section 4.4. Moreover, since for this formulation the single elimination strategy for generating polynomials from the ideal is suitable, this efficient solver can be created using our automatic generator of Gröbner basis solvers presented in Chapter 6.

Identification of basic parameters

In the first step of the process of creating an efficient "online" Gröbner basis solver, we need to compute the number of solutions and to find the standard monomial basis B (4.20) of the quotient ring A (4.12) for our three equations (7.47)-(7.49) in three unknowns.

We compute B in a randomly chosen finite prime field $\mathbb{Z}_p$, in our case we choose $p = 30097$, where exact arithmetic can be used and numbers can be represented in a simple and efficient way.

Since three equations (7.47)-(7.49) from this formulation were obtained from the same initial equations (7.24)-(7.25) as the seven equations from the previous formulation, we have again 16 solutions.

To create the multiplication matrix, we use the graded reverse lexicographic ordering with $f_{3,1} > f_{3,2} > \lambda$. With this ordering, we get the basis $B = \left\{ \left[f_{3,1}^3\right], \left[f_{3,1}^2 f_{3,2}\right], \left[f_{3,1} f_{3,2}^2\right], \left[f_{3,2}^3\right], \left[f_{3,2}^2 \lambda\right], \left[\lambda^3\right], \left[f_{3,1}^2\right], \left[f_{3,1} f_{3,2}\right], \left[f_{3,2}^2\right], \left[f_{3,1}\lambda\right], \left[f_{3,2}\lambda\right], \left[\lambda^2\right], \left[f_{3,1}\right], \left[f_{3,2}\right], \left[\lambda\right], \left[1\right] \right\}$ of the quotient ring $A = \mathbb{C}\left[f_{3,1}, f_{3,2}, \lambda\right] / I$.

Construction of the multiplication matrix

Once we know the basis B, we can find an "elimination template" for constructing the multiplication matrix, i.e. an admissible Elimination trace, see Definition 21. For creating the multiplication matrix we tested all three unknowns $f_{3,1}$, $f_{3,2}$ and λ. We have found that the best choice for the "action variable", which leads to the simplest solution, is λ. Therefore, we construct the multiplication matrix $\mathtt{M}_\lambda$.

The method described in Section 4.4 calls for generating polynomials q_i of the form $q_i = \lambda \mathbf{x}^{\alpha(i)} - \sum_{k=1}^{16} c_k^i \mathbf{x}^{\alpha(k)} \in I$, where $\left[\mathbf{x}^{\alpha(i)}\right] \in B$ and $c_k^i \in \mathbb{C}$.

As described in Section 4.4.3, this can be done using several strategies. One possible way is to start with the initial polynomials and then systematically generate new polynomials from I by multiplying already generated polynomials by individual variables and reducing them each time by G-J elimination, i.e. to use the multiple elimination strategy described in Section 4.4.3 in Algorithm 8. In [89] we used this strategy to solve the previous "7 in 7" formulation and to solve this "3 in 3" formulation, for which it results in three G-J eliminations of $8 \times 22, 11 \times 30$ and 36×50 matrices.

For this formulation, however, it is more suitable to use the second strategy from Section 4.4.3, i.e. the single elimination strategy where all necessary polynomials q_i are generated in one step by multiplying the initial three polynomial equations by selected monomials and reducing all the generated polynomials at once by one G-J elimination.

The set of selected monomials, which should be used to multiply initial polynomial equations, i.e. an admissible Elimination trace, can be found again only once in advance. To do this, we use Macaulay 2. We have found that obtaining all necessary polynomials for creating the multiplication matrix $\mathtt{M}_\lambda$ calls for generating all monomial multiples of the initial three polynomial equations up to the total degree six. This means that we need to multiply our two 3^{rd} degree polynomial equations (7.47) and (7.48) by all monomials up to degree three and our 5^{th} degree polynomial equation (7.49), from the rank constraint, by all 1^{st} degree monomials.

In this way we generate 41 new polynomials which, together with the initial three polynomial equations, form a system of 44 polynomials in 84 monomials. Since these polynomials were generated systematically, they contain also polynomials that do not affect the resulting multiplication matrix, i.e. they are not necessary for generating polynomials q_i . We remove all these unnecessary polynomials using Algorithm 10 presented in Section 4.4.3.

For this formulation, running Algorithm 10 results in 32 polynomial equations in 84 monomials. After rewriting these 32 polynomials in the matrix form and performing the G-J elimination of the corresponding coefficient matrix, we obtain all polynomials which we need for constructing the multiplication matrix $\mathtt{M}_\lambda$.

Before G-J elimination we can also remove "unnecessary monomials" from these 32 polynomials, i.e. monomials which do not have impact on the solution, as described in Section 4.4.3. In this way we obtain polynomials which can be represented by 32×48 coefficient matrix $\mathtt{M}$.

Online Gröbner basis solver

The final online solver for solving all "non-degenerate" instances of three equations (7.47)-(7.49) in three unknowns, i.e. for solving the "8-point radial distortion problem" then does only one G-J elimination of the 32×48 coefficient matrix $\mathtt{M}$. This matrix contains coefficients that arise from concrete image measurements. After G-J elimination of $\mathtt{M}$, the multiplication matrix $\mathtt{M}_\lambda$ can be created from rows which correspond to the polynomials with leading monomials from the set $(\lambda \cdot B) \setminus B$. The eigenvectors of this multiplication matrix $\mathtt{M}_\lambda$ give solutions for $f_{3,1}, f_{3,2}, \lambda$. Backsubstituting the solutions to equations (7.41)-(7.46) gives 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.

The final online solver consists of these steps:

1. Take three input polynomial equations (7.47)-(7.49) with concrete coefficients from $\mathbb{Q}$ and from these coefficients create the 32×48 coefficient matrix $\mathtt{M}$ from found admissible Elimination trace, i.e. from found "elimination template".

2. After the G-J elimination of the coefficient matrix $\mathtt{M}$, extract from this matrix coefficients corresponding to the polynomials q_i necessary for constructing the multiplication matrix $\mathtt{M}_\lambda$.

3. Use extracted coefficients to create the multiplication matrix $\mathtt{M}_\lambda$.
4. Compute 16 left eigenvectors of the multiplication matrix $\mathtt{M}_\lambda$, or equivalently, 16 right eigenvectors of $\mathtt{M}_\lambda^\top$.
5. Extract 16 solutions for $(f_{3,1}, f_{3,2}, \lambda)$ from the computed eigenvectors.
6. Backsubstitute the solutions into equations (7.41)-(7.46) to obtain solutions to the fundamental matrix $\mathtt{F}$.

Input to the automatic generator

The online solver for solving all "non-degenerate" instances of three equations (7.47)-(7.49) in three unknowns can be created using our automatic generator presented in Chapter 6.

The input to our automatic generator for this problem can be seen in Figure (7.3). With this input, the automatic generator automatically performs all steps of the offline phase, i.e. the identification of basic parameters and finding the admissible Elimination trace, and as an output gives the final efficient online solver for solving the "3 in 3" formulation of the 8-point radial distortion problem.

Polynomial eigenvalue solution of the "3 in 3" formulation

The polynomial eigenvalue solution of the "3 in 3" formulation of the problem of simultaneous estimation of the epipolar geometry and the one parameter division model from eight point correspondences starts with three equations (7.47)-(7.49) in three unknowns $f_{3,1}, f_{3,2}, \lambda$ and 39 monomials.

Equations (7.47) and (7.48) are of degree three and equation (7.49) is of degree five. If we take the unknown radial distortion parameter λ to play the role of α in (5.44), we can rewrite these three equations as

$$\left(\lambda^4 \mathtt{C}_3 + \lambda^3 \mathtt{C}_3 + \lambda^2 \mathtt{C}_2 + \lambda\, \mathtt{C}_1 + \mathtt{C}_0\right) \mathbf{v} = 0, \tag{7.50}$$

where $\mathbf{v} = (f_{3,1}^3, f_{3,1}^2 f_{3,2}, f_{3,1} f_{3,2}^2, f_{3,2}^3, f_{3,1}^2, f_{3,1} f_{3,2}, f_{3,2}^2,\ f_{3,1}, f_{3,2}, 1)^\top$ is a 10×1 vector of monomials and $\mathtt{C}_4$, $\mathtt{C}_3$, $\mathtt{C}_2$, $\mathtt{C}_1$ and $\mathtt{C}_0$ are 3×10 coefficient matrices.

Polynomial eigenvalue formulation (5.44) requires square coefficient matrices $\mathtt{C}_\mathtt{j}$. Unfortunately in this case, in contrary to computer vision problems described in [51, 84], we do not have square coefficient matrices. We have only three equations and ten monomials in the vector $\mathbf{v}$. Therefore, we will use the Modified Macaulay based method to transform the system of polynomial equations to the PEP, see Section 5.5.1.

Equations (7.47) and (7.48) are equations of degree one when considered as polynomials in $(\mathbb{C}[\lambda])[f_{3,1}, f_{3,2}]$, while equation (7.49) is of degree three. Therefore, the degree d (5.60) defined in this method as $\mathtt{d} = \sum_{i=1}^{n} (d_i - 1) + 1$ is equal to three. This means that the sets $\overline{S_i}$ (5.64) will consist of monomials $\overline{S_1} = \left\{f_{3,1}^3, f_{3,1}^2 f_{3,2}, f_{3,1} f_{3,2}^2, f_{3,1}^2, f_{3,1} f_{3,2}, f_{3,1}\right\}$, $\overline{S_2} = \left\{f_{3,2}^3, f_{3,1}^2 f_{3,2}, f_{3,1} f_{3,2}^2, f_{3,2}^2, f_{3,1} f_{3,2}, f_{3,2}\right\}$, $\overline{S_3} = \{1\}$ after the dehomogenization (see Modified Macaulay based method in Section 5.5.1).

```
g1 = transpose(gbs_Vector('g1', 7));
g2 = transpose(gbs_Vector('g2', 7));
g3 = transpose(gbs_Vector('g3', 7));
g4 = transpose(gbs_Vector('g4', 7));
g5 = transpose(gbs_Vector('g5', 7));
g6 = transpose(gbs_Vector('g6', 7));
g7 = transpose(gbs_Vector('g7', 7));
g8 = transpose(gbs_Vector('g8', 7));

syms f31 f32 k;
mon = [k*f31,k*f32,k^2,f31,f32,k,1]';

% parametrization of the fundamental matrix with three unknowns
f11 = -g1*mon;
f12 = -g2*mon;
f13 = -g6*mon;
f21 = -g3*mon;
f22 = -g4*mon;
f23 = -g8*mon;

% fundamental matrix
F = [f11 f12 f13; f21 f22 f23; f31 f32 1];

% three polynomial equations
eq(1) = -(k*g6)*mon + g5*mon;
eq(2) = -(k*g8)*mon + g7*mon;
eq(3) = det(F);

% three unknowns
unknown = {'f31' 'f32' 'k'};

% known parameters
vars = transpose([g1(:); g2(:); g3(:); g4(:); g5(:); g6(:);
g7(:); g8(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% call code generator
[res export] = gbs_CreateCode('ku8pt', eq, known, unknown);
```

Figure 7.3: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the solver for the "3 in 3" formulation of the "8-point radial distortion problem".

The extended system of equation (5.65), which is equivalent to the initial system of equations (7.47)-(7.49), will therefore consist of 13 equations, i.e. equations (7.47) and (7.48) multiplied with the monomials $f_{3,1}^2, f_{3,1}f_{3,2}, f_{3,2}^2, f_{3,1}, f_{3,2}, 1$, and equation (7.49). It will contain only 10 monomials. Therefore, this system can be rewritten to the polynomial eigenvalue form (7.50) with coefficient matrices $\mathtt{C}_j$ of size 13×10.

Since we have more equations than the monomials, we can select 10 equations from them, such that the resulting system will have a small condition number. The resulting formulation will be in this case a PEP of degree four.

After solving this PEP, we obtain 40 eigenvalues, solutions for λ, and 40 corresponding eigenvectors $\mathbf{v}$ from which we extract solutions for $f_{3,1}$ and $f_{3,2}$. To do this we normalize solutions for eigenvectors $\mathbf{v}$ to have the last coordinate 1 and extract values from $\mathbf{v}$ that correspond to $f_{3,1}$ and $f_{3,2}$.

Between these 40 eigenvalues there are several "parasitic" zero eigenvalues. Eleven from these zero eigenvalues correspond to zero columns of the coefficient matrices $\mathtt{C}_j$ and can be easily removed before computing the eigenvalue problem using the method describe in Section 5.5.1. This means that we reduce the original 40×40 eigenvalue problem to a 29×29 eigenvalue problem.

However, we still obtain more than 16 solutions which is the number of solutions of the original system of polynomial equations. This is because this polynomial eigenvalue method solves a relaxed version of the proposed problem. This relaxed version does not take into account monomial dependencies induced by the problem, e.g. $\mathbf{v}(1) = \mathbf{v}(8)^3$. Therefore, the solution contains $\mathbf{v}$'s that automatically (within limits of numerical accuracy) satisfy these constraints and additional general eigenvectors $\mathbf{v}$'s that do not satisfy them. However, such $\mathbf{v}$'s can be eliminated by verifying the monomial dependences, or by substituting the solutions to the initial polynomial equations (7.47)-(7.49) as described in Section 5.5.1.

Solutions satisfying the monomial constraints on $\mathbf{v}$ will be obtained for exact as well as noisy data. Since we are solving the minimal problem, noisy data can be viewed as perfect input for a different camera configuration. Hence, we again obtain a solution satisfying the monomial constraints on the vector $\mathbf{v}$.

After finding the solutions for $k, f_{3,1}$ and $f_{3,2}$, we can use equations (7.41)-(7.46) to get solutions for the fundamental matrix $\mathtt{F}$.

Experiments

We have tested our algorithms on both synthetic data (with various levels of noise, outliers, radial distortions and radial distortion center shifts) and on real datasets, and compared them with the minimal Gröbner basis algorithm presented in [89] and with the non-minimal 9-point algorithms for correcting radial distortion [51, 97].

From both of our solvers we get several (complex) roots. In the case of the Gröbner basis solver we get 16 roots, and in the case of the polynomial eigenvalue solver (polyeig solver), we get 29 roots from which maximally 16 roots are satisfying all monomial constraints.

In general, 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

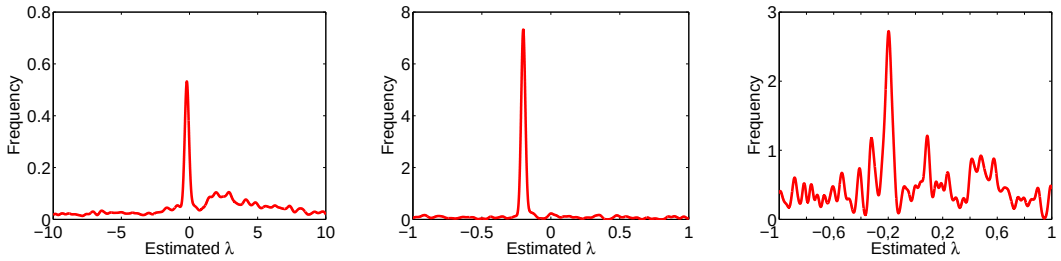

Figure 7.4: Distribution of real roots using kernel voting for 1000 point matches, 200 estimations and $\lambda_{true} = -0.2$. (Left) Roots within the range $[-10, 10]$ and no noise contamination. (Center) Roots within the range $[-1, 1]$ and no noise. (Right) Roots within the range $[-1, 1]$, 80% of inliers and $0.5px$ noise, where noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

roots of the 16 roots obtained by solving the equations for one input as real roots from different inputs, and then use RANSAC [50] or kernel voting [97, 89] for several inputs to select the "best root" among all computed roots. The kernel voting is done by a Gaussian kernel with a fixed variance and the estimate of λ is found as the position of the highest peak. See [97, 89] for more on kernel voting for this problem.

Figure 7.4 shows distributions of real roots for a typical situation constructed using the kernel voting. In this case 200 estimates were computed for the ground truth $\lambda = -0.2$. It can be seen 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 and estimate λ as the position of the highest peak.

Figure 7.4 shows that it is meaningful to vote for λ's either (i) within a range where the most of the computed roots fall (in our case [-10,10]), Figure 7.4 (left), or (ii) within the smallest range in which we are sure that the ground truth lie (in our case [-1,1]), Figure 7.4 (middle). Moreover, from Figure 7.4 (right) it can be seen that this standard kernel voting method is robust to noise and small number of outliers (up to 30%). For a higher number of outliers, more advanced kernel voting techniques, e.g. weighted kernel voting method, which takes into account the number of inliers and combines RANSAC with kernel voting [27], can be used.

Synthetic data sets

We initially studied our algorithms on synthetically generated ground-truth 3D scenes. These scenes were generated using the following procedure:

1. Generate a 3D scene consisting of N $(= 1000)$ random points within a 3D cube.
2. Project $M\%$ of the points on image planes of two displaced cameras. These are matches. Generate $(100 - M)\%$ random points in both images. These are mismatches. Altogether, they become undistorted correspondences.
3. Apply the radial distortion to the undistorted correspondences to generate noiseless distorted points.
4. Add Gaussian noise of standard deviation σ to the distorted points assuming a 1000×1000

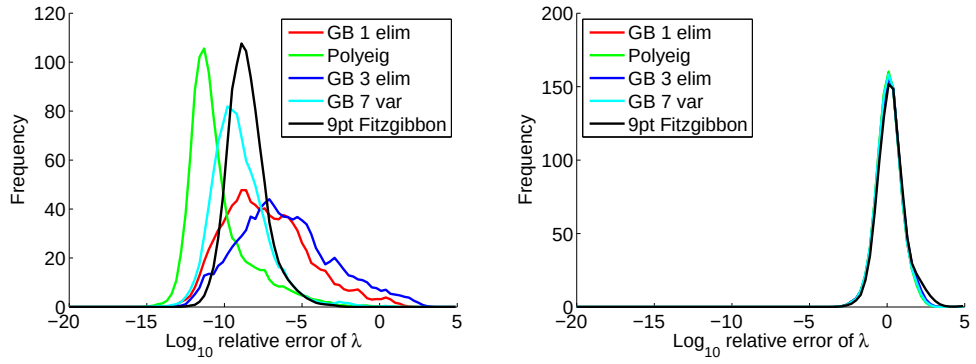

Figure 7.5: Log_{10} relative error of the radial distortion parameter λ obtained by selecting the real root closest to the ground truth value $\lambda_{th} = -0.2$ for (left) noise free synthetic scene and (right) noise $0.5px$, where noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

pixel image. By noise $1px$ we will mean Gaussian noise with standard deviation $\sigma = \frac{1}{3}px$ in all our experiments.

In the first synthetic experiment we have studied the numerical stability of all proposed solvers and compared them with the non-minimal 9-point solver [51] and with the minimal Gröbner basis solver proposed in [89]. We have focused on the radial distortion parameter estimation because once we have a good radial distortion estimate, the fundamental matrix is usually good or can be recomputed using well known algorithms for perspective cameras [61, 112, 132].

Figure 7.5 (left) shows the log_{10} relative error of the radial distortion parameter λ obtained by selecting the real root closest to the ground truth value $\lambda_{th} = -0.2$ for noise free synthetic scene.

In this experiment we have compared all three proposed minimal solvers, the Gröbner basis solver for the "3 in 3" formulation (red), the Gröbner basis solver for the "7 in 7" formulation (cyan) and the polyeig solver (green), with the minimal Gröbner basis solver proposed in [89] (blue) and the non-minimal solver [51] (black).

It can be seen that the polyeig solver (green) is numerically most stable among all tested solvers. In the case of the Gröbner basis solvers the solver for the "7 in 7" formulation (cyan) is more stable than the solvers for the "3 in 3 formulation" (red, blue), likely due to the fact that the "3 in 3" formulation has several critical configurations that the "7 in 7" formulation does not have. Moreover, it can be seen that one Gauss-Jordan elimination (red) slightly improves the precision with respect to three Gauss-Jordan eliminations [89] (blue). For data with noise, the results of all solvers are almost the same, see Figure 7.5 (right).

In the following experiments we will consider only the two proposed solvers for the "3 in 3" formulation. It is because these solvers are faster, simpler and of comparable precision than the proposed Gröbner basis solver for the "7 in 7" formulation. Moreover, the Gröbner basis solver for the "3 in 3" formulation can be easily obtained using our automatic generator presented in Chapter 6, and we will show that this solver provides satisfactory results also for real applications even it suffers from some critical configurations.

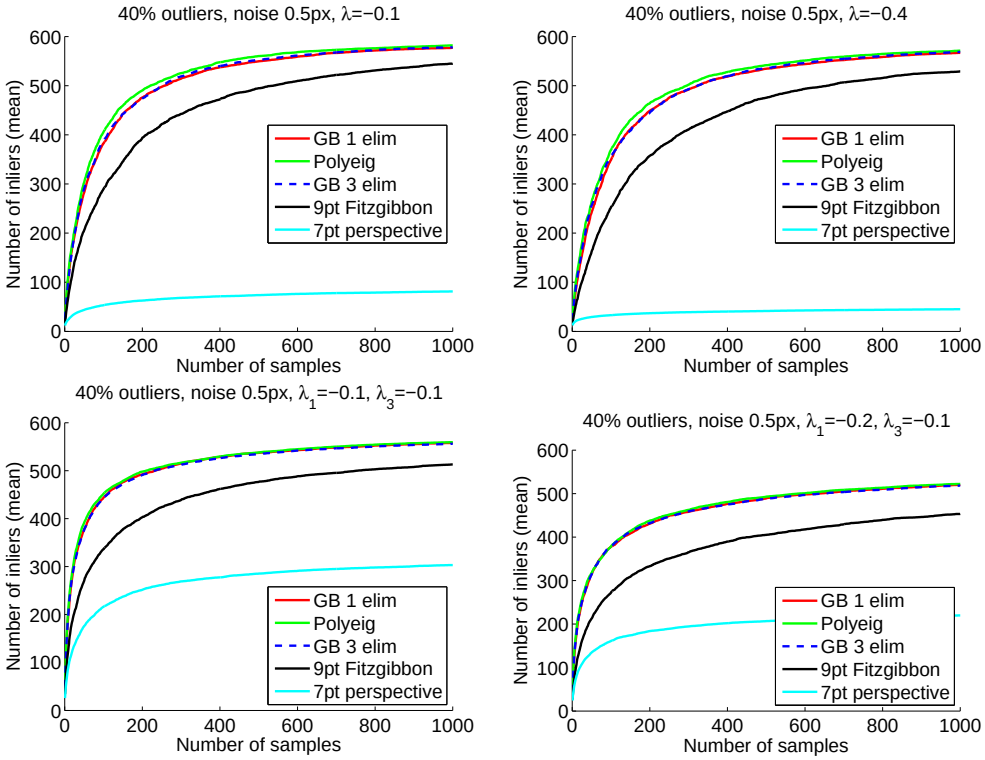

Figure 7.6: The mean value of the number of inliers over 1000 runs of the RANSAC as the function of the number of samples of the RANSAC. The top row shows the results for the division model with parameters $\lambda = -0.1$ (left) and $\lambda = -0.4$ (right) and the bottom row the results for the polynomial model, Eq. (7.51) with $\lambda_1 = -0.1$, $\lambda_2 = 0$, and $\lambda_3 = -0.1$ (left) and $\lambda_1 = -0.2$, $\lambda_2 = 0$, and $\lambda_3 = -0.1$ (right).

Therefore, in the following when talking about the proposed Gröbner basis solver we will mean the Gröbner basis solver for the "3 in 3" formulation.

In our next synthetic experiment, we studied how good the results from our solvers fit the distorted data. To do this we created four synthetic scenes using different radial distortion models. The first two scenes were created using the division model for which the solvers were intended, and the remaining two scenes were created using the three parameter distortion model

$$\mathbf{x}_d \sim \mathbf{x}_u(1 + \lambda_1 r_u + \lambda_2 r_u^2 + \lambda_3 r_u^3). \tag{7.51}$$

The numbers of mismatches in all these scenes were 400, which is 40% and the noise level was 0.5 pixels.

Figure 7.6 shows the mean value of the number of inliers over 1000 runs of the RANSAC as the function of the number of samples of the RANSAC. The top row shows the results for the division model with the ground truth $\lambda = -0.1$ (left) and $\lambda = -0.4$ (right), and the bottom row the results for the model (7.51) with ground truth $\lambda_1 = -0.1$, $\lambda_2 = 0$, and $\lambda_3 = -0.1$ (left) resp. with $\lambda_1 = -0.2$, $\lambda_2 = 0$, and $\lambda_3 = -0.1$ (right).

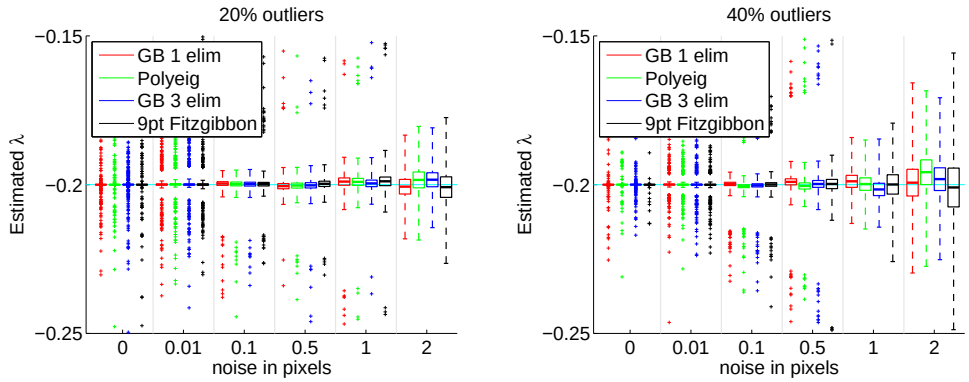

Figure 7.7: RANSAC: Estimated λ's as the function of noise, ground truth $\lambda_{true} = -0.2$ and (left) $inliers = 80\%$, (right) $inliers = 60\%$. Boxes contain values from 25% to 75% quantile. Noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

In this experiment we have compared both proposed minimal solvers for the "3 in 3" formulation, the Gröbner basis solver (red) and the polyeig solver (green), with the minimal Gröbner basis solver proposed in [89] (blue-dashed), the non-minimal solver [51] (black) and the standard perspective 7-pt algorithm [61] (cyan). It can be seen from Figure 7.6 that all minimal solvers give very similar and good results, and outperform the non-minimal solver [51] (black), and also the standard perspective 7-pt algorithm. It was expected that our algorithms would perform better than the 7-pt algorithm, because the 7-pt algorithm does not take into account radial distortion at all.

In the next two synthetic experiments we have studied the robustness of our algorithms to outliers and to increasing levels of Gaussian noise added to the distorted points. The ground truth radial distortion λ_{true} was -0.2 and the level of noise varied from 0 to 2 pixels. We have used two strategies for selecting the best λ among all generated radial distortion parameters, the RANSAC strategy and the kernel voting.

Figure 7.7 shows λ's computed by the four studied algorithms as a function of noise. In this case 500 lambdas were estimated using the RANSAC for each noise level and 80% (left) and 60% (right) of inliers. The results are presented by the Matlab function *boxplot* which shows values 25% to 75% quantile as a box with horizontal line at median. The crosses show data 1.5 times beyond the interquartile range.

The median values (from -0.1968 to -0.2036) for all 8-point algorithms as well as the 9-point algorithm (black) are very close to the ground truth value $\lambda_{true} = -0.2$ for all noise levels. The variances of the 9-point algorithm [51] (black) are slightly larger than the variances of the 8-point algorithms; however, this is not significant. More important is the fact that for larger noise levels the 9-point algorithm sometimes failed and the delivered result is far from the ground truth value inside the RANSAC loop. For the noise level 2 pixels, 5% of the results of the 9-point algorithm were far from the ground truth value.

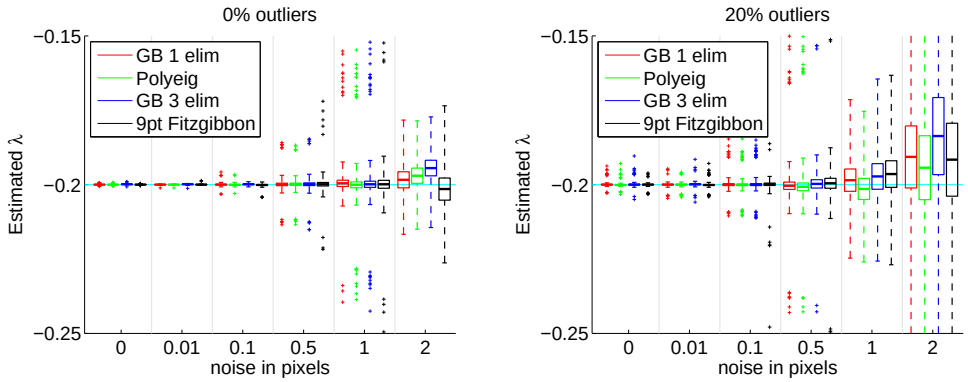

Figure 7.8: Kernel voting: Estimated λ's as the function of noise, ground truth $\lambda_{true} = -0.2$ and (left) $inliers = 100\%$, (right) $inliers = 80\%$. Boxes contain values from 25% to 75% quantile. Noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

We have made the same experiment using the kernel voting strategy for selecting the "best" λ. In this case the resulting λ was estimated using the following procedure:

1. Repeat K times (We use $K = 100$ here, but in many cases K from 30 to 50 is sufficient).
   a) Randomly choose 8 point (9 point) correspondences from given N correspondences.
   b) Find all $\lambda's$ of the minimal (non-minimal) solution to the autocalibration of radial distortion.
   c) Select real $\lambda's$ in a feasible interval, e.g., $-1 < \lambda < 1$ and the corresponding $\mathtt{F}$'s.
2. Use kernel voting on all selected $\lambda's$ to estimate the "best" radial distortion parameter.

Figure 7.8 shows estimated λ's using this procedure as a function of noise. Five hundred lambdas were estimated from one hundred ($K = 100$) 8 and 9-tuples of correspondences randomly drawn for each noise level and 100% (left) and 80% (right) of inliers. Here the results are worse for larger noise levels than in the RANSAC experiment (Figure 7.7); however, for smaller outlier contamination and lower noise levels the results are again very precise.

Since we assume that the distortion center is in the center of the image in all our algorithms, we have studied the robustness of our algorithms to a shift of the radial distortion center in the final synthetic experiment. The ground truth radial distortion λ_{true} was -0.1, the number of outliers 40% and the shift of the radial distortion center from the image center varied from 0% to 30% of the image size (for $1000 \times 1000px$ image from $0px$ to $300px$).

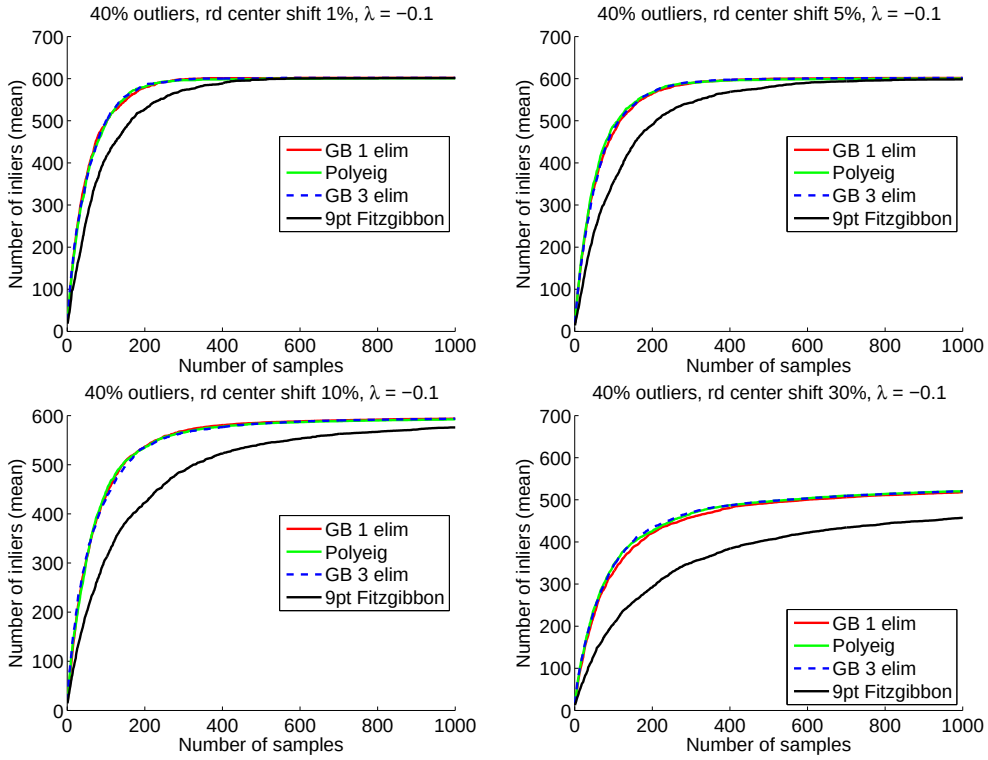

Figure 7.9: The mean value of the number of inliers over 1000 runs of the RANSAC as the function of the number of samples of the RANSAC for $\lambda_{true} = -0.1$, $outliers = 40\%$ and the radial distortion shift 1% of the image size (top left), 5% of the image size (top right), 10% of the image size (bottom left) and 30% of the image size (bottom right).

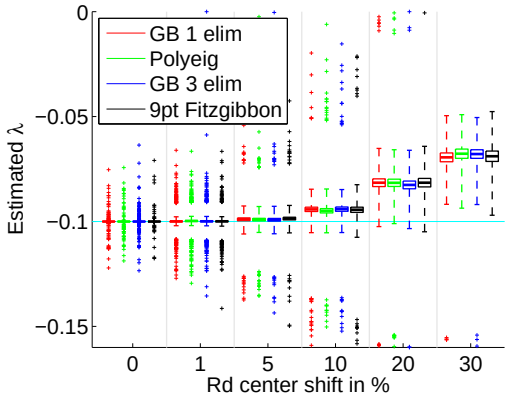

Figure 7.10: Estimated λ's as a function the radial distortion center shift, ground truth $\lambda_{true} = -0.1$, $outliers = 40\%$. Boxes contain values from 25% to 75% quantile.

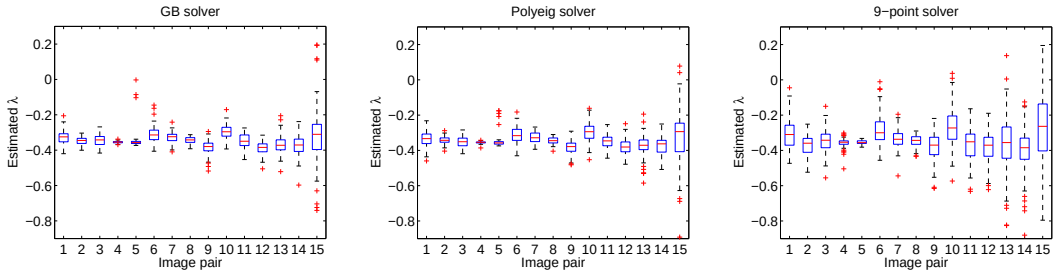

Figure 7.11: 100 λ's estimated using the RANSAC for the real image sequence consisting of 16 images taken by one camera. Blue boxes contain values from 25% to 75% quantile. (Left) The Gröbner basis algorithm, (middle) the polyeig algorithm, (right) the non-minimal 9-point algorithm [51].

Figure 7.9 shows the mean value of the number of inliers over 1000 runs of the RANSAC as a function of the number of samples of the RANSAC. The top left figure shows the result for the radial distortion center shifted by 1% of the image size from the image center, the top right figure for the 5% shift, the bottom left for the 10% shift and finally, the bottom right figure shows the result for the 30% shift.

We again show results for both proposed minimal solvers for the "3 in 3" formulation", the Gröbner basis solver (red) and the polyeig solver (green), the minimal Gröbner basis solver for the same formulation proposed in [89] (blue-dashed) and the non-minimal solver [51] (black). It can be seen from Figure 7.9 that all minimal solvers give very similar and good results for all shifts of the radial distortion center and outperform the non-minimal solver [51] (black). All minimal solvers were able to find the radial distortion parameter and the epipolar geometry which were consistent with all 600 inliers within 200 cycles of the RANSAC for all radial distortion shifts except for the 30% shift. For the 30% shift the results were still useful.

Figure 7.10 shows λ's computed by the four studied algorithms as a function of the radial distortion center shift. One thousand lambdas were estimated using the RANSAC for each radial distortion center shift from 0% to 30% of the image size and 40% of outliers. The results are presented by the Matlab function *boxplot*. The median values of the estimated λ's are quite close to the ground truth value $\lambda_{true} = -0.1$ for all radial distortion shifts.

These RANSAC experiments (Figure 7.9 and Figure 7.10) show that even large shifts of the radial distortion center from the image center do not make problems in finding quite precise distortion parameters and good inliers sets.

Tests on real images

We have tested both our algorithms for the "3 in 3" formulation of the "8-point radial distortion problem" on several sequences of real images. The first image sequence was a sequence with relatively large distortions. These images were obtained as 75% cutouts from 180° angle of view fish-eye images. We use cutouts since the one parameter division model (7.21) is not able to handle 180° images. Tentative point correspondences were found by the wide base-line

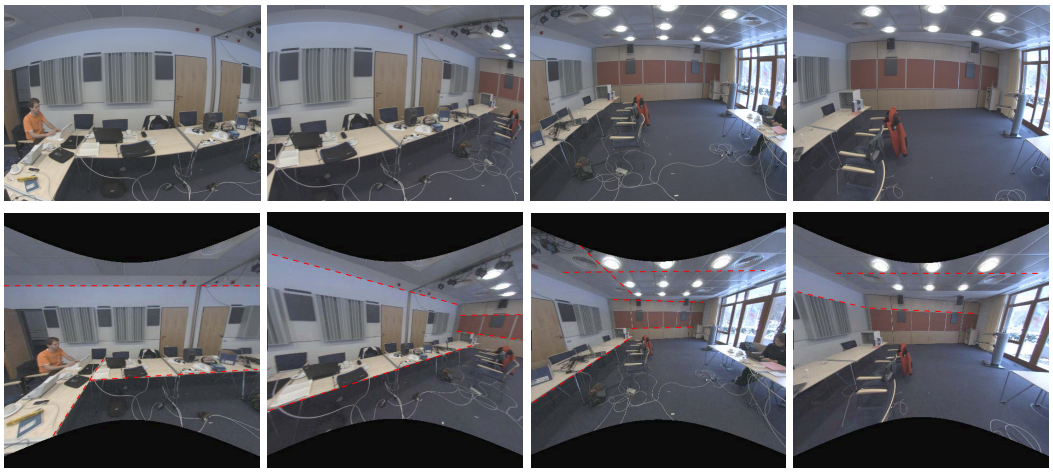

Figure 7.12: Examples of images from the processed image sequence. (Top) Input images with significant radial distortion. (Bottom) Corrected images using the median values from Figure 7.11 (middle).

matching algorithm [103]. They contained correct as well as incorrect matches. Note that for such distorted images the number of incorrect matches is relatively large (usually between 30% and 70%). For all images we used the center of the image as the radial distortion center.

In our first experiment with this image sequence, we have studied the stability of the estimated radial distortion parameter. To do this we have estimated λ's for the image sequence taken by a single camera. Figure 7.11 shows λ's estimated for each consecutive pair of images using our Gröbner basis one-elimination algorithm (left), our polyeig algorithm (middle) and the non-minimal 9-point algorithm [51] (right). In this case 100 λ's were estimated using the RANSAC and the results are presented using the Matlab function *boxplot*.

It can be seen that both minimal solvers give very similar and relatively stable results. The median values (red lines in figures) for these solvers varied from -0.3809 to -0.2935. The variances of the 9-point solver, Figure 7.11 (right), were considerably larger than the variances of both minimal solvers, Figure 7.11 (left and middle). The mean range of the upper and the lower quartile (blue box in Figure 7.11) of estimated λ's was for the non-minimal 9-point algorithm 0.1021, for the polyeig algorithm 0.0521 and for the Gröbner basis algorithm 0.0481 which is approximately two times lower than for the non-minimal case. It is also notable that the number of outliers in this image sequence was not negligible. It varied from 35% to 60%.

Figure 7.12 (top) shows examples of images from the processed image sequence. Corrected images (bottom) were obtained using the median values from λ's computed using our polyeig algorithm together with the RANSAC (Figure 7.11 (middle)). It can be seen that straight lines in the real world are really straight in corrected images.

In the second experiment we have tested the kernel voting strategy for selecting the "best" λ on the same image sequence as it was used in the previous experiment. Figure 7.13 shows results of the kernel voting made on λ's obtained from 500 runs of our polyeig algorithm for image pairs $4-5,\ 5-6,\ 8-9$ and $11-12$. Red vertical lines show estimated λ's using the

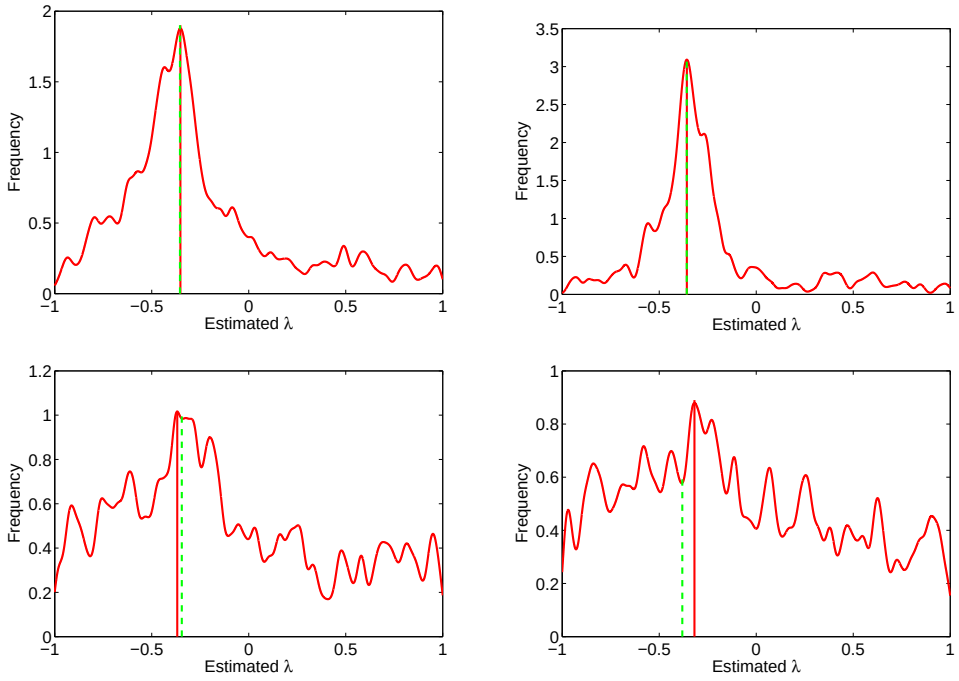

Figure 7.13: Result of kernel voting strategy on image pairs $4-5,\ 5-6,\ 8-9$ and $12-13$ from image sequence from Figure 7.11. Red vertical lines show estimated λ's using kernel voting method and dashed green lines show λ's computed as median values from 100 results of the RANSAC (Figure 7.11 (middle)). In both cases our polyeig algorithm was used to estimate λ.

kernel voting method, and dashed green lines show λ's computed as median values from 100 results of the RANSAC (Figure 7.11 (middle)). In both cases the polyeig algorithm was used to estimate λ.

It can be seen that both methods, the kernel voting and the RANSAC, behave similarly when estimating radial distortion parameter. On image pairs $4-5$ and $5-6$, where the RANSAC method returns very stable results, the kernel voting strategy produces histograms with significant peaks. In this case the results from the RANSAC (green dashed line) are very close to the results from kernel voting (red line). On the other hand, for image pairs $8-9$ and $11-12$, where the variations of the results from the RANSAC are larger, the kernel voting strategy doesn't produce such significant peaks. Still, the results of the kernel voting corresponding to the highest peaks (red lines) are close to the results from the RANSAC (green dashed lines). However, we suggest to use RANSAC or combination of RANSAC with weighted kernel voting [27] instead of the standard kernel voting in cases with higher noise levels and outlier contaminations like it was in this case.

Finally, we have performed the same experiments on several image pairs taken by standard consumer camera at $28mm$ focal length. Figure 7.14 (right) shows λ's estimated for each image

Figure 7.14: (Left) Example of input image, (middle) Corrected image, (right) 100 λ's estimated using the RANSAC for each from 10 image pairs taken by one camera. Blue boxes contain values from 25% to 75% quantile.

pair using our Gröbner basis algorithm. Again 100 λ's were estimated using the RANSAC and the results are presented using the Matlab function *boxplot*. Here the results were even more stable than the results of the same experiment for the fish-eye camera (Figure 7.11). The mean values varied form -0.0243 to -0.0476. These values of λ are quite small what corresponds to a quite small radial distortion of the input images as it can be seen on the example of the input image in Figure 7.14 (left). However, this radial distortion was still large enough to cause problems in 3D reconstruction, and without its correction standard SfM pipelines were not able to give satisfactory results. The example of corrected image can be seen in Figure 7.14 (middle).

Computation complexity

Our current MATLAB implementation of the Gröbner basis solver for the "3 in 3" formulation, which was generated using our automatic generator, needs to perform G-J elimination of a 32×48 matrix and to compute eigenvalues of a 16×16 matrix and runs about $640\mu s$ on a Intel i7 Q720 notebook. This MATLAB implementation uses C++ MEX-function for G-J elimination. The running time of the polyeig solver is about $685\mu s$. This solver needs to compute the inverse of one 10×10 matrix and eigenvalues of a 29×29 matrix. For comparison, our MATLAB implementation of Fitzgibbon's non-minimal algorithm runs about $720\mu s$ and the original implementation of Hongdong Li's algorithm [97] based on MATLAB Symbolic-Math Toolbox runs about $860ms = 860000\mu s$.

Our C++ version of the Gröbner basis solver for the "3 in 3" formulation runs on the same hardware about $120\mu s$.

7.1.4 9-point two different radial distortions problem

In this subsection we describe a solution to the second minimal radial distortion problem, the problem of simultaneous estimation of the fundamental matrix and two radial distortion parameters for two uncalibrated cameras with different distortions from nine image point correspondences. This problem is useful in applications where we have images of one scene taken by two

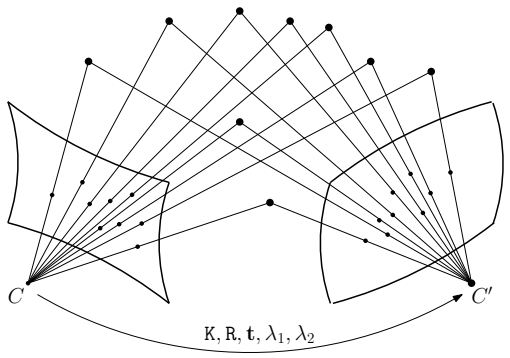

Figure 7.15: Illustration of the 9-point two different radial distortions problem.

different cameras, for example a standard digital camera and a camera with some wide angle lens or an omnidirectional camera.

Previous solutions

Solving for the fundamental matrix under different radial distortions was first studied in [11], where the non-minimal algorithm based on 15 point correspondences was given for a pair of uncalibrated cameras.

The first minimal solution to this problem was proposed in our paper [91]. Unfortunately, this solution was not very practical and efficient since Maple and exact arithmetic was used due to numerical problems of the floating point solver.

The first efficient floating point minimal solver for this problem we proposed in [25, 87]. Here, methods for improving the numerical stability of Gröbner basis solvers presented in paper [33] were used to obtain a robust and a numerical stable solver. The solver consists of LU decomposition of a 393×390 matrix and the eigenvalue computations of a 24×24 matrix. The problem of this solver is that it was created manually and the set of monomials which should be used to multiply initial polynomial equations was selected after detailed studying of the problem which together with special manipulations with these polynomials make the final solver quite complicated and almost not reimplementable.

We have proposed another non-minimal solver [87] for this relative pose problem for two cameras with different distortions. The solver uses 12 point correspondences to estimate the fundamental matrix and two radial distortion parameters and is based on the generalized eigenvalue formulation of the initial polynomial equations.

Next we will propose a new minimal solution to this "9-point two different radial distortions problem" which is based on the specialized Gröbner basis method presented in Section 4.4. This solution is robust and stable and compared to our Gröbner basis solution presented in papers [25, 87] much simpler and easier to implement. Moreover, this solution can be obtained using our automatic generator of Gröbner basis solvers presented in Chapter 6.

Next we provide the formulation of the problem.

Problem formulation

We want to estimate radial distortion parameters and relative pose of two different uncalibrated cameras from the minimal number of image point correspondences in two views. We assume the one-parameter division model (7.19).

It is known that to get solutions to this minimal problem for uncalibrated cameras with different radial distortions λ_1 and λ_2 in each image, we have to use the epipolar constraint for 9 point correspondences

$$\mathbf{x}_{u_i}^{\top}(\lambda_1)\,\mathtt{F}\,\mathbf{x}'_{u_i}(\lambda_2) = 0, \quad i = 1,\ldots,9 \tag{7.52}$$

and the singularity of the fundamental matrix $\mathtt{F}$

$$\det(\mathtt{F}) = 0. \tag{7.53}$$

Assuming $f_{3,3} \neq 0$ we can set $f_{3,3} = 1$ and we obtain 10 equations in 10 unknowns.

This is quite a complex system of polynomial equations. Therefore we first simplify the system by eliminating some variables. This can be done similarly as in the previous "8-point radial distortion problem". However, in this case the first proposed elimination procedure results in a system of eight equations in eight unknowns which is not very suitable for the proposed methods for solving systems of polynomial equations. Therefore, we propose here a solution to of formulation which results from the second elimination procedure.

Eliminating variables - 4 equations in 4 unknowns

Assuming $f_{3,3} = 1$, the epipolar constraint (7.52) gives 9 equations in 16 monomials $(f_{3,1}\lambda_1,$ $f_{3,2}\lambda_1, f_{1,3}\lambda_2, f_{2,3}\lambda_2, \lambda_1\lambda_2, f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2, 1)$ and 10 variables $(f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)$.

Among them, 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 9 equations from the epipolar constraint, we can use these equations to eliminate 6 variables, the four variables $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}$ that appear in one monomial only (and can be straightforwardly eliminated), and two of the variables that appear in two monomials. We decided to eliminate $f_{1,3}$ and $f_{2,3}$.

We reorder the monomials contained in the 9 equations and put the monomials containing $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}$ and $f_{2,3}$ at the beginning. The reordered monomial vector becomes $\mathbf{X} = (f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda_2, f_{1,3}, f_{2,3}\lambda_2, f_{2,3}, f_{3,1}\lambda_1, f_{3,2}\lambda_1, \lambda_1\lambda_2, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2, 1)^T$.

Then, we rewrite the 9 equations from the epipolar constraint in the matrix form $\mathtt{M}\,\mathbf{X} = \mathbf{0}$, where $\mathtt{M}$ is the coefficient matrix and $\mathbf{X}$ is this reordered monomial vector. After performing G-J elimination on the matrix $\mathtt{M}$, we obtain 9 equations of the form

$$p_i = LT(p_i) + g_i(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) = 0, \tag{7.54}$$

where $LT(p_i)$ are monomials $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda_2, f_{1,3}, f_{2,3}\lambda_2,$ $f_{2,3}$ and $f_{3,1}\lambda_1$ for $i = 1,\ldots,9$ and $g_i(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)$ are 2^{nd}order polynomials in four variables $f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$.

This means that we can express the 6 variables, $f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}$ as functions of the other four variables $f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$.

$$\begin{aligned} f_{1,1} &= -g_1(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2), && (7.55)\\ f_{1,2} &= -g_2(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2), && (7.56)\\ f_{1,3} &= -g_6(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2), && (7.57)\\ f_{2,1} &= -g_3(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2), && (7.58)\\ f_{2,2} &= -g_4(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2), && (7.59)\\ f_{2,3} &= -g_8(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2). && (7.60) \end{aligned}$$

Substituting these expressions into the other three equations from the epipolar constraint (7.52) and also into the singularity constraint (7.53) for F gives 4 polynomial equations in 4 unknowns (one of 2^{nd} degree, two of 3^{rd} degree and one of 5^{th} degree)

$$\begin{aligned} \lambda_2(-g_6(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)) + g_5(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) &= 0, && (7.61)\\ \lambda_2(-g_8(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)) + g_7(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) &= 0, && (7.62)\\ f_{3,1}\lambda_1 + g_9(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) &= 0, && (7.63) \end{aligned}$$

$$\det \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & -g_8 \\ f_{3,1} & f_{3,2} & 1 \end{pmatrix} = 0. \quad (7.64)$$

Gröbner basis solution

Next we describe how to create an efficient Gröbner basis solver for this formulation of the "9-point two different radial distortions" problem. We use the specialized Gröbner basis method presented in Section 4.4 and the single elimination strategy for generating polynomials from the ideal to construct a solver by our automatic generator of Gröbner basis solvers presented in Chapter 6.

Identification of basic parameters

In the first step we identify the basic parameters of the system of four equations (7.61)-(7.64) in four unknowns $f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$.

It was found that this system has 24 solutions using Macaulay 2 and random coefficients from some finite prime field.

To create the multiplication matrix, we use the graded reverse lexicographic ordering with $f_{3,1} > f_{3,2} > \lambda_1 > \lambda_2$. With this ordering, we get the basis $B = \{[1], [f_{3,1}], [f_{3,1}^2], [f_{3,1}f_{3,2}], [f_{3,1}f_{3,2}^2], [f_{3,1}\lambda_2], [f_{3,1}\lambda_2^2], [f_{3,2}], [f_{3,2}^2], [f_{3,2}^3], [f_{3,2}^2\lambda_2], [f_{3,2}\lambda_1], [f_{3,2}\lambda_1^2], [f_{3,2}\lambda_2], [f_{3,2}\lambda_2^2], [\lambda_1], [\lambda_1^2], [\lambda_1^3], [\lambda_1^2\lambda_2], [\lambda_1\lambda_2], [\lambda_2], [\lambda_2^2], [\lambda_2^3], [\lambda_2^4]\}$ of the quotient ring $A = \mathbb{C}[f_{3,1}, f_{3,2}, \lambda_1, \lambda_2]/I$.

Construction of the multiplication matrix

Next we find an "elimination template" for constructing the multiplication matrix, i.e. an admissible Elimination trace. We have tested all four unknowns $f_{3,1}, f_{3,2}, \lambda_1$ and λ_2 and found that the best choice for the "action variable", which leads to the simplest solution, is in this case $f_{3,1}$. Hence we construct the multiplication matrix $\mathtt{M}_{f_{3,1}}$.

Using the method described in Section 4.4 and the single elimination strategy for generating polynomials from the ideal we have found that to obtaining all necessary polynomials q_i (4.115) for creating the multiplication matrix $\mathtt{M}_{f_{3,1}}$ we have to generate all monomial multiples of the initial four polynomial equations up to the total degree eight.

This leads to 497 polynomials in 495 monomials. We remove unnecessary polynomials using Algorithm 10 presented in Section 4.4.3, to get much smaller system of 179 polynomial equations in 495 monomials.

However, as described in 4.4.3 some from these 495 monomials do not have impact on the form of polynomials q_i (4.115) necessary for constructing the multiplication matrix, i.e. on the coefficients in the multiplication matrix $\mathtt{M}_{f_{3,1}}$. We remove columns corresponding to such "unnecessary monomials" in the matrix representation of these 179 polynomials and obtain 179×203 matrix. After performing G-J elimination of this matrix we obtain all polynomials which we need for constructing the multiplication matrix $\mathtt{M}_{f_{3,1}}$.

Online Gröbner basis solver

The final online solver for solving all systems of polynomial equations in the form of four equations (7.61)-(7.64) in four unknowns, i.e. for solving the "9-point two different radial distortions problem" then does only one G-J elimination of the 179×203 coefficient matrix. This matrix contains coefficients which arise from concrete image measurements. After the G-J elimination of this matrix, the multiplication matrix $\mathtt{M}_{f_{3,1}}$ is created from rows which correspond to the polynomials q_i (4.115) necessary for its construction. The eigenvectors of the multiplication matrix $\mathtt{M}_{f_{3,1}}$ give solutions for $f_{3,1}, f_{3,2}, \lambda_1$ and λ_2. Finally, backsubstituting these solutions to equations (7.55)-(7.60) gives solutions for the remaining elements of the fundamental matrix $\mathtt{F}$.

This 9-point two different distortions solver is much simpler than the solver presented in [35] which consist of LU decomposition of much larger matrix of the size 393×390 Moreover our solver can be obtained using the automatic generator of Gröbner basis solvers. Next we describe the input to our automatic generator which results in this efficient online solver for the "9-point different distortion problem".

Input to the automatic generator

The input to our automatic generator for the "9-point two different radial distortions" problem is presented in Figure (7.16). With this input the automatic generator automatically performs all steps of the offline phase and as an output gives efficient online solver for solving four equation (7.61)-(7.64) in four unknowns. Note that to obtain the final solver for the whole problem the backubstitution of the obtained solutions needs to be add to this output.

```
g1 = transpose(gbs_Vector('g1', 7));
g2 = transpose(gbs_Vector('g2', 7));
g3 = transpose(gbs_Vector('g3', 7));
g4 = transpose(gbs_Vector('g4', 7));
g5 = transpose(gbs_Vector('g5', 7));
g6 = transpose(gbs_Vector('g6', 7));
g7 = transpose(gbs_Vector('g7', 7));
g8 = transpose(gbs_Vector('g8', 7));
g9 = transpose(gbs_Vector('g9', 7));

syms f31 f32 k1 k2;

mon = [k1*f32,k1*k2,f31,f32,k1,k2,1]';

% parametrization of the fundamental matrix with four unknowns
f11 = -g1*mon;
f12 = -g2*mon;
f13 = -g6*mon;
f21 = -g3*mon;
f22 = -g4*mon;
f23 = -g8*mon;

% fundamental matrix
F = [f11 f12 f13; f21 f22 f23; f31 f32 1];

% four polynomial equations
eq(1) = -(k2*g6)*mon + g5*mon;
eq(2) = -(k2*g8)*mon + g7*mon;
eq(3) = det(F);
eq(4) = k1*f31+g9*mon;

% four unknowns
unknown = {'f31' 'f32' 'k1' 'k2'};

% known parameters
vars = transpose([g1(:); g2(:); g3(:); g4(:); g5(:); g6(:); g7(:);
g8(:); g9(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% call code generator
[res export] = gbs_CreateCode('ku9pt', eq, known, unknown);
```

Figure 7.16: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the solver for the "9-point different radial distortion problem".

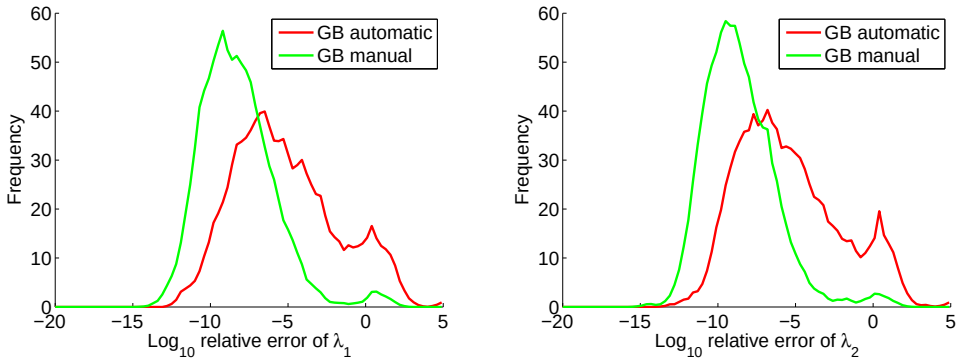

Figure 7.17: Log_{10} relative error of the radial distortion parameter λ_1 (left) and λ_2 (right) obtained by selecting the real roots closest to the ground truth values $\lambda_{1_{th}} = -0.2$ and $\lambda_{2_{th}} = -0.4$.

Experiments

We have tested the presented Gröbner basis solver for the "9-point two different radial distortion problem" on both synthetic data (with various levels of noise, outliers, radial distortions) and on real datasets and compared it with the manually created and numerically optimized Gröbner basis algorithm presented in [35, 87].

Synthetic data sets

Similar to the "8-point radial distortion problem", we use synthetically generated ground-truth 3D scenes generated using the following procedure:

1. Generate a 3D scene consisting of N $(= 1000)$ random points within a 3D cube.
2. Project $M\%$ of the points on image planes of the two displaced cameras with feasible position and orientation. These are matches. Generate $(100-M)\%$ random points in both images. These are mismatches. Altogether, they become undistorted correspondences.
3. Apply different radial distortions to the undistorted correspondences in both images to generate noiseless distorted points.
4. Add Gaussian noise of standard deviation σ to the distorted points assuming a 1000×1000 pixel image. By noise $1px$ we will mean Gaussian noise with standard deviation $\sigma = \frac{1}{3}px$ in all our experiments.

In the first synthetic experiment we have studied the behavior of the presented Gröbner basis solver on noise free data to check its numerical stability. In this experiment 1000 random scenes and feasible camera poses for two cameras with different radial distortions were generated.

We have focused on radial distortion parameter estimation because once we have a good radial distortion estimate, the fundamental matrix is usually good or can be recomputed using well known algorithms for perspective cameras [61, 112, 132].

Figure 7.17 (left) shows the log_{10} relative error of the radial distortion parameter λ_1 for the left camera and Figure 7.17 (right) the log_{10} relative error of λ_2 for the right camera. These

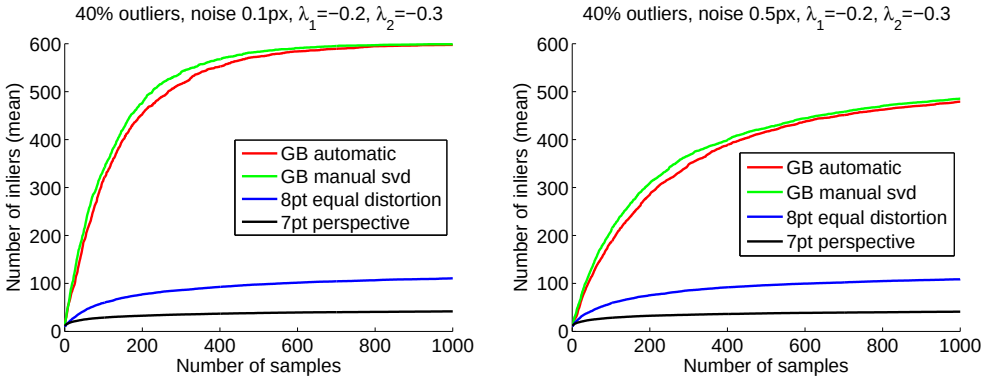

Figure 7.18: The mean value of the number of inliers over 1000 runs of the RANSAC as the function of the number of samples of the RANSAC. The left figure shows the results for 40% of mismatches and the noise level 0.1 pixels and the right figure for 40% of mismatches and the noise level 0.5 pixels, where noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$. The ground truth radial distortion parameters were $\lambda_{1_th} = -0.2$ and $\lambda_{2_th} = -0.3$.

results were obtained by selecting the real roots closest to the ground truth values $\lambda_{1_{th}} = -0.2$ and $\lambda_{2_{th}} = -0.4$.

In this case we have compared two solvers for the "9-point two different radial distortion problem". The first solver (red) is the proposed solver created using the presented specialized Gröbner basis method and our automatic generator presented in Chapter 6. The second solver for comparison is the manually created and numerically optimized solver presented in [35, 87] (green).

It can be seen that the manually created solver (green) provides more accurate results than our Gröbner basis solvers created using the presented method and the automatic generator. It is because the manually created solver uses special methods for improving the numerical stability of Gröbner basis solvers presented in paper [34]. The proposed automatically generated solver (red) however behaves quite stable and provides comparable results to the numerically optimized solver (green) and we will show that it provides good results also in real applications. Moreover, the proposed automatically generated solver is smaller and faster than the numerically optimized solver [35, 87] and can be further numerically optimized with the similar methods as [35, 87].

In the next two synthetic experiments we have studied the robustness of our algorithm to outliers and to increasing levels of Gaussian noise added to the distorted points. The ground truth radial distortion parameters were $\lambda_{1_{true}} = -0.2$ and $\lambda_{2_{true}} = -0.3$.

The first experiment in Figure 7.18 shows the mean value of the number of inliers over 1000 runs of the RANSAC as the function of the number of samples of the RANSAC. The number of mismatches in all scenes was 400 which is 40% and the noise level was $0.1px$ Figure 7.18 (left) and $0.5px$ Figure 7.18 (right).

In this experiment we have compared both considered minimal solvers for the "9-point different radial distortion problem", i.e. the proposed automatic Gröbner basis solver (red) and the numerically optimized Gröbner basis solver [35, 87] (green), with the "8-point equal ra-

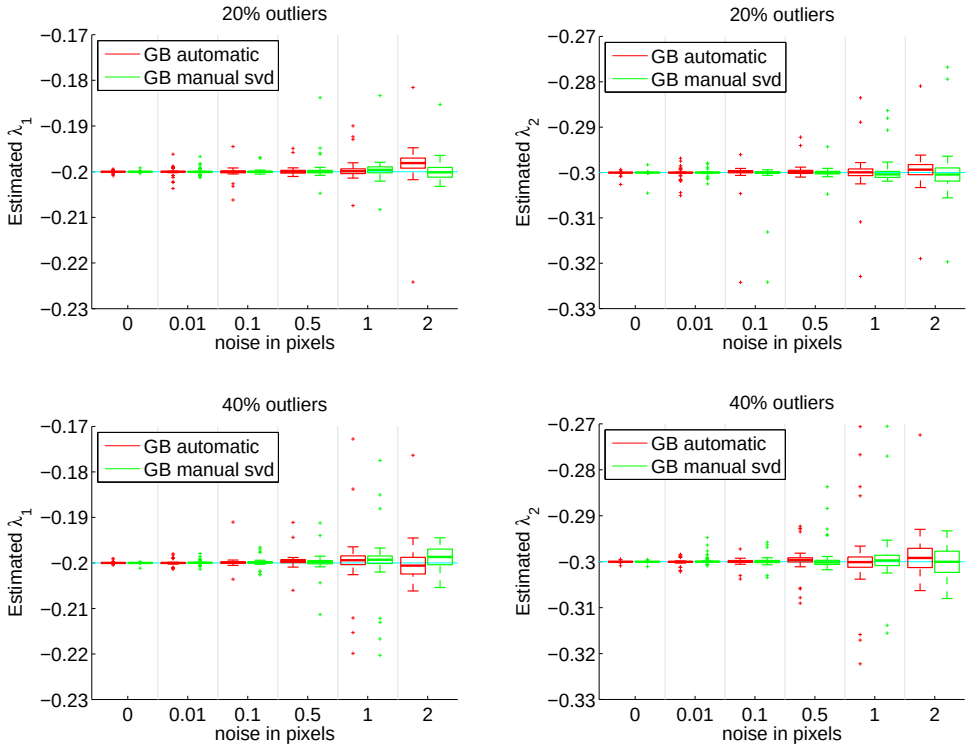

Figure 7.19: RANSAC: Estimated λ's as the function of noise, ground truth $\lambda_{1_{true}} = -0.2$ (left) and $\lambda_{2_{true}} = -0.3$ (right); and $inliers = 80\%$ (top) and $inliers = 60\%$ (bottom). Boxes contain values from 25% to 75% quantile. Noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

dial distortion" polynomial eigenvalue solver presented in Section 7.1.3 (blue) and the standard perspective 7-pt algorithm [61] (black).

It can be seen from Figure 7.18 that both"9-point different radial distortion" solvers give very similar and good results and, as expected, significantly outperform the "8-point equal radial distortion" solver (blue) and the standard perspective 7-pt algorithm (black), which does not take into account radial distortion at all.

In the last synthetic experiment we have studied the robustness of our algorithm to increasing levels of Gaussian noise added to the distorted points. In this case the level of noise varied from 0 to 2 pixels and the ground truth radial distortion parameters were $\lambda_{1_{true}} = -0.2$ and $\lambda_{2_{true}} = -0.3$.

Figure 7.19 shows λ's computed by the two minimal 9-point solvers, the proposed automatic Gröbner basis solver (red) and the numerically optimized manually created Gröbner basis solver [35, 87] (green), as a function of noise. In this case 100 lambdas were estimated using the RANSAC for each noise level and 80% (top) and 60% (bottom) of inliers. The results are presented by the Matlab function *boxplot* which shows values 25% to 75% quantile as a box with horizontal line at median. The crosses show data beyond 1.5 times the interquartile range.

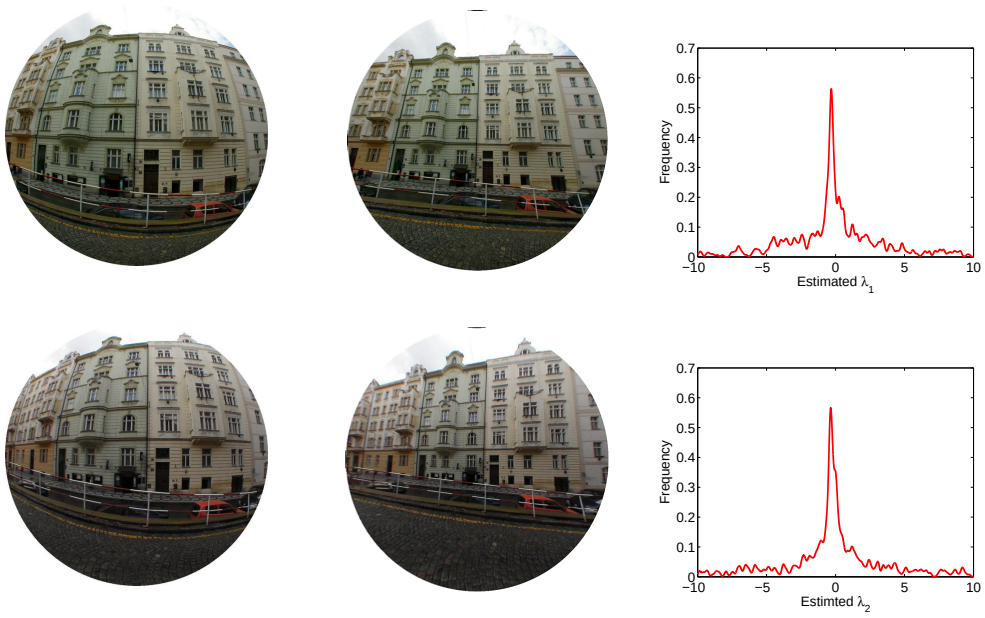

Figure 7.20: Real data, 60% cutouts from omnidirectional images. (Left) Input images with different radial distortions for camera 1 (top) and camera 2 (bottom). (Center) Corrected images. (Right) Distribution of real roots obtained by kernel voting. Estimated $\lambda_1 = -0.31000$ and $\lambda_2 = -0.35125$.

The left figures show the results for the left camera and $\lambda_{1_{true}} = -0.2$ and the right figures for the right camera and $\lambda_{2_{true}} = -0.3$

The median values for both 9-point algorithms are very close to the ground truth values for all noise levels. The variances of the proposed automatic Gröbner basis solver (red) are similar to the variances of the numerically optimized Gröbner basis solver [35, 87].

Tests on real images

We have tested our 9-point minimal algorithm for uncalibrated cameras with different radial distortions on several different sets of images. In the first experiment the input images with different relatively large distortions in each image, Figure 7.20 (left), were obtained as 60% cutouts from fish-eye images taken with two different cameras with different radial distortions. Tentative point matches were then found by the wide base-line matching algorithm [103]. They contained correct as well as incorrect matches. Distortion parameters λ_1 and λ_2 were estimated using our algorithm for uncalibrated cameras with different radial distortions and the kernel voting method for 200 samples. The input (left) and the corrected (center) images are presented in Figure 7.20. Figure 7.20 (right) shows the distribution of the real roots for these images, from which $\lambda_1 = -0.3100$ and $\lambda_2 = -0.35125$ were estimated as the argument of the maximum. The peaks from kernel voting are sharp and the λ's are estimated accurately.

In the second experiment we tested our algorithm on images with significantly different dis-

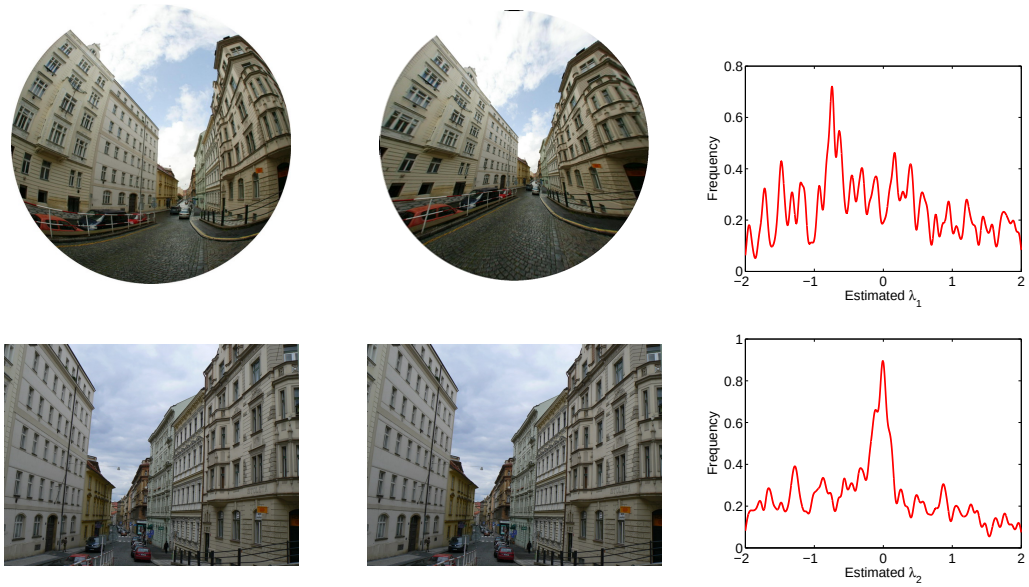

Figure 7.21: (Left) Input images with different radial distortions (top) 66% cutout from omnidirectional image and (bottom) image taken with a standard perspective camera with very mild distortion. (Center) Corrected images. (Right) Distribution of real roots obtained by kernel voting. Estimated $\lambda_1 = -0.743875$ and $\lambda_2 = -0.006500$.

tortions. The left image Figure 7.21 (top), was obtained as a 66% cutout from a fish-eye image and the right image Figure 7.21 (bottom) was taken with a standard perspective camera. Since these images had a rather large difference in radial distortion, the tentative point correspondences contained a larger number of mismatches. Distortion parameters λ_1 and λ_2 were again estimated using our algorithm for uncalibrated cameras with different radial distortions and the kernel voting method. The input (left) and the corrected (center) images are presented in Figure 7.21. Figure 7.21 (right) shows the distribution of real roots for these images from which $\lambda_1 = -0.743875$ and $\lambda_2 = -0.006500$ were estimated. As can be seen the peaks obtained by the kernel voting are not so sharp but still sufficient to get good estimates of λ's even from only 200 samples.

Computation complexity

Our MATLAB+MEX implementation of the Gröbner basis solver needs to perform G-J elimination of a 179×203 matrix and to compute eigenvalues of a 24×24 matrix and runs about $8.9ms$ on a Intel i7 Q720 notebook. The running time of the manually created solver [35, 87] is about $18.8ms$. This solver needs to perform LU decomposition of a 393×390 matrix and to compute eigenvalues of a 24×24 matrix. The C++ version of our Gröbner basis solver runs on the same hardware about $2.56ms$.

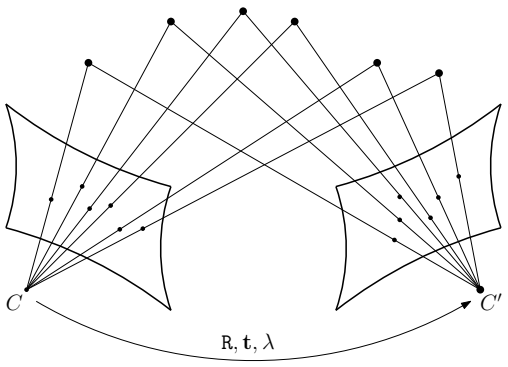

Figure 7.22: Illustration of the 6-point calibrated radial distortion problem.

7.1.5 6-point calibrated radial distortion problem

The final minimal radial distortion problem which we have solved using the specialized Gröbner basis method presented in Section 4.4 is the problem of simultaneous estimation of the essential matrix and one common radial distortion parameter for calibrated camera and six image point correspondences in two views.

This problem is not so practical as the previous two radial distortion problems. The first reason is that this problem results in 52 solutions, and therefore needs to perform eigenvalue computation of a 52×52 matrix which is quite time-consuming. Moreover, all these 52 solutions have to be tested for example in some RANSAC paradigm to select the "best solution", the solution which is consistent with most of the measurements. The second reason is that usually when the camera is internally calibrated it is calibrated together with the radial distortion.

However, sometimes we have information about the focal length from EXIF, so we can consider our camera as calibrated. In such situations we usually do not have information about the radial distortion and therefore the presented "6-point calibrated radial distortion" algorithm may be useful. Moreover, it is interesting to study this problem also from the theoretical point of view, since it results in a quite complicated system of polynomial equations and therefore shows the ability of the proposed specialized Gröbner basis method to create efficient solvers also for such complicated systems.

Previous solutions

The first minimal solution to the " 6-point calibrated radial distortion problem" was proposed in our paper [91]. Unfortunately, this solution, as the solution to the "9-point two different radial distortions problem" proposed in the same paper, was not practical and efficient since Maple and exact arithmetic was used to solve the problem.

An efficient floating point minimal solver for this problem was proposed in our papers [35, 87]. Methods for improving the numerical stability of Gröbner basis solvers presented in paper [33] were used to obtain the solver which consists of LU decomposition of a 320×363 matrix and eigenvalue computation of a 52×52 matrix. The solver is still quite complicated. It

was constructed manually after detailed studying of the problem and fine tuning of the polynomials.

We next propose a new minimal solution to the "6-point calibrated radial distortion problem" based on the specialized Gröbner basis method presented in Section 4.4. This solution is robust and stable and can be obtained using our automatic generator of Gröbner basis solvers.

Next we provide the problem formulation.

Problem formulation

To solve the minimal radial distortion problem for calibrated cameras, we make use of the epipolar constraint for 6 point correspondences

$$\mathbf{x}_{u_i}^{\top}(\lambda)\,\mathtt{E}\,\mathbf{x}'_{u_i}(\lambda) = 0, \quad i = 1, \ldots, 6, \tag{7.65}$$

the singularity of the essential matrix $\mathtt{E}$

$$\det(\mathtt{E}) = 0, \tag{7.66}$$

and the trace constraint, which says that two singular values of the essential matrix are equal

$$2\left(\mathtt{E}\mathtt{E}^T\right)\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^T)\mathtt{E} = \mathtt{0}. \tag{7.67}$$

Again assuming $e_{3,3} \neq 0$, we can set $e_{3,3} = 1$ and obtain 16 equations in 9 unknowns.

This is a quite large and complex system of polynomial equations. Using a similar method as it was used in both uncalibrated cases and assuming that some elements of the essential matrix are non-zero, these equations can be rewritten to 11 polynomial equations in 4 unknowns (one of 3^{rd} degree, four of 5^{th} degree and six of 6^{th} degree). Next we describe this elimination procedure in more detail.

Eliminating variables - 11 equations in 4 unknowns

The epipolar constraint (7.65) gives 6 equations in 15 monomials $(\lambda e_{1,3}, \lambda e_{2,3}, \lambda e_{3,1}, \lambda e_{3,2}, \lambda^2, e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, \lambda, 1)$ and 9 variables $(e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, \lambda)$.

Using the same elimination method as in both uncalibrated cases presented in Section 7.1.3 and 7.1.4, we can eliminate 5 of these 9 variables. All these variables can be eliminated simultaneously.

We have four variables which appear in one monomial only $(e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2})$ and four variables which appear in two monomials $(e_{1,3}, e_{2,3}, e_{3,1}, e_{3,2})$. Since we have six equations, of which each contains all 15 monomials, we can eliminate five from the nine variables. We have selected the first four variables $e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}$ that appear in one monomial only and the fifth variable as $e_{1,3}$.

We reorder the monomials contained in the 6 equations putting monomials containing $e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}$ and $e_{1,3}$ at the beginning of the monomial vector. The reordered monomial vector will be $\mathbf{X} = (e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}, e_{1,3}\lambda, e_{1,3}, e_{2,3}\lambda, e_{3,1}\lambda, e_{3,2}\lambda, \lambda^2, e_{2,3}, e_{3,1}, e_{3,2}, \lambda, 1)^T$.

We rewrite 6 equations from the epipolar constraint to the matrix form $\mathtt{M}\mathbf{X} = \mathbf{0}$. After performing G-J elimination on the matrix $\mathtt{M}$, we obtain 6 equations of the form

$$p_i = LT(p_i) + g_i(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) = 0, \tag{7.68}$$

where $LT(p_i)$ are monomials $e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}, e_{1,3}\lambda$, and $e_{1,3}$ for $i = 1, \ldots, 6$ and $g_i(e_{2,3}, e_{3,1}, e_{3,2}, \lambda)$ are 2^{nd}order polynomials in 4 variables $e_{2,3}, e_{3,1}, e_{3,2}, \lambda$. So, the five variables $e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}$ can be expressed as functions of the other four variables $e_{2,3}, e_{3,1}, e_{3,2}, \lambda$ as

$$e_{1,1} = -g_1(e_{2,3}, e_{3,1}, e_{3,2}, \lambda), \tag{7.69}$$

$$e_{1,2} = -g_2(e_{2,3}, e_{3,1}, e_{3,2}, \lambda), \tag{7.70}$$

$$e_{1,3} = -g_6(e_{2,3}, e_{3,1}, e_{3,2}, \lambda), \tag{7.71}$$

$$e_{2,1} = -g_3(e_{2,3}, e_{3,1}, e_{3,2}, \lambda), \tag{7.72}$$

$$e_{2,2} = -g_4(e_{2,3}, e_{3,1}, e_{3,2}, \lambda). \tag{7.73}$$

We substitute these expressions into the remaining equation from the epipolar constraint and into the singularity and the trace constraint for $\mathtt{E}$ and obtain 11 polynomial equations in 4 unknowns (one of degree 3, four of degree 5 and six of degree 6):
One equation from the epipolar constraint

$$\lambda(-g_6(e_{2,3}, e_{3,1}, e_{3,2}, \lambda)) + g_5(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) = 0, \tag{7.74}$$

one equation from the singularity constraint

$$\det(\mathtt{E}) = 0, \tag{7.75}$$

and 9 equations from the trace constraint

$$2\left(\mathtt{E}\mathtt{E}^T\right)\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^T)\mathtt{E} = \mathtt{0} \tag{7.76}$$

with

$$\mathtt{E} = \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & e_{2,3} \\ e_{3,1} & e_{3,2} & 1 \end{pmatrix}. \tag{7.77}$$

Gröbner basis solution

To create an efficient Gröbner basis solver for solving this "6-point calibrated radial distortion" problem, we use the specialized Gröbner basis method presented in Section 4.4 and the single elimination strategy for generating polynomials from the ideal which is also implemented in the automatic generator of Gröbner basis solvers.

Identification of basic parameters

We first need to identify the basic parameters of the system of 11 equations (7.75)-(7.77) in four unknowns $e_{2,3}, e_{3,1}, e_{3,2}, \lambda$.

Using Macaulay 2 and random coefficients from some finite prime field the basis B (4.20) of the quotient ring $A = \mathbb{C}[e_{2,3}, e_{3,1}, e_{3,2}, \lambda]/I$ and the number of solutions of this system, which is in this case 52, was found. In general, more than one and less than 25 from these solutions are real.

To create the multiplication matrix and to compute B, we use the graded reverse lexicographic ordering with $e_{2,3} > e_{3,1} > e_{3,2} > \lambda$.

Construction of the multiplication matrix

Having the basis B, we know the form of the polynomials q_i (4.115) necessary for constructing the multiplication matrix. Therefore we can find an "elimination template" for its construction, i.e. an admissible Elimination trace, see Definition 21. To create the multiplication matrix we tested all four unknowns $e_{2,3}, e_{3,1}, e_{3,2}$ and λ and selected the multiplication matrix $\mathtt{M}_{e_{2,3}}$ which results in the simplest solution.

Using the method described in Section 4.4 and using the single elimination strategy for generating polynomials from the ideal we have found that to obtain all necessary polynomials q_i (4.115) for creating the multiplication matrix $\mathtt{M}_{e_{2,3}}$ we have to generate all monomial multiples of the initial 11 polynomial equations up to the total degree eight.

We thus obtain 356 polynomials in 495 monomials and after removing unnecessary polynomials using Algorithm 10 presented in Section 4.4.3 we get only 238 polynomials.

In the matrix representation of these 238 polynomials we can remove columns corresponding to "unnecessary monomials", i.e. monomials which do not have impact on the form of polynomials q_i. We hence finally end up with a 238×290 matrix. After performing the G-J elimination of this matrix we obtain all polynomials which are needed for the construction of the multiplication matrix $\mathtt{M}_{e_{2,3}}$.

Online Gröbner basis solver

The final online solver for solving all "non-degenerate" instances of 11 equations (7.75)-(7.77) in four unknowns, i.e. for solving the "6-point calibrated radial distortion problem", does only one G-J elimination of a 238×290 coefficient matrix from the found admissible Elimination trace. After the G-J elimination of this matrix, the multiplication matrix $\mathtt{M}_{e_{2,3}}$ is created from rows which correspond to the polynomials q_i (4.115). Then the eigenvectors of the multiplication matrix $\mathtt{M}_{e_{2,3}}$ are used to obtain solutions for $e_{2,3}, e_{3,1}, e_{3,2}$ and λ. Backsubstituting these solutions to equations (7.69)-(7.73) gives solutions also for the remaining elements of the essential matrix $\mathtt{E}$.

This solver, which is based on the presented specialized Gröbner basis method, is simpler than the previous solver for this problem proposed in [35], which consists of LU decomposition of much larger matrix of the size 320×363.

Next we describe the input to our automatic generator which delivers this efficient online solver for the "6-point calibrated distortion" problem.

Input to the automatic generator

The input to our automatic generator is described in Figure 7.23. This input is very similar to the

```
g1 = transpose(gbs_Vector('g1', 9));
g2 = transpose(gbs_Vector('g2', 9));
g3 = transpose(gbs_Vector('g3', 9));
g4 = transpose(gbs_Vector('g4', 9));
g5 = transpose(gbs_Vector('g5', 9));
g6 = transpose(gbs_Vector('g6', 9));

syms e23 e31 e32 k;

mon = [e23*k,e31*k,k*e32,k^2,e23,e31,e32,k,1]';

% parametrization of the essential matrix with four unknowns
e11 = -g1*mon;
e12 = -g2*mon;
e13 = -g6*mon;
e21 = -g3*mon;
e22 = -g4*mon;

% essential matrix
E = [e11 e12 e13; e21 e22 e23; e31 e32 1];

% eleven polynomial equations
eq(1) = -(k*g6)*mon + g5*mon;
eq(2) = det(E);

Et = transpose(E);
EEt = E*Et;
te = 2*(EEt)*E - trace(EEt)*E
eq(3:11) = te(:);

% four unknowns
unknown = {'e23' 'e31' 'e32' 'k'};

% known parameters
vars = transpose([g1(:); g2(:); g3(:); g4(:); g5(:); g6(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% call code generator
[res export] = gbs_CreateCode('ku6ptr', eq, known, unknown);
```

Figure 7.23: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the solver for the "6-point calibrated radial distortion problem".

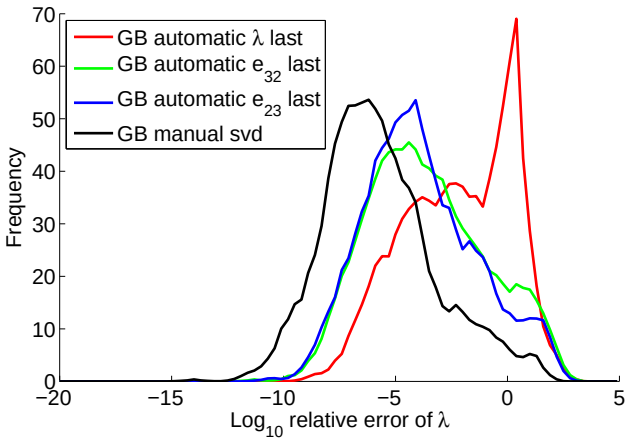

Figure 7.24: Log_{10} relative error of the radial distortion parameter λ obtained by selecting the real root closest to the ground truth value $\lambda_{th} = -0.3$ for noise free synthetic scene.

input for the remaining two radial distortion problems and results in an efficient online solver for solving 11 equations (7.75)-(7.77) in four unknowns. To obtain the final solver for the whole "6-point calibrated radial distortion problem", the backubstitution of the obtained solutions needs to be added to its output.

Experiments

We have evaluated the "6-point calibrated radial distortion solver" on synthetic noise free data.

In our experiment we have studied the numerical stability of the proposed Gröbner basis solver and compared it with the manually created Gröbner basis solver [35, 87].

In this synthetic experiment we also show how important may be the used ordering of variables for the stability of the final solver.

Figure 7.24 shows the log_{10} relative error of the radial distortion parameter λ obtained by selecting the real root closest to the ground truth value $\lambda_{th} = -0.3$ for noise free synthetic scene.

We have compared four solvers for the "6-point calibrated radial distortion problem". Three from these solvers were created using the proposed specialized Gröbner basis method and our automatic generator presented in Chapter 6. They differ only in the used monomial ordering. The first Gröbner basis solver was created for the ordering of variables $e_{23} > e_{31} > e_{32} > \lambda$ (red), the second Gröbner basis solver for the ordering $\lambda > e_{23} > e_{31} > e_{32}$ (green) and the third solver for the ordering $\lambda > e_{31} > e_{32} > e_{23}$ (blue). The final solver is the manually created and numerically optimized solver presented in [35, 87] (black).

It can be seen that the manually created solver (black) is numerically more stable. It is because this solver uses special methods for improving the numerical stability of Gröbner basis solvers presented in paper [34]. However this solver is also the largest and slowest from all tested solvers. In the case of our Gröbner basis solvers created using the automatic generator, the

solvers for the ordering of variables $\lambda > e_{23} > e_{31} > e_{32}$ (blue) and $\lambda > e_{31} > e_{32} > e_{23}$ (green) behave quite similar and stable, however the solver for the ordering $e_{23} > e_{31} > e_{32} > \lambda$ (red) suffers from a quite large instability.

This shows that the ordering of variables may be quite important for the stability of the final solver. In this case the solvers for all orderings of variables have the same size, however the size of the final solvers may for different ordering of variables differ. Therefore, it is important to create solvers for different orderings and select the "best" one w.r.t. the stability or the size.

Computational complexity

The MATLAB implementation of the Gröbner basis solver, which was generated using our automatic generator, needs to perform G-J elimination of a 238×290 matrix and to compute eigenvalues of a 52×52 matrix. This implementation runs about $25.2ms$ on a Intel i7 Q720 notebook. The running time of the manually created solver [35, 87] is about $42.9ms$. This solver needs to perform LU decomposition of a 320×363 matrix and to compute eigenvalues of a 52×52 matrix. The C++ version of our Gröbner basis solver runs on the same hardware about $7.7ms$.

7.1.6 5-point relative pose problem

We will next describe solutions to four relative pose problems for standard perspective cameras without radial distortion. We start with the well known, very practical and popular problem of estimating relative pose of two calibrated cameras from five image point correspondences, also called the "5-point relative pose problem".

Previous solutions

The 5-pt relative pose problem was studied already by Kruppa [80] who has shown that it has at most eleven solutions. Maybank and Faugeras [47] then sharpened Kruppa's result by showing that there are at most ten solutions. Recently, Nister et al. [113] have shown that the problem really requires solving a ten degree polynomial.

Kruppa in [80] also gave an algebraic algorithm for solving the 5-pt problem. It was implemented by Maybank and Faugearas [47] but turned out not to be particularly efficient and practical. More efficient and practical solution has been invented by Philip [117]. He designed a method for finding the solutions by extracting the roots of a thirteen degree polynomial. In [45] was proposed a solution based on sparse resultants.

Nister [112] and Stewénius et al. [132] proposed first efficient solvers to the 5-point relative pose problem that obtain the solutions as the roots of a tenth-degree polynomial.

In both methods [112, 132], the five linear epipolar constraints were used to parametrize the essential matrix as a linear combination of a basis $\mathtt{E}_1, \mathtt{E}_2, \mathtt{E}_3, \mathtt{E}_4$ of the space of all compatible essential matrices

$$\mathtt{E} = x\,\mathtt{E}_1 + y\,\mathtt{E}_2 + z\,\mathtt{E}_3 + \mathtt{E}_4. \tag{7.78}$$

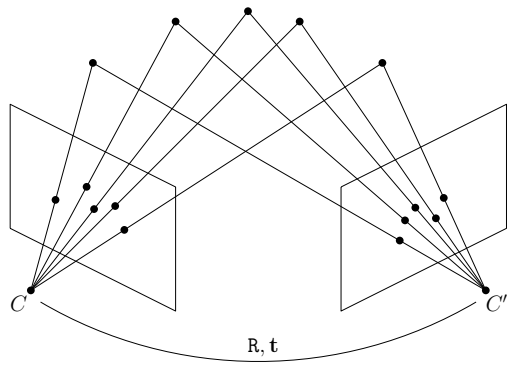

Figure 7.25: Illustration of the 5-point relative pose problem.

Then, the rank constraint (7.6) and the trace constraint (7.7) were used to build ten third-order polynomial equations in three unknowns and 20 monomials. These equations can be written in a matrix form

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.79}$$

with coefficient matrix $\mathtt{M}$, which is in fact the matrix representation (4.24) of these ten polynomials, and the vector of all monomials $\mathbf{X}$. The matrix $\mathtt{M}$ in (7.79) was in both these solutions reduced by G-J elimination.

The method [112] then used relations between polynomials (7.79) to derive three new equations. The new equations were arranged into a 3×3 matrix equation $\mathtt{A}(z)\,\mathbf{Z} = 0$ with matrix $\mathtt{A}(z)$ containing polynomial coefficients in z and $\mathbf{Z}$ containing the monomials in x and y. The solutions were obtained using the hidden variable method [39] by solving the tenth degree polynomial $\det(\mathtt{A}(z))$, finding $\mathbf{Z}$ as a solution of a homogeneous linear system, and constructing $\mathtt{E}$ from (7.78).

The method [132] follows the Gröbner basis approach for solving systems of polynomial equations [39] described in Section 4.3.3 First, a Gröbner basis of the ideal generated by equations (7.79) is found. Then, a multiplication matrix (4.44) is constructed. Finally, the solutions are obtained by computing the eigenvectors of the multiplication matrix. This approach turned out to lead to a particularly simple procedure for the 5-pt problem since the particular Gröbner basis used and the 10×10 multiplication matrix can be constructed directly from the reduced coefficient matrix $\mathtt{M}$ in (7.79).

Another technique, based on the hidden variable resultant for solving the 5-pt relative pose problem, was proposed in [98]. This technique is somewhat easier to understand than [132] but is far less efficient. Maple was used in [98] to evaluate large polynomial determinants.

Next we summarize the formulation of the 5-point relative pose problem.

Problem formulation

In the case of the 5-point relative pose problem, the epipolar constraint

$$\mathbf{x}_i'^{\top}\,\mathtt{E}\,\mathbf{x}_i = 0, \tag{7.80}$$

for $i = 1, \ldots, 5$ results in five equations in nine unknowns of the essential matrix $\mathtt{E}$. Therefore, the essential matrix can be parametrized as a linear combination of a basis $\mathtt{E}_1, \mathtt{E}_2, \mathtt{E}_3, \mathtt{E}_4$ of the 4-dimensional space of all compatible essential matrices. This means that we can write

$$\mathtt{E} = x\,\mathtt{E}_1 + y\,\mathtt{E}_2 + z\,\mathtt{E}_3 + w\,\mathtt{E}_4, \tag{7.81}$$

where x, y, z and w are four unknowns. Since the essential matrix $\mathtt{E}$ is given only up to scale we can set one of these unknowns to one, e.g. $w = 1$, and write

$$\mathtt{E} = x\,\mathtt{E}_1 + y\,\mathtt{E}_2 + z\,\mathtt{E}_3 + \mathtt{E}_4. \tag{7.82}$$

With this parametrization the rank constraint

$$\det(\mathtt{E}) \quad = \quad 0 \tag{7.83}$$

and the trace constraint

$$2\,\mathtt{E}\mathtt{E}^\top\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^\top)\mathtt{E} \quad = \quad \mathtt{0} \tag{7.84}$$

result in ten third-order polynomial equations in three unknowns x, y and z and 20 monomials.

Next we briefly describe the Gröbner basis solution of this system of polynomial equations and then our new polynomial eigenvalue solution based on the specialized resultant based method described in Section 5.5.

Gröbner basis solutions

The Gröbner basis solver for the five point relative pose problem proposed by Stewénius in [132] seems to be the smallest possible solver for this problem based on the Gröbner basis method. This solver leads to one G-J elimination of a 10×20 coefficient matrix $\mathtt{M}$ and to the computation of eigenvalues of a 10×10 multiplication matrix which is constructed directly from the reduced matrix $\mathtt{M}$.

Using the automatic generator presented in Chapter 6 and the specialized Gröbner basis method from Section 4.4, a Gröbner basis solver of the same size as in [132] can be obtained. It is because we obtain all polynomials q_i (4.115) necessary for constructing a multiplication matrix already after the G-J elimination of the coefficient matrix $\mathtt{M}$ in the matrix representation (4.24) of the initial ten polynomials.

The input to the automatic generator for this 5-point relative pose problem is presented in Figure 7.26.

Polynomial eigenvalue solution

To obtain a polynomial eigenvalue solution to the 5-point relative pose problem we use the same formulation as it was used for the Gröbner basis solution.

In this formulation the rank (7.83) and the trace constraints (7.84) lead to ten third-order polynomial equations in three unknowns x, y and z and in 20 monomials and can be written in the matrix form

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.85}$$

```
% symbolic matrices
% these matrices represent basis of the null space
% of linearized epipolar equation y'*E*x = 0
% matrices E1, E2, E3, E4 will be treated as known
E1 = gbs_Matrix('E1%d%d', 3 ,3);
E2 = gbs_Matrix('E2%d%d', 3 ,3);
E3 = gbs_Matrix('E3%d%d', 3 ,3);
E4 = gbs_Matrix('E4%d%d', 3 ,3);

% create essential matrix as a linear combination of the null space
syms x y z;
E = x*E1 + y*E2 + z*E3 +  E4;

% build polynomial equations
%  rank constraint
eq(1) = det(E);

% trace constraint
Et = transpose(E);
EEt = E*Et;
te = 2*(EEt)*E - trace(EEt)*E;
eq(2:10) = te(:);

% three unknowns
unknown = {'x' 'y' 'z'};

% known parameters
vars = transpose([E1(:); E2(:); E3(:); E4(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% specify which "known" variables should be grouped
% into a single input argument (as a vector).
% "kngroups" is a vector where kngroups(k) = l says that k-th
% known variable should be grouped into l-th vector.
kngroups = ones(9,1)*[1 2 3 4];

% call code generator
[res export] = gbs_CreateCode('ku5pt', eq, known, unknown, kngroups);
```

Figure 7.26: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the online solver for the "5-point calibrated relative pose problem".

where $\mathtt{M}$ is a 10×20 coefficient matrix from the matrix representation (4.24) of these ten polynomials and $\mathbf{X} = (x^3, yx^2, y^2x, y^3, zx^2, zyx, zy^2, z^2x, z^2y, z^3, x^2, yx, y^2, zx, zy, z^2, x, y, z, 1)^\top$ is the vector of all 20 monomials. There are all monomials in all three unknowns up to degree three. So we can use any unknown to play the role of α in (5.44). For example, taking $\alpha = z$, these ten equations (7.85) can be rewritten as

$$\left(z^3\mathtt{C}_3 + z^2\mathtt{C}_2 + z\,\mathtt{C}_1 + \mathtt{C}_0\right)\mathbf{v} = \mathbf{0}, \tag{7.86}$$

where $\mathbf{v}$ is a 10×1 vector of monomials, $\mathbf{v} = (x^3, x^2y,\ xy^2, y^3, x^2, xy, y^2, x, y, 1)^\top$ and $\mathtt{C}_3$, $\mathtt{C}_2$, $\mathtt{C}_1$ and $\mathtt{C}_0$ are 10×10 coefficient matrices. $\mathtt{C}_3 \equiv (\mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ m_{10}})$, $\mathtt{C}_2 \equiv (\mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ m_8\ m_9\ m_{16}})$, $\mathtt{C}_1 \equiv (\mathbf{0\ 0\ 0\ 0\ m_5\ m_6\ m_7\ m_{14}\ m_{15}\ m_{19}})$, and $\mathtt{C}_0 \equiv (\mathbf{m_1\ m_2\ m_3\ m_4\ m_{11}\ m_{12}\ m_{13}\ m_{17}\ m_{18}\ m_{20}})$, where $\mathbf{m_j}$ is the j^{th} column from the coefficient matrix $\mathtt{M}$.

Since $\mathtt{C}_3$, $\mathtt{C}_2$, $\mathtt{C}_1$ and $\mathtt{C}_0$ are known square matrices, the formulation (7.86) is a cubic PEP and can be solved using standard efficient algorithms presented in Section 5.4.

In this case the rank of the matrix $\mathtt{C}_3$ is only one while the matrix $\mathtt{C}_0$ is regular. Therefore, we can use the transformation $\beta = 1/z$ and reduce the cubic PEP (7.86) to the problem of finding the eigenvalues of the following 30×30 matrix

$$\mathtt{A} = \begin{pmatrix} 0 & \mathtt{I} & 0 \\ 0 & 0 & \mathtt{I} \\ -\mathtt{C}_0^{-1}\mathtt{C}_3 & -\mathtt{C}_0^{-1}\mathtt{C}_2 & -\mathtt{C}_0^{-1}\mathtt{C}_1 \end{pmatrix}. \tag{7.87}$$

After solving this eigenvalue problem we obtain 30 eigenvalues, solutions for $\beta = 1/z$, and 30 corresponding eigenvectors $\mathbf{v}$ from which we extract solutions for x and y.

However, there are 20 zero eigenvalues among these 30 eigenvalues. In this case these zero eigenvalues can be easily eliminated, since they correspond to the "zero columns" of the matrices $-\mathtt{C}_0^{-1}\mathtt{C}_3$, $-\mathtt{C}_0^{-1}\mathtt{C}_2$ and $-\mathtt{C}_0^{-1}\mathtt{C}_1$ as described in Section 5.5. The elimination of these zero eigenvalues is done in the offline phase of creating the final online solver.

Therefore, to solve the five point relative pose problem it is sufficient to find eigenvalues and eigenvectors of a 10×10 matrix which we obtain from the matrix (7.87) by removing the "zero columns" and the corresponding rows of the matrix (7.87). By the row corresponding to some column we mean the row with the same index as the column.

Resulting 10×10 matrix is in fact the multiplication matrix used in the Gröbner basis solver [132] which is in this case obtained directly from the polynomial eigenvalue formulation without any knowledge about the Gröbner bases or the properties of these multiplication matrices. Moreover, the size of this matrix corresponds to the dimension of the problem which was proved to be 10 [113].

Experiments

In this subsection we evaluate our polynomial eigenvalue solution to the 5-point relative pose problem and compare it with the existing state of the art methods [112, 132].

Since all these three solvers, the Gröbner basis solver [132], the Nister's solver [112] and the proposed polynomial eigenvalue solver, are algebraically equivalent and differ only in the

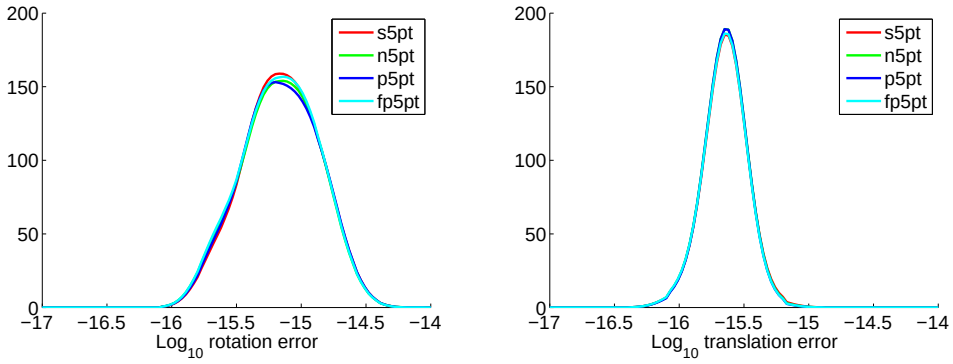

Figure 7.27: Numerical stability of the 5-point relative pose problem considered algorithms. The log_{10} relative rotation error (left) and the log_{10} translation error (right).

way of constructing the final 10×10 matrix which eigenvalues give solutions, respectively in constructing the final tenth degree polynomial, we have evaluated all solver on synthetic noise free data only. We aimed at studying the numerical stability and the speed of the algorithms. The properties of this 5-point relative pose problem in different configurations and under different noise contaminations are detailed studied in [112, 132].

Numerical stability

In our experiments, all scenes were generated using 3D points randomly distributed in a 3D cube. Each 3D point was projected by a camera with a random feasible orientation and position.

To study the numerical stability of the calibrated 5-point relative pose problem we extracted camera relative rotations and translations from estimated essential matrices. From the four possible choices of rotations and translations we selected the one where all 3D points were in front of the canonical camera pair [61].

Let $\mathtt{R}$ be an estimated camera relative rotation and $\mathtt{R}_{gt}$ be the corresponding ground-truth rotation. The rotation error is measured as an angle in the angle axis representation of the relative rotation $\mathtt{R}\,\mathtt{R}_{gt}^{-1}$ and the translation error as an angle between the ground-truth and the estimated translation vectors. Figure 7.27 compares results of different five point solvers: gb5pt denotes the Stewenius' Gröbner basis solver [132], nongb5pt - the Nister's [112], peig5pt - the polynomial eigenvalue solver presented in [84], i.e. the solver without removed zero eigenvalues, and fpeig5pt denotes the fast polynomial eigenvalue solver with removed zero eigenvalues proposed in this section. The log_{10} rotation error is displayed in Figure 7.27 (left) and the log_{10} translation error in Figure 7.27 (right).

The precision and the numerical stability of all solvers is very similar and good. Although there are small differences in the precision of the essential matrix estimation, these differences disappear after using the SVD algorithm for extracting the rotation and the translation from the essential matrix [61]. This is because all 5-point relative pose algorithms balance on the edge of the machine precision and therefore there is negligible difference between them in Figure 7.27.

Computational complexity

We have implemented both our solvers in C++. The Gröbner basis solver which comes directly from the automatic generator needs to compute the inverse of one 10×10 matrix and then to compute eigenvalues of a matrix of the same size. This solver runs $61.2\mu s$ on a Intel i7 Q720 notebook.

For this 5-point relative pose problem we have found that it is more efficient to compute directly the characteristic polynomial of the 10×10 matrix and to find solutions by computing the roots of this 10^{th} degree characteristic polynomial using the Sturm sequences [66], than to use standard eigenvalue algorithms. We have tested several methods for computing characteristic polynomials, i.e. the Faddeev-Leverrier [46], the Krylov's [67] and the Danilevsky method [41]. The running times of the Gröbner basis solver using these methods are $17.2\mu s$ for the Faddeev-Leverrier method, $13.7\mu s$ for the Krylov's and $14.2\mu s$ for the Danilevsky method. All these times were again measured on a Intel i7 Q720 notebook and are about $4\times$ faster than the time of the standard Gröbner basis solver with the eigenvalue computation. More about all tested methods for computing characteristic polynomials and about the stability of final 5-point relative pose solvers can be found in our paper [30].

Our polynomial eigenvalue solver needs to perform the same operations as the Gröbner basis solver, i.e. to compute the inverse of one 10×10 matrix and then to compute eigenvalues of a matrix of the same size e.g. using some the characteristic polynomial method. Therefore its running time is the same as in the case of our Gröbner basis solver for this problem.

Our reimplementation of the state-of-the-art Nisters algorithm [112] runs about $10.6\mu s$. This is the fastest solver for the 5-point relative pose problem. It is because in this solution almost all computations can be done in a closed form.

7.1.7 6-point equal focal length problem

It quite often happens that the only unknown calibration parameter of a camera is the focal length. It is because for today's cameras we can assume that the skew s in the calibration matrix $\mathtt{K}$ (7.3) is 0, the pixel aspect ratio $\alpha = 1$, and the principal point is in the center of the image. In such a cases the following minimal problem of estimating relative camera position for two cameras with unknown, but equal focal length can be used.

Previous solutions

The problem of estimating relative camera pose for two cameras with unknown, but equal, focal length from the minimal number of six point correspondences has 15 solutions [134].

The first minimal solution to this problem proposed by Stewénius et. al. [134] is based on the Gröbner basis techniques and is similar to the Stewénius' solution to the 5-pt problem [132]. Using the linear epipolar constraint, the fundamental matrix is parameterized by two unknowns as

$$\mathtt{F} = x\,\mathtt{F}_1 + y\,\mathtt{F}_2 + \mathtt{F}_3. \tag{7.88}$$

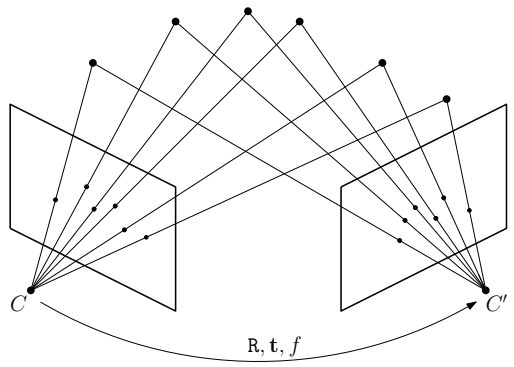

Figure 7.28: Illustration of the 6-point equal focal length problem.

Using the rank constraint for the fundamental matrix (7.14) and the trace constraint for the essential matrix (7.15) then brings ten third and fifth order polynomial equations in three unknowns x, y, and $w = f^{-2}$, where f is the unknown focal length.

The Gröbner basis solver [134] starts with these ten polynomial equations which can be represented by a 10×33 matrix $\mathtt{M}$. Since this matrix doesn't contain all necessary polynomials for creating a multiplication matrix, two new polynomials ($w \det(\mathtt{F})$ and $w^2 \det(\mathtt{F})$) are added and the matrix $\mathtt{M}$ is reduced by the G-J elimination. Further, four new polynomials are added and eliminated then. Finally, two more polynomials are added and eliminated. The resulting system then constitutes a Gröbner basis and can be used to construct the multiplication matrix. The resulting solver therefore consists of three G-J eliminations of three matrices of size 12×33, 16×33, and 18×33. The eigenvectors of the multiplication matrix provide the solutions for the three unknowns x, y, and $w = f^{-2}$.

Another Gröbner basis solver for this problem was proposed in [33]. This solver uses only one G-J elimination of a 34×50 matrix and uses a special technique for improving the numerical stability of Gröbner basis solvers by selecting a suitable basis of the quotient ring modulo the ideal generated by the Gröbner basis. In this paper it was shown that this solver gives more accurate results than the original solver [134].

Yet another solution based on the hidden variable resultant method was proposed in [96] but it has similar problems as the hidden variable solution to the 5-pt problem [98].

Next we provide the formulation of this 6-point equal focal length problem.

Problem formulation

To solve the 6-point equal focal length problem we use a similar formulation as in the case of the the 5-point relative pose problem. We first use the epipolar constraint (7.10) for six point correspondences to parametrize the fundamental matrix $\mathtt{F}$ as a linear combination of a 3-dimensional basis $\mathtt{F}_1, \mathtt{F}_2, \mathtt{F}_3$ of the space of all compatible fundamental matrices

$$\mathtt{F} = x\,\mathtt{F}_1 + y\,\mathtt{F}_2 + z\,\mathtt{F}_3, \tag{7.89}$$

where x, y and z are three unknowns. We can again set one of these unknowns to one, e.g. $z = 1$, and write

$$\mathtt{F} = x\,\mathtt{F}_1 + y\,\mathtt{F}_2 + \mathtt{F}_3. \tag{7.90}$$

In this case we have a camera pair with unknown but equal focal length f and therefore calibration matrices of both cameras are equal, i.e. $\mathtt{K}' = \mathtt{K}$, and diagonal of the form (7.4).

This means that for the essential matrix there holds

$$\mathtt{E} = \mathtt{K}^\top \mathtt{F}\,\mathtt{K} = \mathtt{K}\,\mathtt{F}\,\mathtt{K}. \tag{7.91}$$

Since the essential matrix $\mathtt{E}$ is given only up to scale we can multiply equation (7.91) by $\frac{1}{f}$ from both sides and therefore write the calibration matrix as

$$\mathtt{K} \simeq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix}. \tag{7.92}$$

After substituting the expression for the essential matrix (7.91) into the trace constraint (7.7), we obtain

$$2\,\mathtt{K}\,\mathtt{F}\,\mathtt{K}\,\mathtt{K}\,\mathtt{F}^\top\mathtt{K}\,\mathtt{K}\,\mathtt{F}\,\mathtt{K} - \operatorname{trace}(\mathtt{K}\,\mathtt{F}\,\mathtt{K}\,\mathtt{K}\,\mathtt{F}^\top\mathtt{K})\,\mathtt{K}\,\mathtt{F}\,\mathtt{K} \;=\; \mathtt{0}. \tag{7.93}$$

Since $\mathtt{K}$ is regular we can multiply equation (7.93) by $\mathtt{K}^{-1}$ from left and right. Then, after applying the substitution $\mathtt{Q} = \mathtt{K}\,\mathtt{K}$, we obtain

$$2\,\mathtt{F}\,\mathtt{Q}\,\mathtt{F}^\top\mathtt{Q}\,\mathtt{F} - \operatorname{trace}(\mathtt{F}\,\mathtt{Q}\,\mathtt{F}^\top\mathtt{Q})\,\mathtt{F} \;=\; \mathtt{0}. \tag{7.94}$$

This matrix equation brings nine fifth order polynomial equations in three unknowns x, y, and $w = f^{-2}$, where f is the unknown focal length. Together with the third degree equation from the singularity constraint

$$\det(\mathtt{F}) = 0 \tag{7.95}$$

we obtain a system of ten polynomial equations in three unknowns x, y and $w = f^{-2}$.

Gröbner basis solution

In this subsection we describe all basic steps of the process of creating an efficient Gröbner basis solver for solving this 6-point equal focal length problem using our specialized Gröbner basis method presented in Section 4.4. This solver is similar to the previous Gröbner basis solutions [134, 33] and is based on the single elimination strategy for generating polynomials from the ideal and therefore can be easily obtained using our automatic generator.

Identification of basic parameters

In the first step of creating an efficient solver for this problem we identify the basic parameters of the system of 10 equations (7.94) and (7.95) in three unknowns x, y and $w = f^{-2}$.

In this case we use the graded reverse lexicographic ordering with $x > y > w$. With this ordering and using Macaulay 2 we can find that the basis B (4.20) of the quotient ring $A = \mathbb{C}[x, y, w]/I$ has the form $B = \{[1], [x], [x^2], [xy], [xyw], [xw], [xw^2], [y], [y^2], [y^2w], [yw], [yw^2], [w], [w^2], [w^3]\}$ and that the number of solutions of this system is 15.

Construction of the multiplication matrix

In the next step we find an admissible Elimination trace for constructing the multiplication matrix. For this problem we have generated three different Elimination traces for all three variables, i.e. multiplication matrices for three different "action variables" x, y and w.

Using the single elimination strategy for generating polynomials from the ideal, i.e. Algorithm 9, we have found that to obtain all necessary polynomials q_i (4.115) for creating the multiplication matrix $\mathtt{M}_w$, we have to generate all monomial multiples of the initial ten polynomial equations up to total degree eight. This results in a 236×165 matrix. After removing unnecessary polynomials using Algorithm 10, only 41 polynomials in 60 monomials remain.

For the "action variables" x and y the method described in Section 4.4.3 requites to generate all monomial multiples of initial ten polynomial equations up to total degree seven. This results in a 125×120 matrix. In the reduction step, 94 polynomials out of these 125 are removed, resulting in 31 polynomials in 50 monomials. Since in the matrix representation of these 31 polynomials we can remove columns corresponding to "unnecessary" monomials, i.e. monomials which do not have impact on the form of polynomials q_i we finally end up with a 31×46 matrix. After the G-J elimination of this matrix, all necessary polynomial are obtained and the multiplication matrix $\mathtt{M}_x$ ($\mathtt{M}_y$) can be created.

Online Gröbner basis solver

The online solver for solving all "non-degenerate" instances of ten equations (7.94) and (7.95) in three unknowns x, y and $w = f^{-2}$, i.e. for solving the "6-point equal focal length problem", then does one G-J elimination of the 31×46 coefficient matrix. After the G-J elimination of this matrix, the multiplication matrix $\mathtt{M}_x$ is created from rows which correspond to the polynomials q_i (4.115). Solutions for our three variables x, y and $w = f^{-2}$ are obtained from the eigenvectors of this multiplication matrix $\mathtt{M}_x$. Finally, the fundamental matrix $\mathtt{F}$ is obtained by backsubstituting solutions for x and y into equation (7.90).

Our solver results in a smaller matrix than the solver proposed in [33] which consists of one G-J elimination of a 34×50 matrix.

The input to our automatic generator which produces this efficient online solver is shown in Figure (7.29).

Polynomial eigenvalue solution

The polynomial eigenvalue solution to the 6-pt equal focal length problem starts with the same ten third and fifth order polynomial equations (7.94) and (7.95) in three unknowns x, y and $w = f^{-2}$ as the Gröbner basis solution. These equations can be again written in a matrix form

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.96}$$

```
% symbolic matrices
% these matrices represent basis of the null space
% of linearized epipolar equation y'*F*x = 0
% matrices F1, F2, F3 will be treated as known
F1 = gbs_Matrix('F1%d%d', 3 ,3);
F2 = gbs_Matrix('F2%d%d', 3 ,3);
F3 = gbs_Matrix('F3%d%d', 3 ,3);

% create fundamental matrix as a linear combination of the null space
% w = 1/focal^2
% variables x, y, w are unknown
syms x y w real;
F = x*F1 + y*F2 + F3;

% build polynomial equations
%  rank constraint
eq(1) = det(F);

Q = diag([1 1 w]);

% calibrate fundamental matrix using unknown focal length and use
% the trace constrain for obtained essential matrix
Ft = transpose(F);
FQFtQ = F*Q*Ft*Q;

te = 2*(FQFtQ)*F - trace(FQFtQ)*F;
eq(2:10) = te(:);

% three unknowns
unknown = {'x' 'y' 'w'};

% known parameters
vars = transpose([F1(:); F2(:); F3(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% "kngroups" is a vector where kngroups(k) = l says that k-th
% known variable should be grouped into l-th vector.
kngroups = ones(9,1)*[1 2 3];

% call code generator
[res export] = gbs_CreateCode('ku6pt', eq, known, unknown, kngroups);
```

Figure 7.29: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the solver for the "6-point equal focal length problem"

where $\mathtt{M}$ is a 10×30 coefficient matrix from the matrix representation of these polynomials and $\mathbf{X} = (w^2x^3, w^2yx^2, w^2y^2x, w^2y^3, wx^3, wyx^2, wy^2x, wy^3, w^2x^2, w^2yx, w^2y^2, x^3, yx^2, y^2x, y^3, wx^2, wyx, wy^2, w^2x, w^2y, x^2, yx, y^2, wx, wy, w^2, x, y, w, 1)^\top$ is a vector of 30 monomials. Variables x and y appear in degree three and w only in degree two. Therefore, we have selected $\alpha = w$ in the PEP formulation (5.44). Then, these ten equations can be rewritten as

$$\left(w^2\mathtt{C}_2 + w\,\mathtt{C}_1 + \mathtt{C}_0\right)\mathbf{v} = \mathbf{0}, \tag{7.97}$$

where $\mathbf{v}$ is a 10×1 vector of monomials, $\mathbf{v} = (x^3, x^2y, xy^2, y^3, x^2, xy, y^2, x, y, 1)^\top$ and $\mathtt{C}_2, \mathtt{C}_1$ and $\mathtt{C}_0$ are 10×10 coefficient matrices. $\mathtt{C}_2 \equiv (\mathbf{m_1}\ \mathbf{m_2}\ \mathbf{m_3}\ \mathbf{m_4}\ \mathbf{m_9}\ \mathbf{m_{10}}\ \mathbf{m_{11}}\ \mathbf{m_{19}}\ \mathbf{m_{20}}\ \mathbf{m_{26}})$, $\mathtt{C}_1 \equiv (\mathbf{m_5}\ \mathbf{m_6}\ \mathbf{m_7}\ \mathbf{m_8}\ \mathbf{m_{16}}\ \mathbf{m_{17}}\ \mathbf{m_{18}}\ \mathbf{m_{24}}\ \mathbf{m_{25}}\ \mathbf{m_{29}})$ and $\mathtt{C}_0 \equiv (\mathbf{m_{12}}\ \mathbf{m_{13}}\ \mathbf{m_{14}}\ \mathbf{m_{15}}\ \mathbf{m_{21}}\ \mathbf{m_{22}}\ \mathbf{m_{23}}\ \mathbf{m_{27}}\ \mathbf{m_{28}}\ \mathbf{m_{30}})$, where $\mathbf{m_j}$ is the j^{th} column from the coefficient matrix $\mathtt{M}$ (7.96).

The formulation (7.97) is directly the QEP which can be solved using the methods presented in Section 5.4. After solving QEP (7.97), by transforming it to a GEP, we obtain 20 eigenvalues, solutions for $w = f^{-2}$, and 20 corresponding eigenvectors $\mathbf{v}$ from which we extract solutions for x and y. To do this, we normalize solutions for $\mathbf{v}$ to have the last coordinate equal to 1 and extract values from $\mathbf{v}$ that correspond to x and y, in this case $\mathbf{v}(8)$ and $\mathbf{v}(9)$. Then we use (7.90) to compute $\mathtt{F}$.

Between computed eigenvalues there are again zero eigenvalues, solutions for $1/w$, like in the five point case. However, in this case these eigenvalues can not be so easily eliminated, since here we do not have zero columns corresponding to these eigenvalues. Therefore, this polynomial eigenvalue solver delivers 20 solutions which is more than the number of solutions to the original problem. This is caused by the fact that we are solving a relaxed version of the original problem as described in Section 5.5.1. Therefore, the solution contains not only all $\mathbf{v}'$s that automatically (within limits of numerical accuracy) satisfy the constraints induced by the problem, i.e. $\mathbf{v}(1) = \mathbf{v}(8)^3$, but also additional $\mathbf{v}'$s that do not satisfy them. Such $\mathbf{v}'$s need to be eliminated, e.g., by verifying the monomial dependences as described in Section 5.5.1.

Experiments

We evaluate both our solutions to the 6-point equal focal length problem and compare them with the existing state of the art method [134].

Since all three solvers, the Gröbner basis solver from [134], our Gröbner basis solver presented in this section and the polynomial eigenvalue solver use the same formulation of the problem and differ only in the way of solving the system of equations arising from this formulation, we have evaluated them on synthetic noise free data only.

The properties of this 6-point equal focal length problem in different configurations and under different noise contaminations are studied in [134].

Numerical stability

In all our experiments, the scenes were generated using 3D points randomly distributed in a 3D cube. Each 3D point was projected by a camera with random feasible orientation and position and random or fixed focal length.

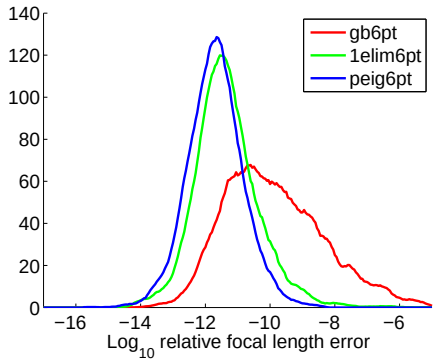

Figure 7.30: The log_{10} relative focal length error for the 6-point equal focal length problem. Here gb6pt denotes Stewenius Gröbner basis solver [134], 1elim6pt - the Gröbner basis solution with single elimination presented in this section and peig6pt our polynomial eigenvalue solver.

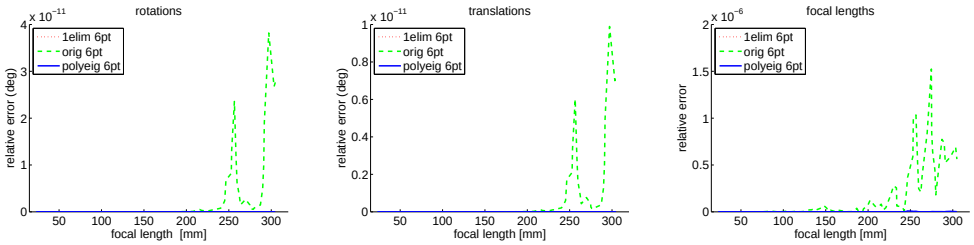

Figure 7.31: Backward moving and zooming camera test.

In evaluation of both our solvers for the 6-point equal focal length problem, we focused on the value of estimated focal length. We measured relative focal length error $(f - f_{gt})/f_{gt}$, where f is an estimated focal length and f_{gt} denotes the ground truth.

Figure 7.30 compares the numerical stability of the three solvers for the 6-point equal focal length problem for noise free data. In this figure, gb6pt denotes the state-of-the-art Gröbner basis solver [134], 1elim6pt - the Gröbner basis solution with single elimination presented in this section and created using our automatic generator from Chapter 6 and peig6pt denotes our polynomial eigenvalue solver. As it can be seen both our solvers presented in this section outperform the state-of-the-art Gröbner basis solver from [134]. The best results are obtained for the polynomial eigenvalue solver which is even slightly more stable than our Gröbner basis solver. It is probably caused by the fact that the Gröbner basis solver performs the G-J elimination on the matrix representing the input and additional polynomials from the ideal and the eigenvalues are computed from the matrix created from these eliminated coefficients. On the other hand the polynomial eigenvalue solver computes eigenvalues, which give solutions to the problem, from the matrix created directly from the coefficients of the input polynomials.

In the next experiment we tested the behavior of the algorithms for the camera pair with a

constant baseline length (0.5m) moving backwards and zooming to keep image filled by the projection of the 3D scene. For each position of the cameras pair, we executed all algorithms several times and selected median values from measured errors. Results are displayed in Figure 7.31. We see that the state-of-the-art 6-point algorithm [134] returns less accurate results with increasing distance and focal length. Both, the single elimination Gröbner basis solver presented in this section and our polynomial eigenvalue solver, return comparable results here.

Computation complexity

We have implemented the Gröbner basis solver and the polynomial eigenvalue solver, in C++.

The presented Gröbner basis solver generated using our automatic generator needs to performs G-J elimination of a 31×46 matrix and computes eigenvalues of the 15×15 matrix. The MATLAB+MEX version obtained directly from the automatic generator runs on Intel i7 Q720 notebook about $650\mu s$ and a little bit optimized C++ version runs about $176.3\mu s$

As in the case of the 5-point relative pose problem we have found that also for this problem it is more efficient to compute directly the characteristic polynomial of the 15×15 matrix and to find solutions by computing the roots of this 15^{th} degree characteristic polynomial using the Sturm sequences [66], than to use standard eigenvalue algorithms. We have tested three methods for computing characteristic polynomials, i.e. the Faddeev-Leverrier [46], the Krylov's [67] and the Danilevsky method [41]. The running times of the Gröbner basis solver using these methods are $21.3\mu s$ for the Krylov's and $22.6\mu s$ for the Danilevsky method. All these times were again measured on a Intel i7 Q720 notebook and are about $8\times$ faster than the time of the standard Gröbner basis solver with the eigenvalue computation. In this case we do not provide results for the Faddeev-Leverrier algorithm [46] because its numerical stability is poor and it failed to deliver any result most of the time. More about the stability of all tested 6-point equal focal length solvers can be found in our paper [30].

In the case of the polynomial eigenvalue solver we have decided to transform the resulting polynomial eigenvalue problem (7.97) to the standard eigenvalue problem (5.54) as described in Section 5.4 and used a standard numerical eigenvalue method to solve it. This solver is easy to implement since it requires only to fill three matrices with appropriate coefficients extracted from the input equations. Then matrix (5.54), which is the input to some standard numerical eigenvalue algorithm, is constructed using these matrices and inverse of one from them.

The polynomial eigenvalue solver for the 6-point equal focal length problem needs to compute the inverse of one 10×10 matrix and then to compute eigenvalues of a 20×20 matrix. This solver runs on the same hardware as the Gröbner basis solvers about $182\mu s$.

7.1.8 6-point one calibrated camera problem

In this subsection we provide efficient and robust minimal solutions to the configuration with one completely calibrated camera and one camera with an unknown focal length. We show that this solution can cope with most unpleasant degeneracies and can be effectively used to reconstruct 3D scenes from collections of images with very few (in principle single) images with known focal length(s).

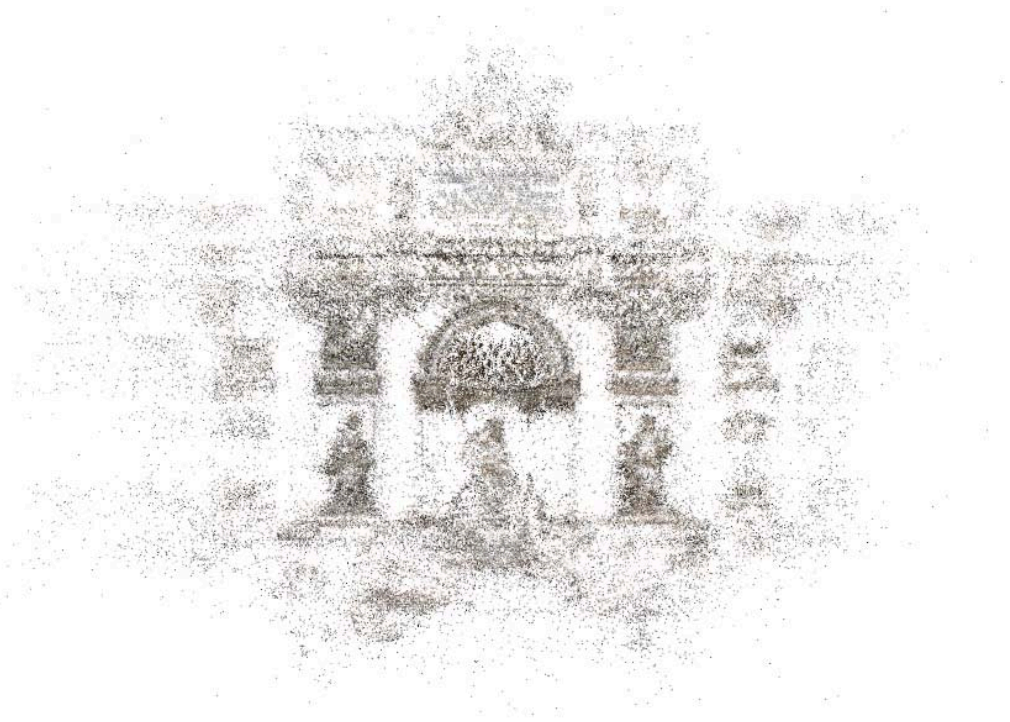

Figure 7.32: A 3D reconstruction of the Fountain di Trevi using our 6-point one calibrated camera problem.

Although this problem looks inferior to the 6-pt problem for two cameras with unknown but equal focal length, described in previous Section 7.1.7, it has several nice and useful properties which the "6-pt equal focal length algorithm" does not have. The most interesting is its resistance to several critical motions which are common in real situations, e.g. when optical axes are parallel or intersecting. These configurations are important since they appear frequently when moving around an object and taking its pictures or taking pictures while walking or from a moving car. Although these configurations are degenerate for the standard 6-pt equal focal length problem, Section 7.1.7, they become tractable when one of the two cameras is fully calibrated.

In this subsection we also show that our algorithms to this problem are useful and practical when combining calibrated images, e.g. taken by a known camera, with images from the Internet. Based on our algorithms we have proposed in [26] a new efficient method for large-scale SfM from unordered data sets downloaded from the Internet e.g. from the Flickr database [1], see Figure 7.32.

Previous solutions

The problem of finding the relative pose between a completely calibrated camera and a camera with unknown focal length was previously studied in [142] where a non-minimal solution was

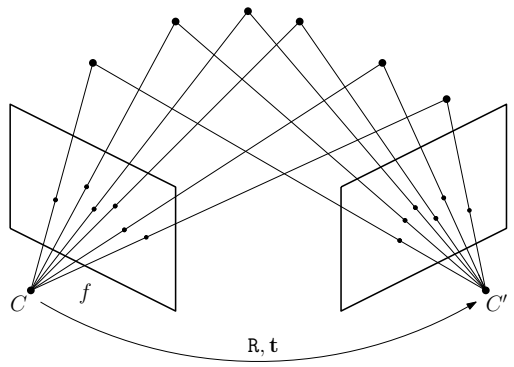

Figure 7.33: Illustration of the 6-point one calibrated camera problem.

proposed. The 7-pt algorithm [61] was first used for computing the fundamental matrix and then the focal length was estimated in a closed-form solution using Kruppa equations.

Here we provide two new minimal solvers for this problem from six point correspondences and compare them with the existing non-minimal solution [142]. Compared to [142], our minimal solution has two advantages. 1) It needs 6 instead of 7 points, which is important for RANSAC and 2) it is more accurate in presence of noise. Our solvers are based on the Gröbner basis and the polynomial eigenvalue methods presented in Sections 4.4 and 5.5. Computational complexity of these algorithms is smaller than for the 6-point equal focal length problem presented in Section 7.1.7.

Let us now formulate the "6-point one calibrated camera problem" as a system of polynomial equations.

Problem formulation

The formulation of the problem of estimating relative pose between a completely calibrated camera and a camera with unknown focal length is very similar to the formulation of the problem for two cameras with equal unknown focal length presented in Section 7.1.7.

In this formulation we again use the epipolar constraint (7.10) to parameterize the fundamental matrix $\mathtt{F}$ with two unknowns x and y as

$$\mathtt{F} = x\,\mathtt{F}_1 + y\,\mathtt{F}_2 + \mathtt{F}_3. \tag{7.98}$$

Since in this case the first camera is calibrated up to an unknown focal length and the second camera is fully calibrated, the essential matrix $\mathtt{E}$ can be written as

$$\mathtt{E} \;=\; \mathtt{K}\mathtt{F}, \tag{7.99}$$

where $\mathtt{K} \simeq diag([f\ f\ 1])$ is a diagonal calibration matrix of the first camera.

To find solutions for unknowns x, y and f we use equations arising from the rank and the

trace constraints which have in this case the form

$$\det(\mathtt{F}) = 0, \tag{7.100}$$

$$2\,\mathtt{F}\,\mathtt{F}^\top\mathtt{Q}\mathtt{F} - \mathrm{trace}(\mathtt{F}\mathtt{F}^\top\mathtt{Q})\,\mathtt{F} = \mathtt{0}. \tag{7.101}$$

Here the matrix equation (7.101) is obtained similarly as equation (7.94) in the "6-point equal focal length problem", i.e. by substituting the expression for the essential matrix (7.99) into the trace constraint (7.7), applying the substitution $\mathtt{Q} = \mathtt{K}\,\mathtt{K}$, and multiplying (7.7) by $\mathtt{K}^{-1}$ from left.

In this case constraints (7.100) and (7.101) bring ten third and fourth order polynomial equations in three unknowns x, y, and $w = f^{-2}$ and 20 monomials. To solve these ten equations, we use both presented methods for solving systems of polynomial equations, i.e. the specialized Gröbner basis method presented in Section 4.4 and the polynomial eigenvalue method described in Section 5.4.

Gröbner basis solution

The Gröbner basis solver for the "6-point one calibrated camera problem" is quite simple and is again based on the single elimination strategy for generating polynomials from the ideal. Therefore it can be easily obtained using our automatic generator of Gröbner basis solvers presented in Chapter 6. The basic steps of the process of creating this Gröbner basis solver which are "hidden" in the automatic generator are as follows.

Identification of basic parameters

In the first step of the process of creating the Gröbner basis solver for this problem we identify the basic parameters of the system of initial 10 equations (7.100) and (7.101) in three unknowns x, y and $w = f^{-2}$.

For this purpose we use the graded reverse lexicographic ordering with $x > y > w$. Using Macaulay 2 and random coefficients from some finite prime field it can be easily found that for this monomial ordering the basis B (4.20) of the quotient ring $A = \mathbb{C}\left[x, y, w\right]/I$ has the form $B = \left\{ \left[1\right], \left[x\right], \left[xy\right], \left[xw\right], \left[y\right], \left[y^2\right], \left[yw\right], \left[w\right], \left[w^2\right] \right\}$ and that the number of solutions of this system is 9.

Construction of the multiplication matrix

Having the basis B we know the form of polynomials q_i (4.115) necessary for constructing the multiplication matrix. We can therefore find an "elimination template", i.e. an admissible Elimination trace, for its construction. For creating the multiplication matrix we have tested all three unknowns x, y and w. Since this is done only once in the "offline phase" it doesn't diminish the overall computational efficiency of the final solver. In this case we have selected multiplication matrix $\mathtt{M}_x$ which results in the simplest solution.

Using the single elimination strategy for generating polynomials from the ideal, i.e. using Algorithm 9, it can be found that obtaining all necessary polynomials q_i (4.115) for constructing the multiplication matrix $\mathtt{M}_x$ calls for generating all monomial multiples of the initial ten polynomial equations up to the total degree five. This means that we need to multiply our 4^{th} degree

polynomial equations by all 1^{st} degree monomials and our 3^{rd} degree polynomial equations by all 2^{nd} degree monomials.

In this way we generated 36 new polynomials which, together with the initial ten polynomial equations (7.100) and (7.101), form a system of 46 polynomials in 56 monomials. Then, we can remove all unnecessary polynomials using Algorithm 10 and we obtain 21 polynomials in 56 monomials.

Since in the matrix representation (4.24) of these 21 polynomials we can remove columns corresponding to monomials which do not have impact on the form of polynomials q_i we finally end up with a 21×30 matrix. After the G-J elimination of this matrix we obtained all polynomials which we need for constructing the multiplication matrix $\mathtt{M}_x$.

Online Gröbner basis solver

The final online solver for solving all "non-degenerate" instances of ten equations (7.100) and (7.101) in three unknowns x, y and $w = f^{-2}$, i.e. for solving the "6-point one calibrated camera problem", does one G-J elimination of a 21×30 coefficient matrix. This matrix contains coefficients which arise from concrete image measurements. After the G-J elimination of this matrix the multiplication matrix $\mathtt{M}_x$ is created from its rows. The solutions for $x, y, w = f^{-2}$ can be then found from eigenvectors of the multiplication matrix $\mathtt{M}_x$.

This online solver is exactly what can be obtained using the automatic generator presented in Chapter 6. The input to the automatic generator, which produces this efficient online solver, is shown in Figure 7.34.

Polynomial eigenvalue solution

The polynomial eigenvalue solver for the 6-pt relative pose problem for one fully and one up to focal length calibrated camera, uses the same ten polynomial equations (7.100) and (7.101) in three unknowns x, y and $w = f^{-2}$ as the Gröbner basis solver. These equations can be rewritten into the matrix form

$$\mathtt{M}\,\mathbf{X} = \mathbf{0}, \tag{7.102}$$

where $\mathtt{M}$ is a 10×20 coefficient matrix from the matrix representation (4.24) and $\mathbf{X} = (wx^3, wyx^2, wy^2x, wy^3, x^3, yx^2, y^2x, y^3, wx^2, wyx, wy^2, x^2, yx, y^2, wx, wy, x, y, w, 1)^\top$ is a vector of 20 monomials.

Unknowns x and y appear in degree three but w appears only in degree one. Therefore, we can "hide" the variable w and rewrite these ten equations as

$$(w\,\mathtt{C}_1 + \mathtt{C}_0)\,\mathbf{v} = \mathbf{0}, \tag{7.103}$$

where $\mathbf{v} = (x^3, yx^2, y^2x, y^3, x^2, yx, y^2, x, y, 1)^\top$ is a 10×1 vector of monomials and $\mathtt{C}_1$, $\mathtt{C}_0$ are 10×10 coefficient matrices such that $\mathtt{C}_1 \equiv (\mathbf{m_1}\ \mathbf{m_2}\ \mathbf{m_3}\ \mathbf{m_4}\ \mathbf{m_9}\ \mathbf{m_{10}}\ \mathbf{m_{11}}\ \mathbf{m_{15}}\ \mathbf{m_{16}}\ \mathbf{m_{19}})$, $\mathtt{C}_0 \equiv (\mathbf{m_5}\ \mathbf{m_6}\ \mathbf{m_7}\ \mathbf{m_8}\ \mathbf{m_{12}}\ \mathbf{m_{13}}\ \mathbf{m_{14}}\ \mathbf{m_{17}}\ \mathbf{m_{18}}\ \mathbf{m_{20}})$, where $\mathbf{m_j}$ is the j^{th} column from the coefficient matrix $\mathtt{M}$.

The formulation (7.103) is directly the generalized eigenvalue problem (5.46), i.e. the polynomial eigenvalue problem of degree one with the regular matrix $\mathtt{C}_0$. Therefore it can be solved by finding the eigenvalues of the matrix $-\mathtt{C}_0^{-1}\mathtt{C}_1$.

```
% symbolic matrices which represent basis of the null space
% of linearized epipolar equation y'*F*x = 0
% matrices F1, F2, F3 will be treated as known
F1 = gbs_Matrix('F1%d%d', 3 ,3);
F2 = gbs_Matrix('F2%d%d', 3 ,3);
F3 = gbs_Matrix('F3%d%d', 3 ,3);

% create fundamental matrix as a linear combination of the null space
% w = 1/focal^2
% variables x, y, w are unknown
syms x y w real;
F = x*F1 + y*F2 + F3;

% build polynomial equations
% rank constraint
eq(1) = det(F);

Q = diag([1 1 w]);

% calibrate fundamental matrix using unknown focal length and use
% the trace constrain for obtained essential matrix
% E = K*F
Ft = transpose(F);

te = 2*F*Ft*Q*F - trace(F*Ft*Q)*F;
eq(2:10) = te(:);

% three unknowns
unknown = {'x' 'y' 'w'};

% known parameters
vars = transpose([F1(:); F2(:); F3(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end

% "kngroups" is a vector where kngroups(k) = l says that k-th
% known variable should be grouped into l-th vector.
kngroups = ones(9,1)*[1 2 3];

% call code generator
[res export] = gbs_CreateCode('ku6ptonoff',eq,known,unknown,kngroups);
```

Figure 7.34: Input Matlab script for our automatic generator of Gröbner basis solvers generator presented in Chapter 6 generating the solver for the "6-point one calibrated camera problem"

After solving (7.103), we obtain 10 eigenvalues, solutions for $w = f^{-2}$ and 10 corresponding eigenvectors $\mathbf{v}$ from which we extract solutions for x and y. Then we use (7.98) to get solutions for $\mathtt{F}$.

Critical motions

Before studying the performance of both presented minimal solutions to the "6-point one calibrated camera problem" on synthetic and real data we briefly discuss in this subsection critical configurations of this problem.

It is known that Euclidean structure can always be recovered from a pair of images acquired by a moving calibrated camera [73]. This is not the case if cameras are calibrated only up to an unknown focal length as critical configurations with constant and varying focal lengths start appearing [73]. The critical motions for a camera pair with varying focal length appear (1) when the principal points are in epipolar correspondence, i.e. the optical axes intersect, (2) whenever the epipolar planes or optical axes are orthogonal. If either principal point coincides with an epipole, both (1) and (2) apply. Having constant focal length provides another useful constraint and hence there are less critical motions. Some of them remain however. For example, (3) arbitrary planar motions when the optical axes lie in the plane (e.g. a driving car with a forward-pointing camera), (4) "turntable rotations" about the intersection point of the two optical axes, when these do not lie in a plane.

In [142] authors demonstrated that none out of (1), (2) and (4) results in degenerated configuration for a fully calibrated and an up to focal length calibrated camera pair. Configuration (3) works too except for the configuration in which the two optical axes are coincident, i.e. pure forward motion. It is possible to prove these results using methods from [73].

Since in [142] these critical motions were not studied in experiments, in the next subsection we show the performance and the comparison of our algorithms with existing ones also in these configurations.

Experiments

In this subsection we evaluate both the generalized eigenvalue solution and the Gröbner basis solution to the "6-point one calibrated camera problem and compare them to the existing non-minimal solution [142]. Since this problem hasn't been solved before from the minimal number of point correspondences we study here critical motions, show real applications and compare the numerical stability and the computational complexity of both presented minimal solvers in more detail.

Synthetic data set

We study the performance of our methods on synthetically generated ground-truth 3D scenes. These scenes were generated as random points in a 3D cube. Each 3D point was projected by a camera with random parameters or parameters testing degenerate configurations (pure translation, etc.). Then, Gaussian noise with standard deviation σ was added to each image point

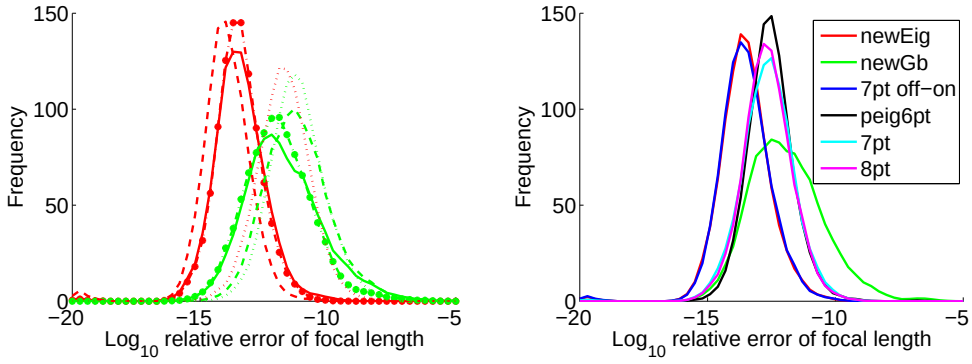

Figure 7.35: (Left) Focal length estimation of the solvers on noise free data set for general motions (solid line), a turntable rotation (dash-dot line), a pure sideways translation (dashed line) and a forward translation with a small sideway motion (dotted line). The generalized eigenvalue solver is shown in red and the Gröbner basis solver in green. (Right) Comparison of the solvers on the general scene, see text.

assuming a 1000×1000 pixel image. By noise $1px$ we will mean Gaussian noise with $\sigma = \frac{1}{3}px$ in all our experiments.

Numerical stability

In this synthetic experiment we study numerical stability of our solvers in various configurations and compare them with other solvers. We focus on focal length estimation because rotation and translation is usually good once we have a good focal length estimate. Figure 7.35 shows the performance of our solver on synthetic noise free scenes in the case of (i) a general motion, (ii) a turntable configuration, (iii) a pure sideways translation and (iv) a forward translation with small sideway motion of cameras such that their optical axes are not coincident. As described in previous subsection, (ii), (iii) and (iv) are critical configurations for camera pairs with constant or varying focal length, i.e. for the 6-point equal focal length problem [134] described in Section 7.1.7 and 7-pt [61] algorithms followed by a focal length extraction.

Figure 7.35 (left) shows that these configurations are not critical for fully calibrated and up to f calibrated camera pairs. Focal lengths estimated in "critical configurations" are equally good as those in a general configuration.

Figure 7.35 (right) shows the stability of the algorithms in general configurations. We compare here both our solvers (newEig, newGB) with the non-minimal off-online solver (7pt on-off) [142] and the polyeig solver of the "6-point equal focal length problem" (peig6pt) presented in Section 7.1.7, the 7-point (7pt) and the normalized 8-point (8pt) algorithms [61]. For the 7-point and the 8-point algorithms we first calculated fundamental matrices and then extracted focal lengths using the Bougnoux method [17]. In case of the 8-point algorithm, we used all image measurements to calculate epipolar geometry. Figure 7.35 (right) shows that our generalized eigenvalue and the non-minimal off-online algorithm [142] give the best estimates of f, but all methods perform almost equally.

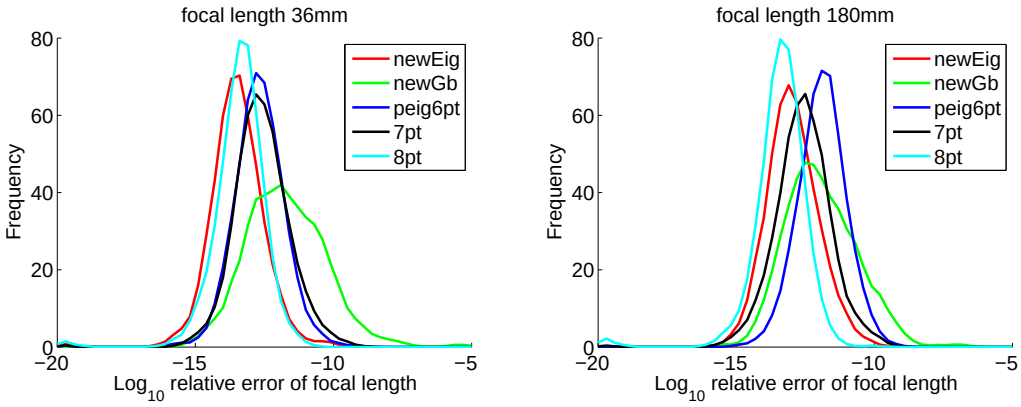

Figure 7.36: Focal length estimation of the solvers on noise free data set for general motion and two different focal lengths $36mm$ (left) and $180mm$ (right).

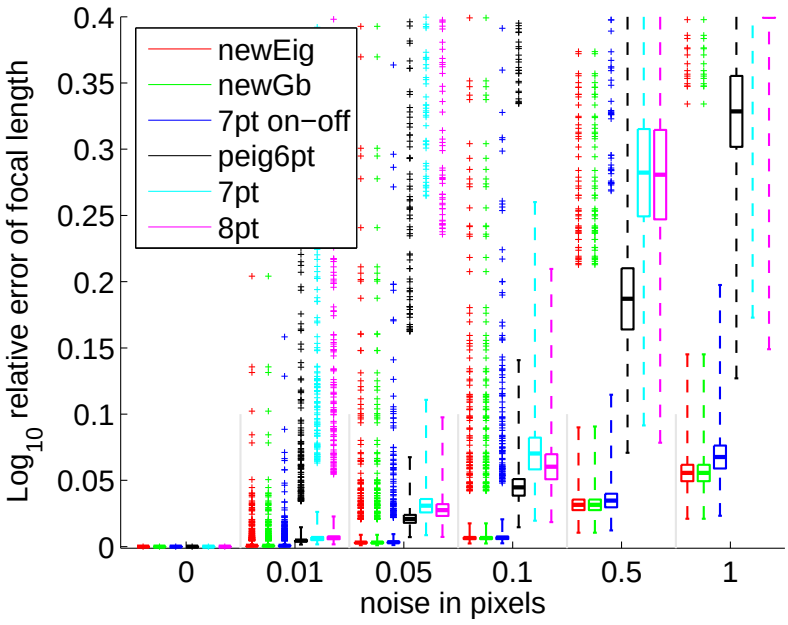

Figure 7.37: The performance of our solvers for general motions with growing noise level. Here noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$, see text.

We have also made a comparison of our algorithms with the non-minimal off-online algorithm for all "critical configurations" like in Figure 7.35 (left). The results of the non-minimal algorithm for all configurations where almost the same as the results of our generalized eigenvalue solver and therefore we do not show them here.

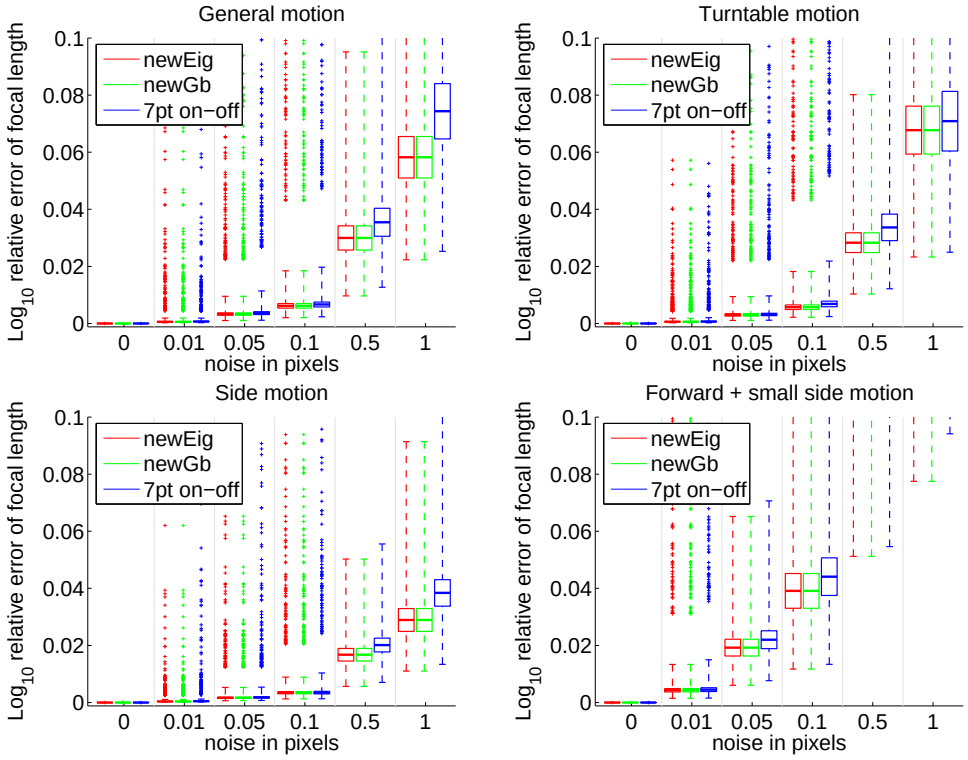

Figure 7.38: The performance of our solvers in the 4 studied special configurations, see text. Noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

In the next experiment we have studied the performance of all algorithms in the case of increasing focal lengths. Figure 7.36 shows the results for the experiment with increasing focal lengths on a noise free data set in a general configuration. Figure 7.36 shows the results for two different focal lengths ($36mm$ and $180mm$) only, however the results for all focal lengths from $25mm$ to $300mm$ were similar. We can conclude that the stability of both our solvers is independent from the true focal length.

Next experiment shows the quality of the focal length estimation for general camera configurations when adding noise to image measurements. First, we fixed the focal length of the first camera to $30mm$ and of the second camera to $50mm$. Then we generated 1000 random camera poses for each tested noise. Results for the general camera motion are shown in Figure 7.37. Our solvers (newEig, newGb) give almost the same results here and they are performing very well even with one pixel noise level. The results of the non-minimal algorithm (7pt on-off) were for smaller noise levels similar to the results of our algorithms, however for larger noise levels the performance of our algorithms was slightly better.

The previous experiment has shown the results for a general camera motion. A different situation occurs when we are testing these solvers in their critical configurations. Adding a small perturbation to the image measurements helps removing the degeneracy and lets solvers

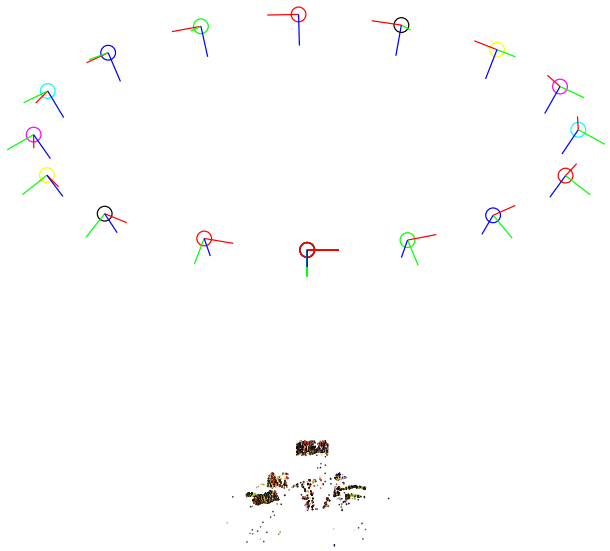

Figure 7.39: 3D reconstruction of cameras in a "turntable configuration" by our method. This is a degenerate configuration for general 6-pt algorithms which often occurs with causal photographers taking pictures of 3D objects.

find an approximate solution. However, the experiments show that the 6-point equal focal length problem described in Section 7.1.7, the 7-point [61] and the 8-point [61] solvers failed to deliver any real solution in more than 90% of all tests. Hence Figure 7.38 shows plots only for our solvers and the non-minimal off-online solver in turntable motion, side motion and forward translation with small side motion and non-coincident optical axes. Our solvers failed to deliver real solutions in less than 2 cases in 1000 calls. Again the results of the non-minimal algorithm were similar to the results of our algorithms for smaller noise levels and all configurations and for larger noise levels the performance of our algorithms was slightly better.

Real experiments - critical motions

In the real data experiments we have aimed at the critical motions described in previous subsection. We captured a set of images of a non-planar scene in a "turntable" configuration and with a sideways moving camera. Figure 7.39 shows a reconstruction using our autocalibration method described in our paper [26] without any additional numerical improvements.

Figure 7.40 (left) shows estimated focal lengths using different algorithms. Note that this is a critical configuration for the standard 6-point equal focal length and the 7-point algorithms. We obtained results for these solvers since image correspondences are not measured perfectly what

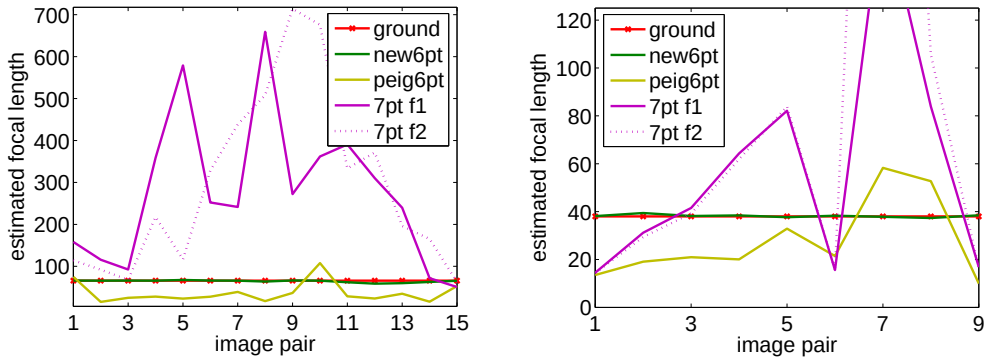

Figure 7.40: The estimated focal length using different algorithms for turn table sequence (left) and side motion sequence (right). For the 7-pt algorithm we extracted two focal lengths - 7pt f1 (solid line) and 7pt f2 (dashed line)

helped finding solutions which were close to a critical configuration. However, it can be seen from Figure 7.40 (left) that estimated focal lengths are far from the ground truth.

Results for the sideways motion configuration are similar to those obtained for the "turntable" configuration. The estimated focal lengths are in Figure 7.40 (right). The results of the non-minimal algorithm were similar to the results of our solvers (new6pt) and therefore are not shown in the figure.

Note that the standard 6-point equal focal length algorithm uses the same focal length in both cameras. Since the difference between estimated focal length and the ground value is large, one cannot get correct Euclidean reconstruction. Hence, building structure from motion using such partial reconstructions is a hard problem and it is not possible to obtain a good reconstruction, e.g., by using bundle adjustment methods [61] without specifying additional constraints, e.g., constant focal length in the whole sequence. We failed to reconstruct our "turntable" sequence even with a robust state of the art systems such as PhotoSynth [107].

Real experiments – images from the Internet

In this experiment we used our 6-point one calibrated camera solvers and the SfM method described in our paper [26] on a set of images downloaded from the Internet. We downloaded 2550 images of Fountain di Trevi and 4000 images of Notre Dame de Paris from Flickr [1] database. In such a huge database of images, it is not a problem to find an image with focal length stored in EXIF and use it as a seed.

We extracted SURF [12] feature points and descriptors of all images and used [77] to obtain image matches. For the best 50 images for each seed we extracted tentative correspondences as points where the best descriptor dominates by 20% over the second best descriptor [111]. Then we used our reconstruction pipeline presented in paper [26] to register images. We did not generate additional seeds and did not use bundle adjustment in this step.

The Fountain dataset contained about $10\%(239)$ seed images with $35mm$ equivalent focal length stored in EXIF. About 14% of them did not contain Fountain scene and 11% were re-

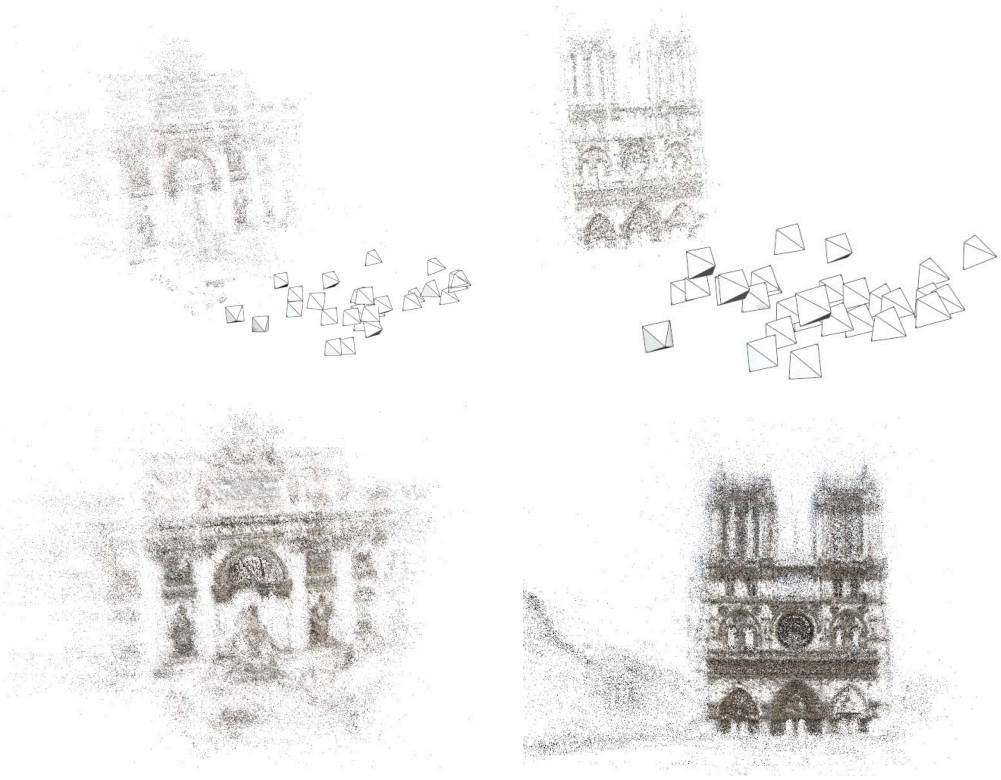

Figure 7.41: Single seed reconstructions on the top, full reconstructions on the bottom.

jected as wrongly calibrated. Numbers for the Notre Dame sequence were similar. A 3D reconstruction of both dataset from a single seed image are shown in Figure 7.41 (top) and the full reconstructions in Figure 7.41 (bottom).

Interesting questions

Here we try to answer questions you may ask yourself:

- *Does the method work when all six point correspondences are projections of the points from a plane in the 3D space?*

  No, this is a degenerated configuration [73].

- *What happens if the calibration of the calibrated camera is inaccurate or even unknown?*

  Figure 7.42 (left) tries to answer this question. We have generated synthetic scenes and used our solver to find unknown focal length (unk fl) without doing any calibration of the first camera. We were interested in the relation between the evaluated and the ground truth focal lengths.

  We found that results for general configurations are almost random. However, the experiment has shown that the estimated focal length is actually the ratio of the two ground truth

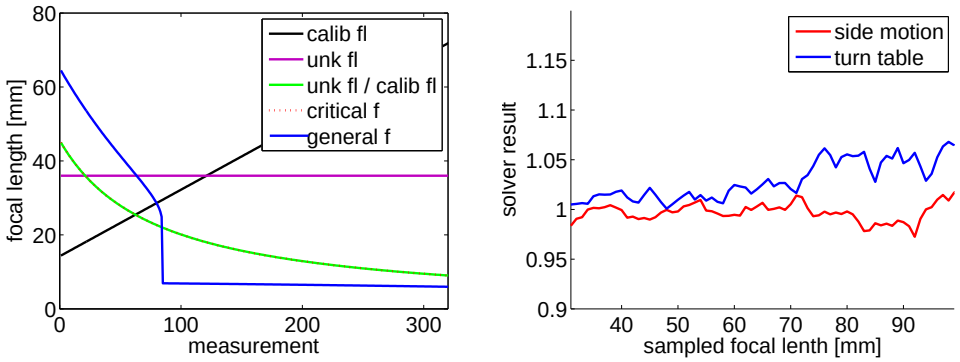

Figure 7.42: (Left) The True focal length of the calibrated camera (calib fl) is increasing. Focal length of the second camera is unknown but constant (unk fl). We have found that estimated focal lengths for critical motion (critical f) is identical to the ratio of the ground truth focal lengths (unk fl/calib fl). General motion (general f) configuration is shown in blue. (Right) The "sampling" experiment on the real data sets with constant focal length camera. Graphs show the ratio of the estimated and the sampled focal lengths. The ground truth ratio is equal to one.

focal lengths for cameras in critical configurations. This experiment shows why "sampling" approach does not help finding absolute focal length given the ratio ρ of focal lengths in a critical configuration. Basically, one gets ρf for every sampled focal length f. In case of the turntable and sideways motion sequence from our real data set, where cameras have constant focal lengths, we obtained results corresponding to the ratio close to one, Figure 7.42 (right).

- *Given the ratio of focal lengths, can we emulate the 6-pt equal focal length algorithm [134] and avoid degenerated configurations ?*

  Using the 6-point one calibrated camera algorithm we can emulate the 6-point equal focal length algorithm by sampling the focal length of calibrated camera and testing if the ratio of estimated and sampled focal lengths is close to the given ratio. As expected, such algorithm will not work for critical motions as shown above.

Computational complexity

Our 6-point one calibrated camera Gröbner basis solver has to perform a single G-J elimination of a 21×30 matrix in order to build the multiplication matrix. We take sparseness of the matrix into account and hence elimination time is negligible comparing to the eigenvector computation. Then the eigenvalues of a 9×9 matrix are computed. The C++ version of this solver runs about $36\mu s$ on Intel i7 Q720 notebook.

The generalized eigenvalue solver for this problem does not have to perform any elimination but has to calculate generalized eigenvectors. This involves the computation of the inverse of

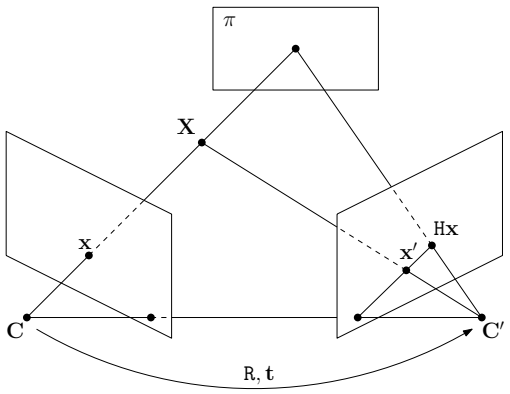

Figure 7.43: Illustration of the plane + parallax problem for camera with unknown focal length.

one 10×10 matrix and then the computation of eigenvectors of a matrix of the same size. This solver runs about $51\mu s$.

Concerning the number of solutions, we found that in average 3.5 out of nine solutions to the 6-point one calibrated camera problem were real and positive. We calculated this by measuring the number of real, positive solutions in 10000 scenes generated in four configurations (i),(ii),(iii) and (iv) as defined in subsection Numerical stability. By evaluating the number of solution separately for different configurations we have obtained approximately the same results.

7.1.9 Plane+parallax for cameras with unknown focal length

The next relative pose problem which we present here is the minimal problem of estimating epipolar geometry and unknown focal length from images of four points lying on a plane and one off-the-plane point. This problem is known as the "plane + parallax" problem and has not been solved for cameras with unknown focal length before. Such problem is important in situations where some dominant plane is present in the scene, like in 3D reconstructions of cities or other man-made objects.

Previous solutions

The plane + parallax problem was previously solved only for uncalibrated cameras [61]. In this case the fundamental matrix $\mathtt{F}$ can be computed uniquely from the images of six points, four of which are coplanar and two are off the plane.

The plane + parallax problem for uncalibrated cameras results in quite simple algorithm [61]. It is because the images of the four coplanar points define a homography $\mathtt{H}$, and the two off-the-plane points can be used to determine the epipole $\mathbf{e}'$ as the intersection of two lines. The fundamental matrix $\mathtt{F}$ can be then computed as $\mathtt{F} = [\mathbf{e}']_\times \mathtt{H}$.

In this section we present a new minimal solution to the plane + parallax problem for camera with constant unknown focal length. This problem has its applications in situations where we assume square pixels and principal point in the center of the image.

Problem formulation

First we formulate the problem of estimating epipolar geometry, i.e. the essential matrix $\mathtt{E}$ and the unknown focal length, from images of five points, four of which are coplanar and one is off the plane.

The images of four coplanar points define the homography $\mathtt{H}$. For all 3D points $\mathbf{X}$ lying on the plane defined by these four points it holds $\mathbf{x}' = \mathtt{H}\mathbf{x}$, where $\mathbf{x}'$ and $\mathbf{x}$ are projections of point $\mathbf{X}$ in the first and second view. The important property holds for points which are not on this plane. The image $\mathbf{x}'$ of some off-the-plane 3D point $\mathbf{X}$ in the second view and the point $\tilde{\mathbf{x}}' = \mathtt{H}\mathbf{x}$, mapped by the homography $\mathtt{H}$, lie on the epipolar line of $\mathbf{x}$, since both are images of points on the ray through $\mathbf{x}$, see Figure 7.43. Therefore $l'_x = \mathbf{x}' \times \mathtt{H}\mathbf{x}$ is an epipolar line in the second view. Thus we can write

$$\mathbf{e}' = \mathbf{x}' + s\mathbf{v}, \tag{7.104}$$

where $\mathbf{v}$ is the normalized direction vector of the epipolar line l'_x and s is an unknown parameter. Another possibility which holds also for epipole at infinity is to write

$$\mathbf{e}'^{\top} \left(\mathbf{x}' \times \mathtt{H}\mathbf{x}\right) = 0, \tag{7.105}$$

which means that the epipole $\mathbf{e}'$ lies on the epipolar line $\mathbf{x}' \times \mathtt{H}\mathbf{x}$. The epipoles at infinity appear in the case of special motion, where the translation is parallel to the image plane, and the rotation axis is perpendicular to the image plane. Since this motion doesn't appear quite often we will present here the solver for the parameterization (7.104). The solver for the parameterization (7.105) can be obtained in a similar way.

It is known [61] that for the fundamental matrix $\mathtt{F}$ it holds

$$\mathtt{F} = \left[\mathbf{e}'\right]_{\times} \mathtt{H}. \tag{7.106}$$

Therefore once the epipole $\mathbf{e}'$ end the homography $\mathtt{H}$ induced by any plane are estimated, the fundamental matrix $\mathtt{F}$ can be computed uniquely. Note that this fundamental matrix is already singular.

The algorithm [61] for computing $\mathtt{F}$ given the homography induced by a plane uses images of six points, four of which are coplanar and two are off the plane. A focal length can be computed from the fundamental matrix $\mathtt{F}$ [61], but it is known as a complicated problem. It is because the estimated focal lengths become often complex since [61] uses fully projective model and associated cameras are often skewed. In contrast, this is not a problem for our algorithm where the focal length is calculated together with the essential matrix. In our case we have only one point off the plane. Therefore we can only parameterize $\mathbf{e}'$ with one unknown parameter s using equation (7.104).

Assume that both cameras are calibrated up to an unknown common focal length f. Then for the essential matrix $\mathtt{E}$ it holds

$$\mathtt{E} = \mathtt{K}^{\top}\mathtt{F}\mathtt{K}, \tag{7.107}$$

where $\mathtt{K} \simeq diag([f\ f\ 1])$ is a diagonal calibration matrix.

In our solution we use the trace constraint

$$2\,\mathtt{E}\mathtt{E}^{\top}\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^{\top})\,\mathtt{E} = \mathtt{0}, \tag{7.108}$$

which says that two singular values of the essential matrix $\mathtt{E}$ are equal and the third is zero.

If we express the essential matrix $\mathtt{E}$ using equations (7.104), (7.106) and (7.107), the matrix equation (7.108) gives us nine fifth degree polynomial equations in two unknowns s and f or in $w = 1/f^2$, from which six are linearly independent. Therefore we have six equations in two unknowns which result in five solutions for f and s.

Gröbner basis solution

Next we describe all basic steps of the process of creating an efficient Gröbner basis solver for solving this plane + parallax focal length problem using our specialized Gröbner basis method presented in Section 4.4. We will call this solver the "4+1 plane + parallax focal length solver" because we have four coplanar points and one off-the-plane point. To construct the solver, we use the single elimination strategy for generating polynomials from the ideal. The efficient solver can be created using the automatic generator of Gröbner basis solvers presented in Chapter 6.

Identification of basic parameters

In the first step of the offline phase of creating an efficient solver for the first parametrization (7.104) of the plane + parallax problem we identify the basic parameters of the system of nine equations (7.108) in two unknowns s and $w = f^{-2}$ from which six are linearly independent.

In this case we use the graded reverse lexicographic ordering with $s > w$.

With this ordering and using Macaulay 2 we can find that the basis B (4.20) of the quotient ring $A = \mathbb{C}[s, w]/I$ has the form $B = \left\{[1], [s], [w], [sw], \left[w^3\right]\right\}$ and the input system has 5 solutions.

Construction of the multiplication matrix

Having the basis B we know the form of the polynomials q_i (4.115) necessary for constructing the multiplication matrix. Therefore in the next step we can find the an admissible Elimination trace for its construction, see Definition 21. For creating the multiplication matrix we have tested both unknowns s and w and finally we have selected to create the multiplication matrix $\mathtt{M}_s$.

In the case of this plane + parallax problem and the first parametrization of the epipole $\mathbf{e}'$ (7.104) the final admissible Elimination trace is quite simple. Using Algorithm 9, i.e. using the single elimination strategy for generating polynomials from the ideal, we found that to obtain all necessary polynomials q_i (4.115) for creating the multiplication matrix $\mathtt{M}_s$, we have to generate all monomial multiples of the initial polynomial equations up to total degree six. This results in 27 polynomials in 28 monomials. After removing unnecessary polynomials and also unnecessary monomials using Algorithm 10 there remain only 10 polynomials in 15 monomials.

After the G-J elimination of the corresponding 10×15 matrix, all necessary polynomials for creating the multiplication matrix $\mathtt{M}_s$ are obtained.

Online Gröbner basis solver

The online solver for solving all "non-degenerate" instances of equations (7.108) with the first parametrization of the epipole $\mathbf{e}'$ (7.104), i.e. for solving the 4+1 plane + parallax problem for cameras with unknown focal length, does one G-J elimination of the 10×15 coefficient

matrix. After this G-J elimination, the multiplication matrix $\mathtt{M}_s$ is created from rows which correspond to polynomials q_i (4.115). Solutions for our variables s and $w = f^{-2}$ are obtained from eigenvectors of this multiplication matrix $\mathtt{M}_s$. Finally the essential matrix $\mathtt{E}$ is obtained by backsubstituting solutions for s to equations (7.104), (7.106), (7.107) and (7.108).

Input to the automatic generator

The online solver for solving all "non-degenerate" instances of the input nine equations (7.108) in two unknowns and the parametrization of the epipole $\mathbf{e}'$ (7.104) can be created using our automatic generator presented in Chapter 6.

As we have already mentioned in Chapter 6 for some problems it is necessary to instantiate coefficients of initial polynomial equations with exact numbers from $\mathbb{Z}_p$. It is because random coefficients may not fulfill some coefficient dependencies and the automatically generated solver may not work correctly or even can not be generated, e.g. system with random coefficients may not have a solution.

This is the case also of the "4+1 plane + parallax focal length solver". It is because from the initial nine equations (7.108) in two unknowns s and w only six are linearly independent and the coefficient of these six equations fulfill several constraints. These dependences may not be fulfill for random coefficients from $\mathbb{Z}_p$.

Therefore, we will present here the input to our automatic generator that includes also instantiating of input equations with correct coefficients from $\mathbb{Z}_p$. These coefficients fulfill all dependencies. The instantiating is done using the function

```
cfg.eqinstance = pp5pttestfocal_inst(cfg);
```

The form of this function can be seen in Figure (7.44). This function first generates a scene containing four coplanar points and some off-the-plane points, projects them on two cameras and computes a homography between projections of the four coplanar points. This is done in $\mathbb{Z}_p$ and is included in the function

```
[g m mp mpp] = Generate2ViewSetupZp(1, prime);
```

Then the function `pp5pttestfocal_inst(cfg)` parametrizes the epipole using projections of one off-the-plane point and using homography computed in $\mathbb{Z}_p$. Finally, the function creates the fundamental matrix using the equation (7.106) and generates nine input equations with coefficients form $\mathbb{Z}_p$ that fulfill all dependencies.

The final input to our automatic generator for the plane + parallax focal length problem can be seen in Figure (7.45). With this input, the automatic generator automatically performs all steps of the offline phase, i.e. the identification of basic parameters and finding an Elimination trace, and as an output it gives the online solver for solving the 4+1 plane + parallax focal length problem.

Polynomial eigenvalue solution

The polynomial eigenvalue solution to the plane +parallax focal length problem is very simple. It starts with six linearly independent equations in two unknowns s and w obtained from the trace

```
function [eq] = pp5pttestfocal_inst(cfg)

    prime = cfg.prime;

    [g m mp mpp] = Generate2ViewSetupZp(1, prime);

    H = g.H12;

    % point outside the plane
    x = g.m3d{1}(:,1);
    xp = g.m3d{2}(:,1);

    % vector of the epipolar line
    v = gbs_Vector('v', 3);

    syms s w;

    % epipole parametrized as a point on the epipolar line,
    % here v = cross(Hxp,x)
    e = mod(expand(xp+s*(xp(3)*H*x-xp*H(3,:)*x)), prime);
    % e = xp+s*v;
    Ex = vec2skew(e);

    Q = diag([1,1,w]);

    % fundamental matrix cross(e,H) has already det=0

    F = mod(expand(Ex*H), prime);

    % trace constraint
    Ft = transpose(F);
    FQFtQ = mod(mod(expand(F*Q),prime)*mod(expand(Ft*Q),prime),prime);
    te = mod(mod(expand(2*(FQFtQ)*F),prime) -
    mod(expand(trace(FQFtQ)*F), prime), prime);

    % build equations
    eq(1:9) = te(:);
end
```

Figure 7.44: Function pp5pttestfocal_inst which instantiates the input equations with correct coefficients from $\mathbb{Z}_p$ in the plane + parallax focal length problem.

```
cfg = gbs_InitConfig();

% create symbolic matrices
% Homography matrix is computed from 4 point on the plane
H = gbs_Matrix('H%d%d', 3 ,3);

% point outside the plane
x = gbs_Vector('x', 3);
xp = gbs_Vector('xp', 3);

% vector of the epipolar line
v = gbs_Vector('v', 3);

syms s w;

% epipole parametrized as a point on the epipolar line,
% here v = cross(Hxp,x)
e = xp+s*(xp(3)*H*x-xp*H(3,:)*x);
Ex = vec2skew(e);

% fundamental matrix cross(e,H) has already det=0
F = Ex*H;
% trace constraint
Ft = transpose(F);
Q = diag([1,1,w]);
FQFtQ F*Q*Ft*Q
te = 2*(FQFtQ)*F - trace(FQFtQ)*F;

% build equations
eq(1:9) = te(:);

% collect known & unknown variables
vars = transpose([H(:); x(:); xp(:)]);
known = {};
for var = vars
    known = [known {char(var)}];
end
unknown = {'s' 'w'};

kngroups = [1 1 1 1 1 1 1 1 1 2 2 2 3 3 3];

% call code generator
cfg.eqinstance = pp5pttestfocal_inst(cfg);
[res export] = gbs_CreateCode('pp5ptfocal', eq, known, unknown,
kngroups, cfg);
```

Figure 7.45: Input Matlab script for our automatic generator of Gröbner basis solvers presented in Chapter 6 generating the solver for the 4+1 plane + parallax focal length problem.

constraint (7.108). These equations consist of monomials $w^2s^3, ws^3, w^2s^2, s^3, ws^2, w^2s, s^2, ws, w^2, s, w$ and 1.

Variable s appears in degree three and w only in degree two. Therefore, we have selected $\alpha = w$ in the PEP formulation (5.44). Then, these six equations can be rewritten as

$$\left(w^2\mathtt{C}_2 + w\,\mathtt{C}_1 + \mathtt{C}_0\right)\mathbf{v} = \mathbf{0}, \tag{7.109}$$

where $\mathbf{v}$ is a 4×1 vector of monomials, $\mathbf{v} = (s^3, s^2, s, 1)^\top$ and $\mathtt{C}_2, \mathtt{C}_1$ and $\mathtt{C}_0$ are 6×4 coefficient matrices.

Since we have more equations than the monomials, we can select 4 equations from them, such that the resulting system will have a small condition number. The resulting formulation will be in this case the QEP which can be solved using the methods presented in Section 5.4. After solving QEP (7.97), by transforming it to a GEP, we obtain 8 eigenvalues, solutions for $w = f^{-2}$, and 8 corresponding eigenvectors $\mathbf{v}$ from which we extract solutions for s. The essential matrix $\mathtt{E}$ is obtained by backsubstituting obtained solutions for s to equations (7.104), (7.106), (7.107) and (7.108).

Experiments

We evaluated both our solutions to the plane+parallax focal length problem on synthetic data with various levels of noise and focal lengths. We aimed at studying the numerical stability and behavior of presented solutions in the presence of noise.

In our synthetic experiments we use synthetically generated groundtruth 3D scenes. These scenes were generated using 500 randomly distributed 3D points on a plane and 500 out of the plane points randomly distributed in a 3D cube. Each 3D point was projected by a camera with random feasible orientation and position and random or fixed focal length. Gaussian noise with standard deviation σ was added to the projected image points assuming a 1000×1000 pixel image. By noise $1px$ we will mean Gaussian noise with standard deviation $\sigma = \frac{1}{3}px$ in all our experiments.

Synthetic experiments

To study the numerical stability of the two proposed solutions to the plane+parallax focal length problem we extracted 1000 random samples of four coplanar points and one off the plane point. For each sample, first the homography was computed from four-tuple, then the focal length and the fundamental matrix were estimated by the proposed algorithms.

In evaluation of both our solvers for the plane+parallax focal length problem, we focused on the value of estimated focal length. We measured relative focal length error $(f - f_{gt})/f_{gt}$, where f is an estimated focal length and f_{gt} denotes the ground truth.

Figure 7.46 compares the numerical stability of both proposed solvers, the Gröbner basis and the polynomial eigenvalue solver, for noise free data. It can be seen that the precision and the numerical stability of both solvers is very similar and sufficiently good.

In the second synthetic experiment we studied the robustness of our algorithm to increasing levels of Gaussian noise added to image measurements. In this case the level of noise varied from 0 to 2 pixels and the ground truth focal length was $f_{gt} = 1.5$.

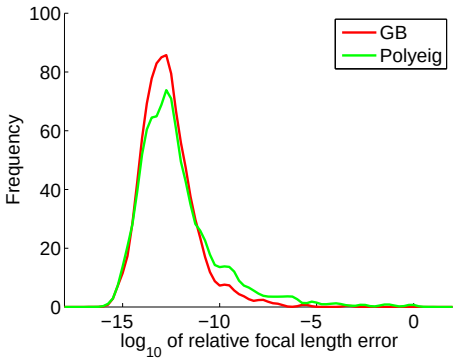

Figure 7.46: The log_{10} relative focal length error for the plane+paralax focal length problem. Here GB (red) denotes the Gröbner basis solution and Polyeig (green) denotes the polynomial eigenvalue solution.

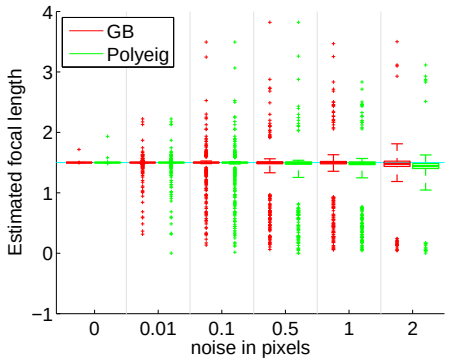

Figure 7.47: Estimated focal length as a function of noise, ground truth $f_{gt} = 1.5$. Boxes contain values from 25% to 75% quantile. Noise $1px$ means Gaussian noise with $\sigma = \frac{1}{3}px$.

Figure 7.47 shows 500 focal lengths computed for each noise level by the two minimal 4+1 plane+parallax focal length solvers, the Gröbner basis solver (red) and the polynomial eigenvalue solver (green), as a function of noise. In this experiment the focal length for each sample was obtained by selecting the real root closest to the ground truth value $f_{gt} = 1.5$. In practice, RANSAC or kernel voting can be used to select the correct solution among all computed solutions.

The results are presented by the Matlab function *boxplot* which shows values 25% to 75% quantile as a box with horizontal line at median. The crosses show data beyond 1.5 times the interquartile range. The median values for both presented solvers are very close to the ground truth value $f_{gt} = 1.5$ for all noise levels.

Computational complexity

Our Gröbner basis solver for the plane + parallax focal length problem is very simple and fast. This solver needs to perform a single G-J elimination of a 10×15 matrix and then to compute eigenvalues of a 5×5 matrix. The C++ version of the solver generated using our automatic generator runs about $9.5\mu s$ on Intel i7 Q720 notebook. Note that this time can be improved using characteristic polynomial [41] and Sturm sequences [66] instead of computing eigenvalues using standard numeric methods.

The polynomial eigenvalue solver for this plane + parallax focal length problem needs to compute the inverse of one 4×4 matrix and then to compute eigenvalues of a 8×8 matrix. This solver runs on the same hardware as the Gröbner basis solvers about $18.5\mu s$.

8 Conclusion

"As for the search for truth,
I know from my own painful searching,
with its many blind alleys,
how hard it is to take a reliable step, be it ever so small,
towards the understanding of that which is truly significant."

Albert Einstein

The main goal of this thesis was to apply the algebraic geometry methods for solving systems of polynomial equations to design efficient specific solvers for solving particular systems of equations, automate these algebraic methods, and use them to efficiently solve some open minimal problems in computer vision, and to improve already existing important minimal solvers.

In this thesis we have studied two classes of standard algebraic techniques for solving systems of polynomial equations; the Gröbner basis and the resultant based techniques. These techniques are general, i.e. they are developed for general systems of polynomial equations and do not make any special assumptions about these systems.

However, many problems that lead to systems of polynomial equations have some special properties and also special requirements, e.g., in computer vision we usually need to be able to solve many "consistent" instances of a particular problem, i.e. systems of polynomial equations of the "same form", and we need to do it very fast.

Therefore, in this thesis we have focused on these properties and requirements, and based on them, we have proposed modifications of the two studied general methods, i.e. the specialized Gröbner basis method presented in Section 4.4 and the specialized resultant based method for solving systems of polynomial equations presented in Section 5.5. Both these specialized methods can be used to create efficient specific solvers for solving systems of polynomial equations that are in the form of one "representing system", and fulfill requirements of many problems appearing in computer vision.

The main difference between the proposed specialized methods and the general methods is that the specialized methods use the structure of the system of polynomial equations F representing a particular problem to design an efficient specific solver for this problem. These specialized methods consist of two phases. In the first phase, preprocessing and computations common to all systems of polynomial equations in the form of the "representing system" F are performed and an efficient specific solver is constructed. For a particular problem this phase needs to be performed only once.

In the second phase, the efficient specific solver is used to solve concrete instances of the particular problem, i.e. systems in the form of the "representing system" F with concrete coefficient from $\mathbb{C}$. The efficient specific solver is not general and solves only systems of polynomial equations in the form of F, see Definition 20. However, the specific solver is faster than a general solver and suitable for applications that appear in computer vision.

In the case of the specialized Gröbner basis method presented in Section 4.4 we have summarized and extended the Gröbner basis method previously used to manually create solvers for some computer vision problems [131, 132, 135, 134]. We have proposed several extensions, such as the identification of the form of necessary polynomials, several new strategies for generating these polynomials and procedures for removing unnecessary polynomials. These extensions allowed us to create smaller, more efficient and more stable solvers than previously manually created solvers [131, 132, 135, 134] and to solve new minimal problems.

The second specialized method presented in this thesis is based on the hidden variable resultants and the polynomial eigenvalue problems [9]. In Section 5.5 we have proposed several new strategies for transforming the initial system of polynomial equations to the polynomial eigenvalue problem [9] and for reducing the size of this problem. Thanks to these proposed strategies we were able to create efficient and numerically stable polynomial eigenvalue solvers for several minimal computer vision problems.

The nice property of both presented specialized methods is the fact that they can be easily automated and therefore used even by non-experts to solve problems leading to systems of polynomial equations.

In Chapter 6 we have proposed the automatic generator of efficient specific solvers based on the presented specialized Gröbner basis method. We have shown that this automatic generator can be used to create efficient solvers for problems leading to relatively complicated systems of polynomial equations and that it usually results in smaller, more efficient and more stable solvers than manually created ones.

In Chapter 7 we have demonstrated the usefulness of both proposed specialized methods presented in Sections 4.4 and 5.5 and the usefulness of our automatic generator by providing new efficient and numerical stable solutions to several important relative pose problems, most of them previously unsolved.

We have presented new solutions to the following five minimal problems that haven't been solved before:

- the problem of simultaneous estimation of the fundamental matrix and a common radial distortion parameter for two uncalibrated cameras from eight image point correspondences presented in Section 7.1.3,
- the problem of simultaneous estimation of the fundamental matrix and radial distortion parameters for two uncalibrated cameras with different radial distortions from nine image point correspondences presented in Section 7.1.4,
- the problem of simultaneous estimation of the essential matrix and a common radial distortion parameter for two partially calibrated cameras and six image point correspondences presented in Section 7.1.5,

- the 6-point relative pose problem for one fully calibrated and one up to focal length calibrated camera presented in Section 7.1.8, and
- the problem of estimating epipolar geometry and unknown focal length from images of four points lying on a plane and one off-the-plane point, i.e. the "plane + parallax" problem for cameras with unknown focal length presented in Section 7.1.9.

Beside these new solutions we have provided new polynomial eigenvalue and Gröbner basis solutions to two well known important relative pose problems:

- the 5-point relative pose problem for two calibrated cameras presented in Section 7.1.6, and
- the 6-point relative pose problem for two cameras with unknown equal focal length presented in Section 7.1.7.

We have demonstrated the quality and effectiveness of all proposed solvers on synthetic and real data.

Bibliography

[1] Flickr - http://www.flickr.com/.

[2] 2D3. Boujou - http://www.2d3.com.

[3] J. Abbott and A.M. Bigatti. CoCoALib: a c++ library for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it/cocoalib.

[4] F. S. Acton. *Numerical methods that work*. Mathematical Association of America, 1990.

[5] W.W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*. Graduate Studies in Mathematics. American Mathematical Society, 1994.

[6] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2003.

[7] E. A. Arnold. Modular algorithms for computing gröbner bases. *Journal of Symbolic Computation*, 35(4):403–419, 2003.

[8] W. Auzinger and H. J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *International Conference on Numerical Mathematics*, 1988.

[9] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Van Der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[10] L. Ballan, G. J. Brostow, J. Puwein, and M. Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. In *ACM SIGGRAPH'10)*, pages 1–11, 2010.

[11] J. P. Barreto and K. Daniilidis. Fundamental matrix for cameras with radial distortion. In *ICCV'05*, volume 1, pages 625–632, 2005.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[13] D. Bayer and M. Stillman. A criterion for detecting m-regularity. *Inventiones Mathematicae*, 87:1–11, 1987.

[14] T. Becker, H. Kredel, and V. Weispfenning. *Gröbner bases: a computational approach to commutative algebra*. Springer-Verlag, London, UK, April 1993.

[15] W. Bosma, J. Cannon, and C. Playoust. The magma algebra system i: the user language. *Journal of Symbolic Computation*, 24(3-4):235–265, October 1997.

[16] W. Bosma and J.J. Cannon. *Handbook of MAGMA Functions*. Computational Algebra Group, School of Mathematics and Statistics, University of Sydney, 1996.

[17] S. Bougnoux. From projective to euclidean space under any practical situation, a criticism of self-calibration. In *ICCV'98*, 1998.

[18] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475-511, 2006.

[19] B. Buchberger. An algorithmic criterion for the solvability of a system of algebraic equations. *Aequationes Mathematicae*, 4:374383, 1970. (German).

[20] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of groebner bases. In *EUROSAM'79*, pages 3–21, 1979.

[21] B. Buchberger. *Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory.* Reidel Publishing Company, Dodrecht - Boston - Lancaster, 1985.

[22] B. Buchberger. Bruno buchberger's phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, pages 475–511, 2006.

[23] B. Buchberger and F. Winkler. *Gröbner Bases and Applications*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1998.

[24] M. Bujnak. *Algebraic solutions to absolute pose problems*. PhD thesis, Czech Technical University in Prague, 2012.

[25] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *CVPR'08*, 2008.

[26] M. Bujnak, Z. Kukelova, and T. Pajdla. 3d reconstruction from image collections with a single known focal length. In *ICCV'09*, pages 1803–1810, 2009.

[27] M. Bujnak, Z. Kukelova, and T. Pajdla. Robust focal length estimation by voting in multiview scene reconstruction. In *ACCV'09*, pages 13–24, 2009.

[28] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *ACCV'10*, pages 11–24, 2010.

[29] M. Bujnak, Z. Kukelova, and T. Pajdla. Efficient solutions to the absolute pose of cameras with unknown focal length and radial distortion by decomposition to planar and non-planar cases. *IPSJ Transactions on Computer Vision and Applications(CVA)*, 4:78–86, May 2012.

[30] M. Bujnak, Z. Kukelova, and T. Pajdla. Making Minimal Solvers Fast. In *CVPR'12*, 2012.

[31] M. Byröd. *Numerical Methods for Geometric Vision: From Minimal to Large Scale Problems*. PhD thesis, Centre for Mathematical Sciences LTH, Lund University, Sweden, 2010.

[32] M. Byröd, M. Brown, and K. Åström. Minimal solutions for panoramic stitching with radial distortion. In *BMVC'09*, 2009.

[33] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *ICCV'07*, 2007.

[34] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *ECCV'08*, 2008.

[35] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *CVPR'08*, 2008.

[36] M. Caboara, M. Kreuzer, and L. Robbiano. Efficiently computing minimal sets of critical pairs. *JSYMC*, 38:1169–1190, 2004.

[37] J. C.Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *ISSAC'02*, pages 75–83, 2002.

[38] S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the grbner walk. *Journal of Symbolic Computation*, 24:465–469, 1997.

[39] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, 2nd edition, 2005.

[40] D.A. Cox, J. Little, and D.O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, 2010.

[41] A. M. Danilevskii. The numerical solution of the secular equation (russian). *Matem. sbornik*, 44:169–171, 1937. In Russian.

[42] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 3-1-5 — A computer algebra system for polynomial computations. 2012. http://www.singular.uni-kl.de.

[43] M. Dhome, M. Richetin, J.-T. Lapreste, and G. Rives. Determination of the attitude of 3d objects from a single perspective view. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(12):1265 – 1278, Dec 1989.

[44] M. Elkadi and B. Mourrain. Some Applications of Bezoutians in Effective Algebraic Geometry. Technical Report RR-3572, INRIA, December 1998.

[45] I.Z. Emiris. *Sparse elimination and applications in kinematics*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1994.

[46] D.K. Faddeev and V.N. Faddeeva. *Computational methods of linear algebra*. W. H. Freeman, San Francisco, California, USA, 1963.

[47] O. D. Faugeras and S. J. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, 1990.

[48] J. Faugère, P. Gianni, D. Lazard, and F. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, October 1993.

[49] 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.

[50] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[51] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *CVPR'01*, volume 1, page 125, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

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

[53] R. Gebauer and H.M. Möller. On an installation of buchberger's algorithm. *Journal of Symbolic Computation*, 6(2-3):275–286, December 1988.

[54] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Modern Birkhäuser Classics. Birkhäuser, 2008.

[55] V. P. Gerd and Y .A. Blinkov. Involutive polynomial bases. Technical Report IT-271, Laboratoire d'Informatique Fondamentale de Lille, 1995.

[56] C. Geyer and H. Stewenius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. In *CVPR'07*, 2007.

[57] A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso. "one sugar cube, please" or selection strategies in the buchberger algorithm. In *ISSAC'91*, pages 5–4, 1991.

[58] M. Grau and M. Noguera. A variant of Cauchy's method with accelerated fifth-order convergence. *Applied Mathematics Letters*, 17:509–517, 2004.

[59] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry, available at `http://www.math.uiuc.edu/Macaulay2/`.

[60] R. Hartley and H. Li. An Efficient Hidden Variable Approach to Minimal-Case Camera Motion Estimation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 34(12):2303–2314, Dec 2012.

[61] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[62] J. H. He. Newton-like iteration method for solving algebraic equations. *Communications in Nonlinear Science and Numerical Simulation*, 3(2), 1998.

[63] J.H. He. Improvement of newton iteration method. *International Journal of Nonlinear Sciences and Numerical Simulation*, 1(2):239–240, 2000.

[64] J.H. He. A new iteration method for solving algebraic equations. *Applied Mathematics and Computation*, 135(1):81–84, February 2003.

[65] H. H. H. Homeier. On newton-type methods with cubic convergence. *Journal of Computational and Applied Mathematics*, 176(2):425–432, April 2005.

[66] D. G. Hook and P. R. McAree. Graphics gems. In Andrew S. Glassner, editor, *Graphics gems*, chapter Using Sturm sequences to bracket real roots of polynomial equations, pages 416–422. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[67] A.S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover Books on Mathematics Series. Dover Publications, 2006.

[68] A. Irschara, C. Zach, J. M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR'09*, pages 2599–2606, 2009.

[69] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR'11*, pages 3121–3128, 2011.

[70] K. Josephson and M. Byröd. Pose estimation with radial distortion and unknown focal length. In *CVPR'09*, 2009.

[71] K. Josephson, M. Byröd, F. Kahl, and K. Åström. Image-based localization using hybrid feature correspondences. In *BenCOS'07*, 2007.

[72] A. Joux and V. Vitse. A variant of the f4 algorithm. In *CT-RSA'11*, pages 356–375, 2011.

[73] F. Kahl and B. Triggs. Critical motions in euclidean structure from motion. In *CVPR'99*, pages 366–372, 1999.

[74] M. Kalantari, A. Hashemi, F. Jung, and J. Pi. Guedon. A new solution to the relative orientation problem using only 3 points and the vertical direction. *Journal of Mathematical Imaging and Vision*, 39(3):259–268, March 2011.

[75] D. Kapur and Y.N. Lakshman. Elimination methods: An introduction. *Symbolic and Numerical Computation for Artificial Intelligence, B. Donald et al. (eds.)*, pages 45–87, 1992.

[76] D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using dixon resultants. In *ISSAC'94*, I, pages 99–107, 1994.

[77] J. Knopp, J. Sivic, and T. Pajdla. Location recognition using large vocabularies and fast spatial matching. Technical Report CTU-CMP-2009-01, Czech Technical University in Prague, 2009.

[78] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 1*. Computational Commutative Algebra. Springer, 2000.

[79] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 2*. Number sv. 1 in Computational Commutative Algebra. Springer, 2005.

[80] E. Kruppa. Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. *Sitzungsberichte der Mathematisch Naturwissenschaftlichen Kaiserlichen Akademie der Wissenschaften*, 122:1939–1948, 1913.

[81] Y. Kuang and K. Åström. Numerically stable optimization of polynomial solvers for minimal problems. In Andrew Fitzgibbon, editor, *Lecture Notes in Computer Science*, volume 7574, pages 100–113. Springerm, Heidelberg, 2012.

[82] K. Kühnle and E.W. Mayr. Exponential space computation of gröbner bases. In *ISSAC'96*, pages 63–71, 1996.

[83] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *ECCV'08, Part III*, volume 5304 of *Lecture Notes in Computer Science*, pages 302–315, 2008.

[84] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In *BMVC'08*, 2008.

[85] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to the minimal absolute pose problems with known vertical direction. In *ACCV'10*, 2, pages 216–229, 2010.

[86] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, 2012.

[87] Z. Kukelova, M. Byröd, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. *Computer Vision and Image Understanding*, 114(2):234–244, February 2010.

[88] Z. Kukelova, M. Heller, and T. Pajdla. Hand-eye calibration without hand orientation measurement using minimal solution. In *ACCV'12*, 2012.

[89] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *CVPR'07*, 2007.

[90] Z. Kukelova and T. Pajdla. Solving polynomial equations for minimal problems in computer vision. In *CVWW'07*, 2007.

[91] Z. Kukelova and T. Pajdla. Two minimal problems for cameras with radial distortion. In *OMNIVIS'07*, 2007.

[92] Z. Kukelova and T. Pajdla. A minimal solution to radial distortion autocalibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2410–2422, December 2011.

[93] D. Lazard. Gröbner-bases, gaussian elimination and resolution of systems of algebraic equations. In *EUROCAL'83*, pages 146–156, 1983.

[94] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR'07*, 2007.

[95] B. Leibe, K. Schindler, and L. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *ICCV'07*, 2007.

[96] H. Li. A simple solution to the six-point two-view focal-length problem. In *ECCV'06 - Part IV*, pages 200–213, 2006.

[97] H. Li and R. Hartley. A non-iterative method for correcting lens distortion from nine-point correspondences. In *OMNIVIS05*, 2005.

[98] H. Li and R. Hartley. Five-point motion estimation made easy. In *ICPR'06*, volume 1, pages 630–633, 2006.

[99] T. Y. Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica*, 6:399, January 1997.

[100] F. Macaulay. Some formulae in elimination. In *London Mathematical Society*, volume 3, pages 3–27, 1902.

[101] D. Manocha. Solving systems of polynomial equations. *IEEE Computer Graphics and Applications*, 14:46–55, 1994.

[102] D. Manocha and J. F. Canny. Multipolynomial resultant algorithms. *Journal of Symbolic Computation*, 15:99–122, 1993.

[103] 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, September 2004.

[104] E. W. Mayr and A. R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46:305–329, 1982.

[105] C. E. McKay. An analysis of improvements to buchberger's algorithm for gröbner basis computation. Master's thesis, University of Maryland, College Park, 2004.

[106] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 2001.

[107] Microsoft. Photosynth - `http://www.photosynth.net`.

[108] H. M. Möller, T. Mora, and C. Traverso. Gröbner bases computation using syzygies, 1992.

[109] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70(3):311–329, May 1995.

[110] H. M. Möller and R. Tenberg. Multivariate polynomial system solving using intersections of eigenspaces. *Journal of Symbolic Computation*, 32(5):513–531, November 2001.

[111] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP'09*, volume 1, pages 331–340, 2009.

[112] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, June 2004.

[113] D. Nistér, R. Hartley, and H. Stewenius. Using galois theory to prove structure from motion algorithms are optimal. In *CVPR'07*, pages 1–8, 2007.

[114] D. Nistér and H. Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, January 2007.

[115] M. Oskarsson, A. Zisserman, and K. Åström. Minimal projective reconstruction for combinations of points and lines in three views. *Image and Vision Computing*, 22(10):777–785, 2004.

[116] S. Petitjean. Algebraic geometry and computer vision: Polynomial systems, real andcomplex roots. *Journal of Mathematical Imaging and Vision*, 10(3):191–220, May 1999.

[117] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599, 1996.

[118] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3rd edition, 2007.

[119] S. Ramalingam, S. Bouaziz, and P. F. Sturm. Pose estimation using both points and lines for geo-localization. In *ICRA'11*, pages 4716–4723, 2011.

[120] J. F. Ritt. *Differential Algebra*. Colloquium Publications. American Mathematical Society, 2008.

[121] G. Salmon. *Lessons introductory to the modern higher algebra*. Hodges, Figgis, and Co., 1885.

[122] A. Segers. Algebraic attacks from a gröbner basis perspective. Master's thesis, Technische Universiteit Eindhoven, 2004.

[123] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH'06*, pages 835–846, 2006.

[124] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[125] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR'08*, 2008.

[126] Waterloo Maple Software. *Maple User Manual*. Maplesoft, 2005.

[127] T. Stegers. Faugére's f5 algorithm revisited. Master's thesis, Department of Mathematics, Technische Universitât Darmstadt,, 2005.

[128] H. J. Stetter. Multivariate polynomial equations as matrix eigenproblems. *In: Contributions to Numerical Mathematics, World Scientific Series in Applicable Analysis*, 2:355–371, 1993.

[129] H. J. Stetter. Matrix eigenproblems are at the heart of polynomial system solving. *ACM SIGSAM Bulletin*, 30(4):22–25, December 1996.

[130] H. J. Stetter. *Numerical Polynomial Algebra*. SIAM: Society for Industrial and Applied Mathematics, 2004.

[131] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Centre for Mathematical Sciences LTH, Lund University, Sweden, 2005.

[132] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[133] H. Stewénius, C. Engels, and D. Nister. An efficient minimal solution for infinitesimal camera motion. In *CVPR'07*, pages 1–8, 2007.

[134] H. Stewénius, D. Nistér, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. *Image and Vision Computing*, 26(7):871–877, July 2008.

[135] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *OMNIVIS'05*, 2005.

[136] H. Stéwenius, F. Schaffalitzky, and D. Nistér. How hard is 3-view triangulation really? In *ICCV'05*, volume 1, pages 686–693, 2005.

[137] P. Stiller. Sparse resultants. Technical Report ISC-96-01-MATH, Texas A M University, Institute for Scientific Computation, 1996.

[138] A. Torii, Z. Kukelova, M. Bujnak, and T. Pajdla. The six point algorithm revisited. In Reinhard Koch and Fay Huang, editors, *Computer Vision ACCV 2010 Workshops*, volume 6469 of *Lecture Notes in Computer Science*, pages 184–193. Springer Berlin / Heidelberg, 2011.

[139] J. F. Traub. *Iterative Methods for Solution of Equations.* Prentice-Hall, Englewood Cliffs, NJ., 1964.

[140] C. Traverso. Gröbner trace algorithms. In *ISSAC'88*, pages 125–138, 1989.

[141] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *ICCV'99*, volume 1, page 278, 1999.

[142] M. Urbanek, R. P. Horaud, and P. Sturm. Combining off- and on-line calibration of a digital camera. In *Third International Conference on 3D Digital Imaging and Modeling*, pages 99–106, 2001.

[143] J. Verschelde. *Homotopy continuation methods for solving polynomial systems*. PhD thesis, University of California, Berkeley, 1996.

[144] S. Weerakoon and T. G. I. Femando. A variant of newton's method with accelerated third-order convergence. *Applied Mathematics Letters*, 13:87–93, 2000.

[145] W. T. Wu. On the decision problem and the mechanization of theorem-proving in elementary geometry. *Scientia Sinica*, 21:159–172, 1978.

[146] W. T. Wu. Basic principles of mechanical theorem proving in elementary geometrics. *Journal of Automated Reasoning*, 2(3):221–252, August 1986.

[147] W. T. Wu. On zeros of algebraic equations - an application of ritt's principle. *Kexue Tongbau*, 31(1):1–5, 1986.

[148] X. Wu and D. Fu. New high-order convergence iteration methods without employing derivatives for solving nonlinear equations. *Computers & Mathematics with Applications*, 41(3/4):489–495, 2001.