INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

# Pose Estimation Revisited

Bodo Rosenhahn

Bericht Nr. 0308
September 2003


CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

# Pose Estimation Revisited

Bodo Rosenhahn

Bericht Nr. 0308
September 2003

e-mail: bro@ks.informatik.uni-kiel.de

“Dieser Bericht gibt den Inhalt der Dissertation wieder, die der Verfasser
im April 2003 bei der Technischen Fakultät der
Christian-Albrechts-Universität zu Kiel eingereicht hat.
Datum der Disputation: 19. August 2003.”

| | |
|---|---|
| 1. Gutachter | Prof. G. Sommer (Kiel) |
| 2. Gutachter | Prof. D. Betten (Kiel) |
| 3. Gutachter | Dr. L. Dorst (Amsterdam) |

Datum der mündlichen Prüfung: 19.08.2003

# ABSTRACT

The presented thesis deals with the 2D-3D pose estimation problem. Pose estimation means to estimate the relative position and orientation of a 3D object with respect to a reference camera system. The main focus concentrates on the geometric modeling and application of the pose problem. To deal with the different geometric spaces (Euclidean, affine and projective ones), a homogeneous model for conformal geometry is applied in the geometric algebra framework. It allows for a compact and linear modeling of the pose scenario. In the chosen embedding of the pose problem, a rigid body motion is represented as an orthogonal transformation whose parameters can be estimated efficiently in the corresponding Lie algebra. In addition, the chosen algebraic embedding allows the modeling of extended features derived from sphere concepts in contrast to point concepts used in classical vector calculus. For pose estimation, 3D object models are further treated two-fold, feature based and free-form based: While the feature based pose scenarios provide constraint equations to link different image and object entities, the free-form approach for pose estimation is achieved by applying extracted image silhouettes from objects on 3D free-form contours modeled by 3D Fourier descriptors. In conformal geometric algebra an extended scenario is derived, which deals beside point features with higher-order features such as lines, planes, circles, spheres, kinematic chains or cycloidal curves. This scenario is extended to general free-form contours by interpreting contours generated with 3D Fourier descriptors as $n$-times nested cycloidal curves. The introduced method for shape modeling links signal theory, geometry and kinematics and is applied advantageously for 2D-3D silhouette based free-form pose estimation. The experiments show the real-time capability and noise stability of the algorithms. Experiments of a running navigation system with visual self-localization are also presented.

# ACKNOWLEDGMENT

This thesis was developed with the help of several researchers. Without them the thesis would not be in this form. The interaction with other researchers during conferences and exchange projects gave me the motivation to continue with my work and to look forward and extend my research. I can not mention everyone who assisted me during my work, but I would like to acknowledge the most important persons.

First of all, I appreciate my supervisor Prof. Dr. Gerald Sommer for the interesting topic of research, for giving me hints during problem solving and for supporting my conference visits and exchange stays. I will never forget the long discussions, sometimes extending to 3 or 4 hours.

Furthermore, I acknowledge the support and help of all members of the Cognitive Systems Group, University Kiel. My special thanks go to Christian Perwass with whom I had endless discussions and who gave me private lessons in geometric algebra, signal theory and stochastics. He influenced my results fundamentally, since he firstly formalized the Fourier descriptors in conformal geometric algebra and explained the basics of stereographic projections to me. I further acknowledge my colleagues Vladimir Banarer, Sven Buchholz, Oliver Granert, Martin Krause, Nils Siebel and Yohannes Kassahun. I am also thankful to our technical staff, Henrik Schmidt and Gerd Diesner and our secretary Francoise Maillard. They have been helpful all the time and helped to avoid some catastrophes during talks or presentations.

I also thank my students for their work. Especially Oliver Granert, Daniel Grest and Andreas Bunten, with whom I had long discussions about various topics. They helped me to learn more about mathematics, programming and robotics. I liked their autonomous style of working, with lots of creative ideas and visions.

I am also grateful for all the discussions I had with other scientists, especially Dieter Betten, Thomas Bülow, Leo Dorst, Ulrich Eckhardt, Michael Felsberg, Wolfang Förstner, David Hestenes, Reinhard Klette, Reinhard Koch,

Norbert Krüger, Volker Krüger, Joan Lasenby, Hongbo Li, Josef Pauli, Jon Selig and Frank Sommen.

I further acknowledge Silke Kleemann, Anke Kleemann, Christian Perwass, Hongbo Li and Jon Selig for proofreading parts of the thesis.

I also acknowledge the DFG Graduiertenkolleg No. 357 and the EC Grant IST-2001-3422 (VISATEC) for their financial support during my PhD studies.

Last, but not least, I thank my family, my parents Lothar and Regine Rosenhahn, my brothers Axel and Carsten Rosenhahn and my son Julian Rosenhahn. With special thanks I dedicate this thesis to my wife Anke Kleemann.

# CONTENTS

**1. Introduction** . . . . . 1

1.1 Literature overview of pose estimation algorithms . . . . . 7

1.2 The contribution of the thesis . . . . . 13

**2. Foundations of the 2D-3D pose estimation problem** . . . . . 17

2.1 The scenario of pose estimation . . . . . 17

2.2 The stratification hierarchy as part of the pose estimation problem . . . . . 19

2.3 Principles of solving the pose estimation problem . . . . . 21

**3. Introduction to geometric algebras** . . . . . 25

3.1 The Euclidean geometric algebra . . . . . 29

3.1.1 Representation of points, lines and planes in the Euclidean geometric algebra . . . . . 31

3.1.2 Rotations and translations in the Euclidean space . . 32

3.2 The projective geometric algebra . . . . . 35

3.2.1 Representation of points, lines and planes in the projective geometric algebra . . . . . 35

3.3 The conformal geometric algebra . . . . . 39

3.3.1 Stereographic projections . . . . . 39

3.3.2 Definition of the conformal geometric algebra . . . . 43

3.3.3 Geometric entities in conformal geometric algebra . . 44

3.3.4 Conformal transformations . . . . . 48
3.3.5 Rigid motions in CGA . . . . . 50
3.3.6 Twists and screw transformations . . . . . 51

**4. Stratification of the pose estimation problem** . . . . . 57
4.1 Relating entities in Euclidean, projective and conformal geometry . . . . . 58
4.1.1 Change of representations of geometric entities . . . . 60
4.1.2 Pose constraints in conformal geometric algebra . . . 63

**5. Pose constraints for point, line and plane correspondences** . . 65
5.1 Point-line constraint . . . . . 66
5.2 Point-plane constraint . . . . . 68
5.3 Line-plane constraint . . . . . 69
5.4 Constraint equations for pose estimation . . . . . 72
5.5 Numerical estimation of pose parameters . . . . . 74
5.5.1 Generating an example system of equations . . . . . 76
5.6 Experiments with pose estimation of rigid objects . . . . . 77
5.6.1 Adaptive use of pose estimation constraints . . . . . 81

**6. Pose estimation for extended object concepts** . . . . . 85
6.1 Pose estimation of kinematic chains . . . . . 85
6.1.1 Constraint equations of kinematic chains . . . . . 87
6.2 Circles and spheres . . . . . 88
6.2.1 The problem of tangentiality constraints . . . . . 88
6.2.2 Operational definition of circles and spheres using twists 91
6.2.3 The constraint equations of circles and spheres to lines 92
6.3 Cycloidal curves . . . . . 93
6.3.1 Algebraic curves . . . . . 94

6.3.2 Cycloidal curves in conformal geometric algebra . . . 96
6.3.3 Ray-tracing on cycloidal curves . . . . . 101
6.3.4 Constraint equations for pose estimation of cycloidal curves . . . . . 104
6.4 Experiments for pose estimation of extended object concepts 105
6.4.1 Pose estimation experiments of kinematic chains . . . 105
6.4.2 Pose estimation experiments with circles and spheres 111
6.4.3 Pose estimation experiments with cycloidal curves . . 112

**7. Pose estimation of free-form objects** . . . . . 119
7.1 Pose estimation constraints for free-form contours . . . . . 122
7.1.1 Multiplicative formalization of 3D Fourier descriptors 124
7.1.2 Coupling kinematic chains with Fourier descriptors . 125
7.2 Experiments on pose estimation of free-form contours . . . . 126
7.2.1 The algorithm for pose estimation of free-form contours 126
7.2.2 The performance of the pose estimation algorithm . . 130
7.2.3 Simultaneous pose estimation of multiple contours . . 139
7.2.4 Pose estimation of deformable objects . . . . . 143

**8. Conclusion** . . . . . 149
8.1 Further extensions, open problems and future work . . . . . 151

**A. Basic Mathematics** . . . . . 153
A.1 Mathematic spaces . . . . . 153
A.2 Lie groups and Lie algebras . . . . . 157
A.3 The pinhole camera model . . . . . 161
A.4 Signal theory . . . . . 164
A.4.1 Foundations . . . . . 164
A.4.2 Trigonometric interpolation . . . . . 165

A.4.3 The 1D discrete Fourier transformation . . . . . 165

**B. Implemented modules of the navigation system** . . . . . 169

B.1 The tracking problem . . . . . 169

B.1.1 The tracking assumption . . . . . 170

B.1.2 The matching algorithm . . . . . 171

B.1.3 Image feature extraction . . . . . 175

B.2 The hardware setup . . . . . 178

B.2.1 Behavior based robotics . . . . . 179

B.3 The navigation system . . . . . 180

B.4 Experiments with the navigation system . . . . . 182

**Index** . . . . . 205

## Chapter 1

# INTRODUCTION

Mobile robots are of interest for many purposes: they are, for example, used indoors in storage buildings to shelve and transport goods or to clean buildings. Outdoors they may be used to cut grass, dig holes or detect mines. They can be found in several fields, to support different tasks and work as construction or inspection robots, fire fighting robots, entertainment robots, mining robots or search and rescue robots. They are also used in extraordinary situations to explore dangerous places like volcanos, caverns, the deep sea or the universe. An overview of different projects concerning mobile robots can be found on the web page [162], which is organized by the technical committee for service robots of the IEEE[1] Robotics and Automation Society. People build robots to support the human being, so that the robots can undertake tasks which are dangerous, boring or impossible for humans to perform. Robots have properties and qualities which qualify them for many tasks: They can easily be replaced, they do not need to breathe or sleep and can be used in environments which are dangerous, poisonous or risky for humans. Equipped with lifting gears and other working tools they are often stronger and more precise than human beings.

Besides their physical properties, robots are equipped with computers. Unfortunately they lack intelligence. Human beings are able to deal very efficiently with changing and unknown environments. They are able to think and learn. Already children learn tasks like object recognition, object localization or grasping through years of development and experience. Implementing such basic properties on a computer is indeed very hard and though the computer capabilities (e.g. speed and memory) have been increased immensely over the last decades, hitherto only partial solutions exist for those tasks.

[1] The IEEE (Eye-triple-E) is a non-profit, technical professional association. The full name is the Institute of Electrical and Electronics Engineers, Inc., although the organization is most popularly known and referred to by the letters I-E-E-E.

One basic task all mobile robots need to perform is to navigate with respect to unknown or only partially known environments. In its original sense the term *navigation* means to direct a ship to its destination[2][60]. The practice of navigation entered almost unchanged into the domain of robotics. For example, Levitt and Lawton [109] define navigation as a process answering the following three questions:

1. *Where am I?*
2. *Where are other places with respect to me?*
3. *How do I get to other places from here?*

In the context of robot navigation, the robot's sensory input must be used to answer the above questions. In [86] the term *navigation* is defined as follows:

**Definition 1.1** *A movable object has the task to move from its current position to a goal, on the base of fragmentary information and in consideration of boundary conditions.*

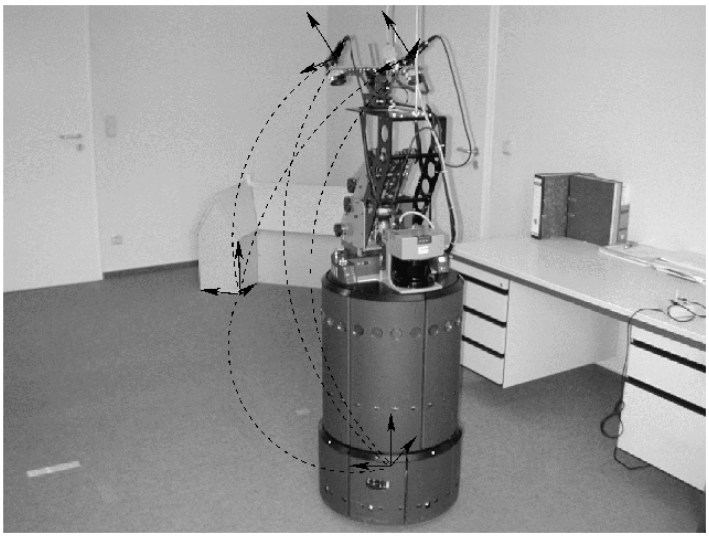


Fig. 1.1: Self localization means to estimate the relative position and orientation of the mobile robot with respect to its environment.

In case of robot navigation the fragmentary pieces of information are the sensory input, and the boundary conditions are a priori knowledge of the

[2] from Latin *navis* ship and *agere* to drive

scenario. It is clear that the sensory system and the scenario influence the strategy of navigation in a fundamental manner.

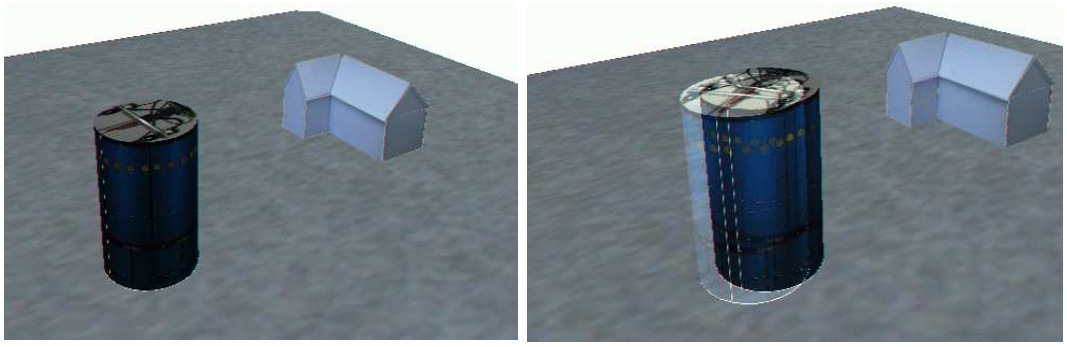


Fig. 1.2: The left image shows the scenario in a virtual environment. The right image visualizes the odometric error of the robot after a few movements.

Many systems use for example ultrasonic sensory data to navigate in environments [70, 31, 120]. Such systems are for example motivated from bats who can use ultrasonic data very efficiently for navigation and hunting in the darkness. In contrast to ultrasonic based navigation, the use of visual information for navigation becomes more and more popular [98, 106]. The reasons for the growing interest in visual navigation are its closeness to human navigation and the generalization capabilities of the human eye as sensor in combination with the brain as visual system: Humans do not only navigate with the help of their eyes, they also perform object recognition, object grasping and object manipulation with their help. Such abilities are important for cognitive tasks but are nearly impossible for ultrasonic sensory data. Therefore my aim is to use visual information for navigation processes. This gives the option for further use of information for more complex behaviors in artificial systems, as well. The task is to navigate a mobile robot with respect to a priori known landmarks of the environment. In my case, these landmarks are CAD-models[3] of furniture like chairs, tables, doors, cupboards or more general structures like passages or hall crossings. Figure 1.1 shows the initial situation of the experimental setup. The knowledge of a landmark (in this case the model house) is known and (for the initial situation) the relative movements of the coordinate systems of the mobile robot's base and the mounted stereo camera systems with respect to the model house are assumed to be known. Indeed it is possible to move the robot and translate

[3] Computer Aided Design models

the robot movements in terms of another (global) coordinate system. However, the movements of the robot are not exact. The accumulated relative movements of the robot lead to increasing inexactness. This means that after several movements the robot's internal coordinate system is no longer correct and the robot is in danger of loosing the orientation. Figure 1.2 shows a visualization of the problem in a virtual environment.

Hence, there is a need to estimate the relative error between visually observed image information and the 3D object model. This leads to the main problem I want to discuss in this thesis, the

**2D-3D pose estimation problem.**

The motivation to choose the 2D-3D pose estimation problem as the main problem of this thesis is based on its central position for robot navigation and other perception-action systems. In [69], page 6, *pose* is defined as follows:

**Definition 1.2** *By pose, we mean the transformation needed to map an object model from its own inherent coordinate system into agreement with the sensory data.*

Thus pose estimation serves to relate several coordinate frames of measurement data and model data by estimating the transformation between them. In a general sense pose estimation can be classified into three categories: 2D-2D, 3D-3D and 2D-3D [85]. In the first and second category, both the measurement data and the model data are 2D or 3D respectively. The third category consists of those scenarios where the measurement data are 2D and model data are 3D. This is the situation for this thesis since 3D objects are observed in 2D images. Roughly speaking, 2D-3D pose estimation means to estimate the relative position and orientation of a 3D object to a reference camera system. This will be explained in detail later.

The aim therefore is to find a rigid motion of an object which leads to the best fit of the reference model with actually extracted image entities. The main focus in this thesis are the geometric aspects of the pose problem. This leads to the following main questions to discuss:

1. How to model the involved geometry and the involved mathematical spaces?
2. What is a rigid motion and what is the best way to estimate it?
3. How to code or describe an object?

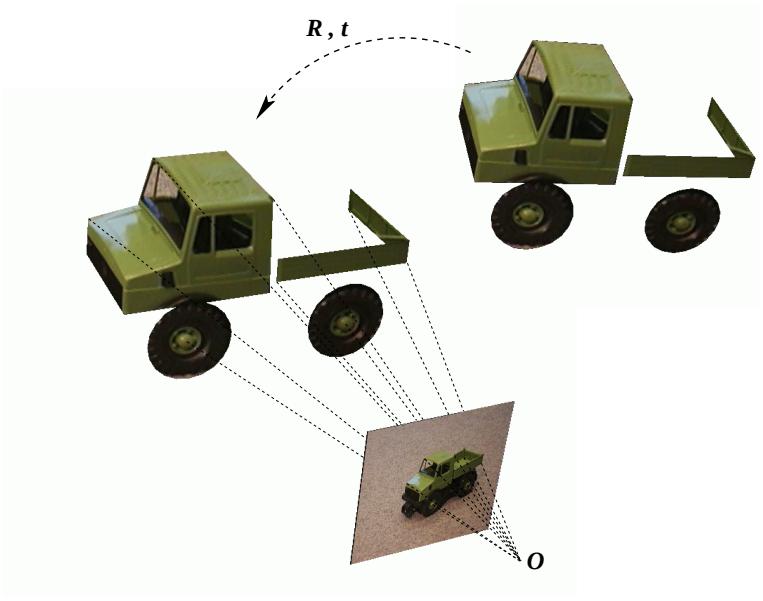



Fig. 1.3: Visualization of the 2D-3D pose estimation scenario. The aim is to find the rigid body motion which leads to a best fit between the object and the image data.

4. How to define a best fit of 3D object data to 2D image information?

5. What kind of image features to extract?

Basic approaches to answer these questions will be discussed in chapter 2 of this thesis. Basically, two principles of modeling the pose problem can be distinguished: firstly, it is possible to project the transformed 3D object in the image plane and to compare the transformed projected object in the 2D projective or affine plane, respectively. Secondly, it is possible to reconstruct the image information to one-dimensional higher entities (for example image points can be reconstructed to projection rays in the 3D space) and the comparison can be done in the 3D space. These principles will be discussed in more detail in chapter 2 and are important for the geometric aspects of the pose problem. The 2D-3D pose estimation problem is visualized in figure 1.3.

Note that throughout this work the 3D object model (independent of its representation) is always assumed to be known and it will not be discussed how to acquire such a model. This topic is also interesting and is for example discussed by N. Krüger [100] or M. Zerroug [188].

The geometric modeling of the 2D-3D pose estimation problem is one of the oldest computer vision problems. Solutions with projective camera models have been proposed for several variations of this problem. I will now introduce the topics for the pose estimation problem, but a comparison with existing work is difficult, since every method has advantages and disadvantages for variations of the pose problem. In this thesis the properties will be discussed in the context of robot navigation and robot vision problems. This means, I have several assumptions which are important in this context, but might be neglected in other situations. The assumptions are:

1. Indirectness: The problem should be formulated in an indirect manner, to deal with deviations and uncertainties. Indirectness means in this thesis that the pose estimation problem is not modeled directly for example by using invariances or closed form solutions, but by constraint equations which relate object features to image features. This results in the estimation of a pose by minimizing an error measure expressed with constraint equations. Using constraint equations is one key point to deal with the pose problem, and the following points 2-6 are properties which are postulated for the constraint equations.

2. Linearity: The problem should be formulated in a linear manner to get constraint equations, which are compact and easy to interpret. This is accompanied by easier optimization strategies for numerical estimation of the pose parameters.

3. Fully projective: The constraint equations should be applied on a fully projective camera model and not on reduced models like orthographic ones. The reason for preferring projective camera models is their exactness in contrast to orthographic models. Since the robot is moving in indoor scenarios and no large distant objects are used for pose estimation, this is of importance.

4. Distance measure: The constraint equations should contain a Euclidean distance measure to ensure well conditioned systems of equations. A geometric interpretation of the constraint equations helps to explain and discuss properties of the developed algorithms, e.g. to analyze degenerate situations or scenarios. Furthermore, a distance measure is the basis to deal with uncertainty and adaptivity within constraint equations.

5. Multiple object features: The algorithm for solving the pose parameters by using the constraint equations should be able to deal with different

kinds of entities in the same manner. Object entities are for example points, lines, circles, spheres, kinematic chains, etc. This is of importance since it leads to more flexibility for cognitive tasks.

6. Free-form objects: The algorithm for solving the pose parameters should also be able to extend the scenario to general free-form contours. This is of importance, since there exist scenarios (e.g. in natural environments) in which it is not possible to extract features like corners or edges, but just general contours.

## 1.1 Literature overview of pose estimation algorithms

In this section I will give a brief summary of important works concerning the 2D-3D pose estimation problem and I will sketch some properties of the methods.

There exist several variations in the methods of pose estimation. An overview of existing techniques for pose estimation can also be found in J.S. Goddards PhD-thesis [65]. To get an order within the whole literature, I will separate the literature into the following different topics:

1. Point based pose estimation: This class deals with the pose estimation for corresponding 2D image and 3D object points. The first pose estimation algorithms were concerned with point based pose estimation and therefore many fundamental and important publications can be found for this class.

2. Pose estimation by using higher order entities: This class deals with the first generalization of point concepts, for example to 2D line to 3D point or 2D line to 3D line correspondences. Furthermore, works concerning conics, kinematic chains or higher order 3D curves are presented.

3. Pose estimation using free-form objects: This class deals with the representation and pose estimation of free-form objects. For example 3D parametric surfaces, 3D meshes and active contours are used in the literature.

4. Uncertainty, adaptivity and multiple object features: This class deals with works presenting more general systems, the modeling of uncertainties and constraint fusion.

5. Estimating pose parameters: This class reviews pose estimation algorithms and their numerical performance in the literature. It is of importance since numerical inefficiencies are more sensitive to image noise, inexact calibration or correspondence outliers.

Note, that the last two topics are no real classes in contrast to the first three ones. Instead, they are connected to the first three topics but are treated here as extra parts to stress their importance within the pose estimation algorithms.

**1. Pose estimation using point-based methods**
Point based pose estimation means to estimate the pose of an object by determining the identification and location of feature points on an object from 2D image points in a scene. A rigid body is generally assumed, but no complete explicit geometric model is given. Methods of this class were firstly studied in the 80's and 90's and pioneering work was done by Lowe [116, 117] or Grimson [69]. Lowe applied a Newton-Raphson minimization method to the pose estimation problem and showed the direct use of numerical optimization techniques in the context of noisy data and in gaining fast (real-time capable) algorithms. His work is based on pure point concepts and he expressed the constraint equations in the 2D image plane. To linearize the equations an affine camera model is assumed and an extension to a fully projective formulation can be found in Araujo et al. in [3]. The minimum number of correspondences that produce a unique solution are three (non-collinear and non-coplanar) points. Four coplanar and non-collinear points also give a unique solution [65]. In general the accuracy increases with the number of used point features. Over-determined solutions are for example also used for camera calibration [47].

A pose estimation algorithm based on *dual quaternions* [21] is given by Walker et al. [179]. The method uses the real-part of the dual quaternion to estimate the rotational part and the dual-part of the dual-quaternion to estimate the translational part of the pose. This effect is also discussed by Daniilidis in [41] in the context of hand-eye calibration.

The use of 3D Plücker lines [21] was investigated by Shevlin [163] and Selig [161]. Their result is that although Plücker lines are well known in robotic kinematics, their coupling with computer vision tasks is often neglected. Plücker lines have interesting properties [141], especially in the context of noise, and are also explained and analyzed in this thesis.

Holt and Netravali discussed in 1991, [83], the pose estimation problem for

$N$ corresponding points whose coordinates are known in some 3D reference frame and their 2D perspective projection onto an image plane. If $N$ is three it is shown, that in general there can be up to 4 solutions unless the points are collinear. If the points are collinear, there exist up to infinite many solutions. The case of 4 coplanar points is examined for various combinations of object and image point collinearity conditions. Four non coplanar points are shown to have at most 4 solutions.

**2. Pose estimation by using higher order entities**
Holt and Netravali discussed in 1996, [84], the uniqueness of the structure and motion problem for various point and line correspondences. Their result is that two points and one line, or two lines and one point across three views give in general one unique solution for the structure and motion of the object. Similarly, if one point and one line correspondence is known over four views there is also a unique solution for motion and structure.

Navab and Faugeras discussed in 1993, [125], the pose determination problems for line correspondences. The lines are represented as Plücker lines and they show that the maximum number of solutions for orientation determination is four, and three is the maximum number of solutions for the pose determination problem.

Ghosh et al. developed in [63] a non-linear theory for perspective vision and discuss points, lines (Plücker lines) and polynomials of second order. Their main argument is that vision problems by their nature are perspective and many geometric structures of a perspective system are lost if they are studied via linearization. They introduce a non-linear theory and call it *perspective system theory*. Unfortunately no experiments are presented in their work. Neither for their non-linear equations, nor a comparison with linearized approaches is done.

In 1991 Homer [85] proposed a solution for pose determination from line-plane correspondences and described the pose problem in the 3D framework, overcoming non-linearities which occur in the 2D Euclidean plane. Hourad [87] extended this approach in 1995 to point and line correspondences and showed that it is possible to decouple the recovery of the rotational pose parameters from their translational counterparts completely.

In Winkler et al. [183] a neural network based approach to obtain rough 3D pose results from 2D camera images is presented. Training is done on synthetic views which are generated from a 3D CAD-like object model containing line segments. Once trained, the network can also classify real images. They also recommend a unit quaternion representation for pose estimation

and claim a real-time capable algorithm, with a processing time of 90 msec. The image processing is implemented on a Datacube MV200 and the network response is computed under an SGI Indigo workstation.

In Klingspohr et al. [97] a gaze estimation system is presented by modeling image conics. The main focus of this approach is real-time property, and the gaze direction is estimated as two unknown angles. Only conics are modeled and no coupling with other entities is discussed.

Coupling the pose problem of rigid objects with kinematic chains is done, for example, by Bregler et al. [25] who coupled twist representations of rigid body motions with the pose problem. A kinematic chain is basically an object with including joints. In Breglers work, an orthographic camera model is assumed to simplify the equations. A hierarchical pose estimation algorithm is presented 1997 by Hauck et al. in [74]. The recognition task is solved by using a tree-like structure for the components of the object. To estimate the pose and joint configurations firstly the static component (root) of the object and afterwards the relative movements of the remaining components are determined recursively. A comparable approach to this work for pose estimation of kinematic chains can be found in Weik et al. [181]. There, also a hierarchical pose estimation algorithm is presented and applied to a silhouette-based, volumetric, reconstructed object. In this thesis, kinematic chains are modeled in section 6.1 and experiments concerning pose estimation of kinematic chains can be found in section 6.4.1.

Hel-Or et al. present in [76] a method for localization and interpretation of modeled objects that are general enough to cover articulated and other types of constrained models. Joints of articulated objects are expressed as spatial constraints that are fused in the pose estimation during an *interpretation process*. The framework is based on Kalman filtering and example experiments are done with a lamp-shade, comparable to the experiment in figure 6.30 of this thesis.

**3. Pose estimation using free-form objects**
An overview of free-form object representations is given in [33]. There, several mathematical forms are discussed, e.g. parametric forms, algebraic implicit surfaces, superquadrics, generalized cylinders or polygonal meshes.

In Kriegmann et al. [99] a work is presented, which extends point features to *viewpoint dependent* point features, which include vertices, t-junctions, cusps, three-tangent junctions, edge inflections, etc. The objects are modeled by algebraic surfaces as implicit polynomials. This means, the surface is given by the zero set of a polynomial $f(\boldsymbol{x}) = f(x, y, z) = 0$. In the experimental

part, a cylinder with a cylindrical notch is used. Their implementation for solving the system of equations requires 20 hours on a SPARC Station 1.

Zerroug et al. present in [188] the pose estimation problem of multi-part curved objects. They discuss the computation for a scaled orthographic camera model and propose the refinement with another perspective version of their algorithm. The object models are represented by a graph representation containing nodes and arcs. For estimating the pose parameters, the rotation is described by the Euler angles and a closed form solution is proposed. Their algorithm for pose estimation uses point correspondences gained from the graph model and image features.

Drummond et al. [46] present an approach for real-time tracking of articulated objects. Similar to [25] they do not search for the Lie group action in $SE(3)$, but they linearize the equations and end up in searching for the generating Lie algebra element in $se(3)$. They present a system which is running on a SGI O2 (225 MHz R10K) at PAL frame rate (25Hz). Furthermore, extensions to multiple cameras and multiple targets are discussed.

K. Stark presents in [171] a method for tracking the pose of 3D objects based on an active contour model. The system consists of two components, a 2D tracker for the image processing and a 3D pose tracker for estimating the pose with the use of the known 3D object model. It is a silhouette based approach and B-splines are used to track an object silhouette in an image sequence. In this context also aspect changes are modeled, and experiments on a SUN SPARC 20 station lead to a frame rate of 26 fps[4].

Works concerning affine pose estimation of free-form contours by using Fourier descriptors are made by K. Arbter or T.H Reiss [7, 138]. A full projective approach does not exist so far and the main problems occurring there are discussed in the appendix of [146].

Farouki et al. present in [53] bipolar or multipolar coordinates to model and analyze free-form shapes. In contrast to the use of Cartesian coordinates, entities like the ellipse, a hyperbola, the circle of Apollonius, the limacon of Pascal or the Cartesian oval are represented in a linear manner in this model. They discuss different applications, for example in the field of geometrical optics to characterize the geometry of spherical waves and their reflections or refractions.

**4. Uncertainty, adaptivity and multiple object features**
Ayache and Faugeras provide in [10] a method for representing uncertainty

[4] frames per second.

in measurements of points, lines and planes that are used in building visual maps of the environment for a mobile robot. Two images are taken at each robot location as a stereo pair. An extended Kalman filter (EKF) is used as a model for estimating the position of these primitives and the uncertainty through a covariance matrix. The noise of the feature location from the image is modeled as zero mean Gaussian noise with a known variance. As the robot moves, additional lines are taken and are used to update the filter and to improve the estimation of the 3D locations.

Modeling uncertainties during matching and pose estimation is also done by J.R. Beveridge in [18]. There, local search strategies are proposed to reduce the search space and to identify outliers. The combination of different entities with different reliabilities of measurements is mostly ignored in the literature. For example Hel-Or and Holt consider this problem in [75, 84].

The fusion of different local entities is also discussed by N. Krüger [103]. In this paper different image information like local phase, color, orientation are attached to image points, and relations among these features are automatically estimated in an interpretation process.

**5. Estimating pose parameters**
In the literature several algorithms for numerical pose estimation can be found. These are SVD-approaches[5], extended Kalman filter or gradient descent methods. A comparison of four approaches is made by Lorusso et al. in [115]. The algorithms deal with 3D point based pose estimation and are based on a SVD decomposition, unit quaternion (UQ), dual quaternion and eigensystem (OM) computation. The comparison consists of three parts, accuracy, stability and relative efficiency. Their results are not in agreement with results presented in [179] and they found out, that the SVD and UQ methods are very similar and usually the most stable. The OM method is not as stable for planar data sets, but superior for large degenerate data sets. The DQ algorithm was never the most stable and usually broke down before the others. Unfortunately, they do not compare a gradient descent method within this context. A gradient descent method will be proposed in this thesis. In contrast, one of my numerical results will be (see figure 5.6), that the gradient descent method behaves more stable (in the context of Gaussian noise) then the SVD approach proposed in [115].

A. Ude presents in [177] a nonlinear least squares optimization algorithm by using unit quaternion functions based on unconstrained optimization techniques. It results in a Gauss-Newton iteration which is rescaled to the unit

[5] singular value decomposition

sphere after each iteration. They demonstrate their approach for pose estimation from 2D to 3D line segment correspondences.

Unfortunately the fusion of kinematics and projective geometry is nearly always neglected and Faugeras' stratification hierarchy [48] is either implicitly assumed, or ignored in the context of the pose estimation problem. Only few works deal with these problems. For example A. Ruf treats this problem in the context of point correspondences in [157]. Farouki discusses in [52] the historical connection between geometry and kinematics. His main point is, that the further development of such connections is crucial for computer aided design and manufacturing applications. In this context the relations of implicit and parametric forms of curves and their motions are discussed. Different topics, for example the *four-bar linkage* are analyzed in a historical context and their attractiveness for computer science problems is pointed out. Especially the notes about Newton, Descartes or Galileo are interesting and help to arrange the history about geometry and their connection to kinematics. Since this work is rather a historical survey, no experiments on pose estimation are presented.

## 1.2 The contribution of the thesis

This thesis deals with the 2D-3D pose estimation problem. The main focus concentrates on the geometric modeling of the pose problem. At first, the mathematical spaces involved in the 2D-3D pose estimation problem are considered and the main principles dealing with this problem are discussed in chapter 2. Chapter 2 starts with the Faugeras' stratification hierarchy [48], which contains the Euclidean, affine and projective space. It turns out, that all levels of the hierarchy are involved in the pose estimation problem. To deal with kinematics on the one hand and projective geometry on the other hand, a conformal embedding is proposed and derived in chapter 3. This means that points are modeled as stereographically projected points [126]. Underlined with a homogeneous model for stereographic geometry, this leads to a model which provides important properties for the pose scenario: the geometric coupling of Euclidean, kinematic and projective geometry. This model is then used in the geometric algebra framework [77]. The conformal geometric algebra [110] allows to deal with higher order entities (lines, planes, circles, spheres) in the same manner as with points. It is further possible to model the conformal group on these entities by applying special operators in a multiplicative manner. In this work not the whole conformal group is used for the pose estimation problem, but just the elements (*rotors*, *translators*),

which lead to *motors*, expressing rigid body motions. This means that I work with a geometric model which copes with projective and kinematic geometry and expresses rigid motions as special products. The geometric algebra is introduced in chapter 3. In this chapter the reader can find links to other references introducing the basics of geometric algebras. The coupling of projective and conformal geometry is explained in chapter 4. Linear operators are defined which switch an entity given in the projective space, to one in the conformal space and vice versa. From this starting point, in chapter 5 a scenario is derived which fulfills all assumptions which I have on the pose problem: it results in an **indirect**, **linear** and **fully projective** description of the pose problem, which contains a **Euclidean distance measure** and is suited for **simultaneous use of multiple object features**. Though it is indeed not hard to solve the pose problem by using pure point concepts, the fusion of projective and kinematic geometry and its combination with higher order entities requires an extended point of view to the pose scenario. This is achieved by using a homogeneous model for stereographic projections. Therefore, the mathematical language in this thesis will be *Clifford* or *geometric algebras*. From the numerical point of view, the estimation of pose parameters is not trivial, since they are expressed as an exponential function and therefore as a polynomial of infinite degree. The exponential representation of motors is used and applied on the Taylor series expansion of first order for approximation. This means that in this work (chapter 5.5) no search for the parameters of the Lie group $SE(3)$ is done to describe the rigid body motion [62], but for the parameters which generate their Lie algebra $se(3)$ [124]. The elements of $se(3)$ are also called *twists*. This leads to linear equations in the generators of the unknown 3D rigid body motion. Iterating the linearized equations leads to fast and accurate approximations of the pose. This is comparable to a gradient descent approach performed in the 3D space.

These fundamentals build the basis of the thesis and extensions of the pose problems are build upon them: From simple point and line features scenarios are discussed which extend the entities to kinematic chains (chapter 6.1), circles and spheres (chapter 6.2) and cycloidal curves (chapter 6.3): Cycloidal curves are a special case of algebraic curves. In general, a cycloidal curve is generated by a circle rolling on a circle or line without slipping [108]. This leads to epitrochoids, hypotrochoids and trochoids as special classes of entities which will be used in the context of 2D-3D pose estimation.

In chapter 7 a generalization of cycloidal curves (they are later called *2twist generated curves*) to $n$-times nested cycloidal curves is presented: There exists a direct connection of these curves to Fourier-descriptors. The

generalization of the cycloidal curves will result in general 3D-*free-form* contours. To cope with the algebraic coupling of free-form contours within the pose estimation problem, twists are used as a link between these different topics. Twists are well known from Lie groups and Lie algebras [62, 124] and are applied on rigid body motions. In this thesis, twists will be used on the one hand within the pose estimation problem and on the other hand to model object contours. This unification facilitates a compact description of the pose problem for free-form contours. To estimate the pose of free-form contours ICP (Iterative Closest Point) algorithms are applied, which are well known for aligning 3D object models. Originally ICP starts with two data sets and an initial guess for their rigid body motion. Then the transformation is refined by repeatedly generating pairs of corresponding points of the sets and minimizing the error metric. The ICP algorithms are mostly applied on 2D or 3D point sets. Furthermore, they will later be used to compare a 3D contour, modeled by Fourier descriptors, with 3D reconstructed projection rays. Different works concerning ICP algorithms can be found in [158, 40, 89, 187]. The use of Fourier descriptors is accompanied by some features, which can advantageously be applied within the pose estimation problem: instead of estimating the pose for a whole 3D contour, low-pass descriptions of the contour can be used for an approximation. This leads to a speed up of the algorithm and helps to avoid local minima. It can basically be interpreted as a multi-resolution method. In chapter 7 also extensions to free-form pose estimation are discussed. In these extensions I deal with the simultaneous use of multiple contours, the treatment of hidden or partially hidden contour parts (chapter 7.2.3) or the adaptive modeling of *object deformations* (chapter 7.2.4).

The thesis ends with a discussion in chapter 8. The basic mathematics used throughout this thesis are introduced in the appendix A. There can be found the basic definitions of mathematic spaces, groups, algebras etc.

Since the starting point of this thesis is visual landmark based robot navigation, appendix B deals with the implementation details of a running navigation system. A system is presented which enables a mobile robot to navigate with respect to an a priori known landmark. In this part, the landmark is assumed as a given CAD-model containing just points and lines. The tasks to be solved in this context are image feature extraction by using a modified Hough transformation [88], pose estimation, matching by using a local search strategy [18] and path planning. Another main point of this part is the presentation of a behavior based control system of the robot. For this purpose the implemented modules are separated in parallel running processes, embedded in a subsumption architecture [73]. This module was

part of a student project I supervised. This is explained in more detail in [68].

## Chapter 2

# FOUNDATIONS OF THE 2D-3D POSE ESTIMATION PROBLEM

This chapter introduces the foundations of the 2D-3D pose estimation problem. At first the general scenario is presented. Then the involved mathematical spaces are localized in the scenario. Thirdly the main principles of how to cope with the pose estimation problem are explained and discussed.

## 2.1 The scenario of pose estimation

In the scenario of figure 2.1 the following situation is visualized: Assumed is a 3D object, which can contain different entities like 3D points, 3D lines, 3D spheres, 3D circles, kinematic chain segments, boundary contours or contour parts. Furthermore corresponding features in an image of a calibrated camera are extracted. The aim is to find the rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ of the object which lead to the best *fit* of the reference model with the actually extracted entities. So far, it is not clarified how to define such a *fit*, or a *fit quality*. Though it is clear by intuition, a mathematic formalization is not easy.

In this thesis I will concentrate on two different kinds of pose estimation, which are on the one hand feature based and on the other hand free-form based. Examples of entities used in this thesis are given in figure 2.2 and, as can be seen, there is a crossover between simple features, constraint curves and surfaces to free-form curves and surfaces. It will later be shown that it is possible to define classes of entities which extend the simple feature based pose estimation to a subclass of free-form pose estimation, and twists will be used to achieve this.

I concentrate on both kinds of pose estimation, since there exist many

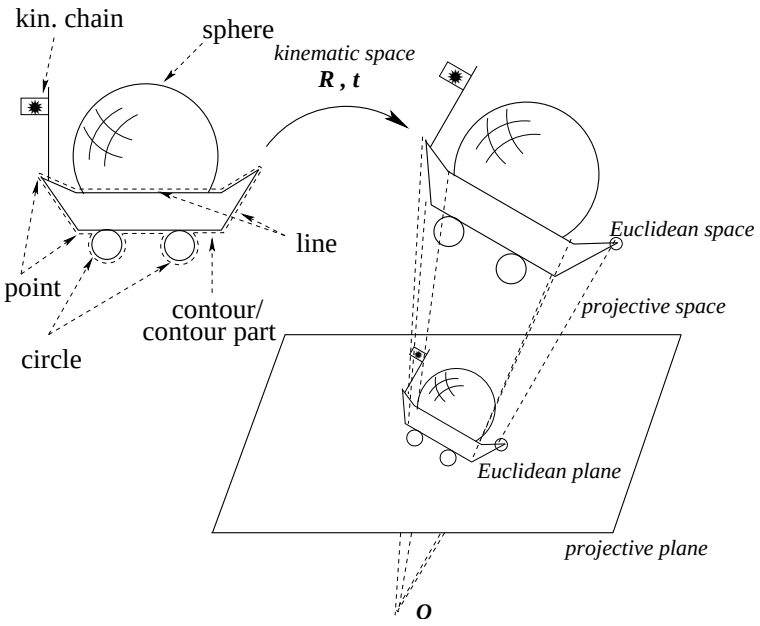



Fig. 2.1: The scenario. The solid lines describe the assumptions: the camera model, the model of the object (consisting of points, lines, circles, spheres and kinematic chains) and corresponding extracted entities on the image plane. The dashed lines describe the pose of the model, which lead to the best fit of the object with the actually extracted entities.

| Feature based pose ⟶ | | | Free−form pose |
|---|---|---|---|
| ***points*** | ***circles*** | ***cycloidal curves*** | ***Fourier descriptors*** |
| ***lines*** | ***spheres*** | ***ruled surfaces*** | ***active contours*** |
| ***planes*** | ***kinematic chains*** | | ***algebraic implicit surfaces*** |
| | | | ***polygonal meshes*** |
| | | | ***superquadrics*** |

Fig. 2.2: Examples for the crossover from feature to free-form based pose estimation.

scenarios in which it is not possible to extract point-like features such as corners or edges, but only general contours. These are the scenarios addressed with the free-form models. Additionally, from a statistical point of view, pose estimations of global object descriptions are more accurate and robust than those from a sparse set of local features. But on the other hand feature based pose estimation can be performed much faster. While the feature based pose estimation can be performed in less then 5 ms, real-time performance of free-

form contours is dependent on the scenario and only possible with optimized parameters. In the experiments, results are obtained in $40 - 250$ ms on a Linux 850 MHz machine for the used test objects. This will be discussed in more detail later on.

The geometric scenarios for pose estimation always assume the knowledge of the correspondences. The correspondence problem deals with the question, which object feature belongs to which image feature. For the navigation scenario or online demos this problem must be solved. Basically either a ROI-search[1] window, local search strategies [18], or (modified) ICP-algorithms [40, 187], depending on the scenario and situation are used. The used and proposed methods are discussed in detail in chapter 7.2.1 and B.1.2.

## 2.2 The stratification hierarchy as part of the pose estimation problem

In the scenario of figure 2.1 four mathematical frameworks can be identified: The first one is a projective plane of a camera, embedded in the second framework, a 3D projective space. In these spaces it is possible to project or reconstruct entities by applying e.g. projection matrices as explained in chapter A.3. The third one is the framework of kinematics, which contains the map of the *direct affine isometries* [62] and can be used to describe rigid body motions. A set of entities with the property that the distances between any two of the entities never varies is called a *rigid body*, and a continuous transformation with the property that it preserves distances during transformation is called a *rigid body motion*. A rigid body motion corresponds to the Euclidean transformation group $SE(3)$[62]. Being a transformation by itself, it contains rotations and translations. The fourth framework is the Euclidean space or Euclidean plane, which is important to define distances between entities. The basic definitions of these spaces are the following [62]:

1. The **Euclidean space** is a vector space $V$ with a symmetric positive definite bilinear form (which induces an Euclidean norm).
2. The **affine space** consists of a set of points, a derived vector space and two operations, namely vector addition and scalar multiplication.
3. The **kinematic space** is an affine space with the group of rigid motions as special affine transformation.

[1] region of interest

4. The **projective space** $\mathcal{P}(V)$ induced by $V$ is the set $(V\backslash\{0\})/\sim$ of equivalence classes of nonzero vectors in $V$ under the equivalence relation $\sim$ defined such that
$$\forall u, v \in V\backslash\{0\} : u \sim v \Leftrightarrow \exists \lambda \in \mathbb{R} : v = \lambda u.$$

Mathematically a projective space $\mathcal{P}(V)$ is a set of equivalence classes of vectors in $V$. The idea behind projective geometry is to view an equivalence class $(u)_\sim$ as an atomic object, forgetting the internal structure of the equivalence class. For this reason it is customary to call an equivalence class $a = (u)_\sim$ a *point* (the entire equivalence class $(u)_\sim$ is collapsed into a single object, viewed as a point). A more detailed introduction of the involved mathematical spaces in terms of classical matrix calculus can be found in chapter A.1 of the appendix.

The idea is to reach the Euclidean plane or Euclidean space in the end. Only in this way it is possible to cope with the problem of noisy data in a geometric context and to evaluate the quality of the estimated pose. But since the Euclidean space is not well suited to describe projective geometry and kinematics, the aim is to transform the generated constraint equations into a distance measure of the Euclidean space in the very last step. Before this step, the other spaces are used to represent the situation in a suitable way. The spaces of the pose estimation scenario mentioned above are exactly the spaces of the stratification hierarchy which Faugeras introduced in 1995 [48]. The three main representations he considered are the projective, affine and metric ones. All strata are involved in the 2D-3D pose estimation problem.

The stratification hierarchy proposed by Faugeras has its roots in the vector space concepts and assumes points as the represented basic geometric entities. The other spaces are derived from the vector space concepts: Well known is the homogeneous extension to express an Euclidean space as affine space and to use the homogeneous component for distinction between points and directions in the affine space. The projective space as a set of equivalence classes is directly build on the homogeneous vector space concepts. This way to stratify the vision space is clearly motivated by the underlying point concepts of the vector spaces and is shown in the first row of table 2.1.

I am using geometric algebras to represent and handle different mathematical spaces of geometric meaning. The maximum sized algebra used so far, is an algebra to handle conformal transformations [80]. A transformation is said to be conformal, if it (locally) preserves angles. This algebra contains algebras for modeling projective and Euclidean geometry as sub-algebras. It

| Concept | Stratification | | | | |
|---|---|---|---|---|---|
| Vector calculus | Euclidean | $\subseteq$ | affine | $\subseteq$ | projective |
| Geometric algebra | Euclidean | $\subseteq$ | projective | $\subseteq$ | conformal |

Tab. 2.1: Stratification of mathematical spaces.

results in other layers of stratification which are proposed in this thesis. In geometric algebras, so-called multivector concepts are used to model geometry beyond point concepts. In the next section it will be explained why it is necessary also to extend geometric algebras to homogeneous models. This leads to a different stratification of the space, since this stratification is not based on pure point concepts any more. Instead, the stratification consists of algebras for modeling the Euclidean, projective and conformal space. This is shown in the second row of table 2.1.

## 2.3 Principles of solving the pose estimation problem

One main problem of 2D-3D pose estimation is how to compare 3D Euclidean object features with 2D projective image data. There are two strategies for comparison: On the one hand, it is possible to project the transformed entity onto the image plane and to compare it with the extracted image data. This leads to a comparison in the projective plane or Euclidean plane respectively. The second possibility is to reconstruct the object from the image data and to compare the (one dimension higher) entities with the 3D object features. Both approaches have advantages and disadvantages. Here I want to discuss a few properties of both strategies: To enable comparisons in the first strategy, the projected object features have to be scaled in their homogeneous component. This leads to fractions with the unknown transformation in both, the numerator and denominator. The equations are not linear any more and are harder to solve numerically. Though the equations can also be expressed as projective linear system of equations, the problem is then to loose a distance measure and to risk badly conditioned equations. To avoid such problems, orthographic projections (see e.g. [25]) are used, but then the camera model is not perspective any more. Since the second strategy uses projectively reconstructed data, this problem does not occur there. But the problem is that the distance measures in the 3D space differ from those in the image plane: Though the distance of two image points may be constant, the distance of two 3D projectively reconstructed points may vary with the distance of the points to the optical center of the camera. This necessitates

| strategy | linear | Euclidean distance measure | full perspective |
|---|---|---|---|
| Euclidean | no | yes | yes |
| Orthographic projective | yes | yes | no |
| Full projective | yes | no | yes |
| 3D kinematic | yes | yes | yes |

Tab. 2.2: Principles for formalizing the pose problem.

that the constraints must be adapted with respect to the projective depth for degenerate situations[2]. Table 2.2 summarizes the main principles of solving the pose problem by using constraint equations.

In this thesis (similar to e.g. [179]) projectively reconstructed 3D data from image data will be used and the (one dimension higher) entities will be compared with the 3D object model features. There are three main arguments why I choose this strategy: firstly I want to describe the constraints in an as simple way as possible and want to gain real-time performance. For this purpose, the reconstructed data are easier to handle in the 3D kinematic space than in the 2D space. This is shown in figure 2.3: comparing

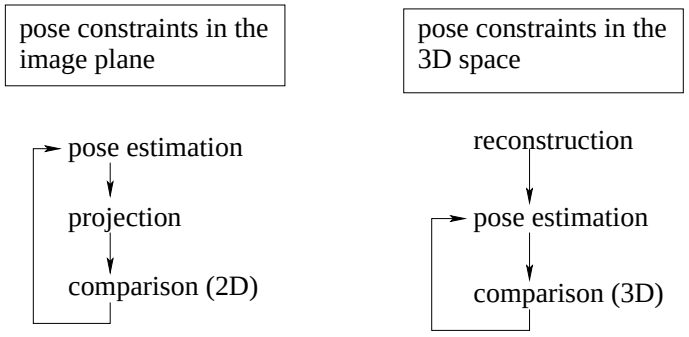



Fig. 2.3: Comparison of iterating constraints defined in the image plane, or constraints defined in the 3D space.

iterative pose results in an image plane requires the estimation of a projection for comparison in the image plane after each iteration, whereas once reconstructed image data can be compared immediately after each iteration. This means, that repeated projection estimation can be avoided. The second

[2] Problems can occur if the object is very large (e.g. a hallway), with some object features very near to the camera plane and other object features far away from the camera plane. In such situations, the spatial distance (which will be minimized) of the near object features influence the equations to a lesser extent then the distant object features.

advantage of using 3D constraints is that the error measures are formalized in the 3D Euclidean space and are directly connected to a spatial distance measure. This helps to qualify the pose result and to detect outliers. This is in contrast to other approaches, where the minimization of errors has to be computed directly on the manifold of the geometric transformation [35, 177]. The third argument is that the depth dependence of the 3D constraints can be adapted to each situation. As will be shown later (see chapter 5.6.1) the constraints can be scaled and transformed in depth-depending constraints comparable to the situation observed in the 2D image plane. But the main argument is that only in this model it is possible to gain linearity, a Euclidean distance measure and a full-perspective camera model without introducing special operations (e.g. using covariance matrices, etc.).

With this analysis I will answer the five main questions of pose estimation given in the introduction as follows:

1. *How to model the involved geometry and the involved mathematical spaces?*
   *Answer:* The involved mathematical spaces are the Euclidean, affine and projective ones. The main problem is, how to fuse kinematics with projective geometry. Since both spaces are subsumed by a homogeneous model for stereographic projections, a conformal model will be proposed in this thesis.

2. *What is a rigid body motion and what is the best way to estimate it?*
   *Answer:* (a) A rigid motion is a map of direct affine isometries. It is a continuous transformation that preserves distances during the transformation. It corresponds to the special Euclidean transformation group $SE(3)$ and contains rotations and translations.
   (b) There exists no unique answer to this question, since the answer is dependent on the scenario and situation. In this work, I will use and compare three different algorithms for estimating pose parameters. The first is a simple SVD-approach [136], the second a Kalman filter (developed by Y. Zhang [190]) and the third is a gradient descent method. Since I am dealing with different entities at the same time and want to weight the equations, it turns out that the gradient descent method is the best approach within my scenario, comparing adaptivity, computing time and robustness.

3. *How to code or describe an object ?*
   *Answer:* I choose two ways to handle objects: Objects can be described as a set of features (e.g. points, lines, kinematic chains, etc.) or an object can be modeled as one whole function. I will end up with free-form contours as the most general object description.

4. *How to define a best fit of 3D objects to 2D image information?*
   *Answer:* The fit quality is expressed as a distance measure in the 3D space. Therefore I work with reconstructed image information (e.g. points are reconstructed to projection rays, and image lines are reconstructed to 3D planes). The image information is transformed from the 2D projective plane via the 3D projective space to the 3D conformal space, where it is rescaled to the Euclidean space, without loosing linearity in the unknowns.

5. *What kinds of image features to extract?*
   *Answer:* There also exists no unique answer to this question, since the usefulness of image features is very dependent on the scenario itself. It is easy to imagine a scenario in which image points are easy to extract and can easily be applied to 3D features of the object model. But in a different scenario, point features might be useless and line or curve segments of the object are more stable to extract. This is the reason why in this thesis the simultaneous use of different entities for pose estimation is investigated. Only in this way it is possible to gain systems which can deal with different scenarios and situations.

   In this thesis, the image processing is kept simple: I use either color markers for the detection of point correspondences, the Hough transformation [88] for the detection of image lines, or contour algorithms for image feature extraction. To establish correspondences I use either a ROI-search window, a local search strategy [18], or (modified) ICP-algorithms [40, 187], depending on the scenario and situation.

## Chapter 3

# INTRODUCTION TO GEOMETRIC ALGEBRAS

What is currently called *geometric algebra* [80] can be seen as a Clifford[1] algebra with its main focus on a suited geometric interpretation. Clifford's contribution of inventing a geometric extension of the real number system to a complete algebraic representation of directed numbers is historically reviewed by M. Yaglom in [184]. In [184] there is also enlightened the relation to former works of Grassmann (1809-1877) or Hamilton (1805-1865). The term geometric algebra was introduced by David Hestenes in the 1960's, who made further development of Clifford algebra in classical geometry and mechanics.

Clifford (or geometric) algebras have the properties of compact symbolic representations of *higher order entities* and of linear operations acting on them. A *higher order entity* can be seen as a subspace of a vector space. This means, e.g. lines and planes are so-called higher order entities which are represented as unique elements in a Clifford algebra. Furthermore, many geometric concepts which are often introduced separately in special algebras are unified in geometric algebras. So the concepts of duality in projective geometry, Lie algebras and Lie groups, incidence algebra, Plücker representations of lines, complex numbers, quaternions and dual quaternions can all be found in suitable geometric algebras with associated splits.

This chapter starts with a general introduction to geometric algebras and proceeds with algebras to model the Euclidean, projective and conformal space. Parts of this chapter (e.g. notations, definitions) are based on the following works [77, 78, 110, 44, 45, 80, 23, 81, 107, 131, 164].

In general a geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ $(p, q, r, \in \mathbb{N}_0)$ is a linear space of

[1] William K. Clifford lived 1845-1879.

dimension $2^n$, $n = p + q + r$, with a subspace structure, called blades, to represent so-called multivectors as higher grade algebraic entities in comparison to vectors of a vector space as first grade entities. A geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ is constructed from a vector space $\mathbb{R}^{p,q,r}$, endowed with the signature $(p, q, r)$, by application of a *geometric product*. The notation $\boldsymbol{\mathcal{G}}_{p,q,r}(\mathbb{R}^{p,q,r})$ is sometimes used to stress the vector space and its signature from which the geometric algebra is built. Note that $\mathbb{R}^{p,q,r} \subseteq \boldsymbol{\mathcal{G}}_{p,q,r}$. This means that the generating vector space is always an element of its generated geometric algebra and therefore the vectors of the vector space can be found as elements in each geometric algebra. This will be explained in more detail later. The product defining a geometric algebra is called *geometric product* and is denoted by juxtaposition, e.g. $\boldsymbol{AB}$ for two multivectors $\boldsymbol{A}$ and $\boldsymbol{B}$. The geometric product of vectors consists of an outer ($\wedge$) product and an inner ($\cdot$) product. Their effect is to increase or to decrease the grade of the algebraic entities respectively.

To be more detailed, let $\mathbf{e}_i$ and $\mathbf{e}_j$ ($\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^{p,q,r}$ ) be two orthonormal basis vectors of the vector space. Then the geometric product for these vectors of the geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ is defined as

$$\mathbf{e}_i\mathbf{e}_j \;:=\; \begin{cases} 1 \in \mathbb{R} & \text{for} \quad i = j \in \{1, \ldots, p\} \\ -1 \in \mathbb{R} & \text{for} \quad i = j \in \{p+1, \ldots, p+q\} \\ 0 \in \mathbb{R} & \text{for} \quad i = j \in \{p+q+1, \ldots, n\} \\ \mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i & \text{for} \quad i \neq j. \end{cases} \tag{3.1}$$

The geometric product of the same two basis vectors leads to a scalar, whereas the geometric product of two different basis vectors leads to a new entity, which is called a *bivector*. This bivector represents the subspace, spanned by these two vectors. A vector space with signature $(p, q, r)$, $p \neq 0$, $q \neq 0$ is called pseudo-Euclidean. If $r \neq 0$ its metric is degenerate. Although the dual-quaternions which have some importance in kinematics are isomorphic to a degenerate geometric algebra, see [13], in this thesis only non-degenerate geometric algebras $\boldsymbol{\mathcal{G}}_{p,q,r} \simeq \boldsymbol{\mathcal{G}}_{p,q}$ are used with $r = 0$. Besides $\boldsymbol{\mathcal{G}}_{p,q,r} \simeq \boldsymbol{\mathcal{G}}_{p}$ is used if $r = 0$ and $q = 0$, which means that there is an Euclidean metric.

Geometric algebras can be expressed on the basis of graded elements. Scalars are of grade zero, vectors of grade one, bivectors of grade two, etc. A linear combination of elements of different grades is called a multivector $\boldsymbol{M}$ and can be expressed as

$$\boldsymbol{M} \;=\; \sum_{i=0}^{n} \langle \boldsymbol{M} \rangle_i, \tag{3.2}$$

where the operator $\langle . \rangle_s$ denotes the projection of a general multivector to the entities of grade $s$. The dimension of the subspace of grade $i$ is $\begin{pmatrix} n \\ i \end{pmatrix}$. A multivector of grade $i$ is called an $i$-blade if it can be written as the outer product of $i$ vectors. This means in general that every $i$-blade is a homogeneous multivector of grade $i$ but not vice versa. A multivector $\boldsymbol{A}$ of grade $i$ is sometimes written as $\boldsymbol{A}_{\langle i \rangle}$.

The inner $(\cdot)$ and outer $(\wedge)$ product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \langle \boldsymbol{\mathcal{G}}_{p,q} \rangle_1 \equiv \mathbb{R}^{p+q}$ are defined as

$$\boldsymbol{u} \cdot \boldsymbol{v} := \frac{1}{2}(\boldsymbol{u}\boldsymbol{v} + \boldsymbol{v}\boldsymbol{u}), \tag{3.3}$$

$$\boldsymbol{u} \wedge \boldsymbol{v} := \frac{1}{2}(\boldsymbol{u}\boldsymbol{v} - \boldsymbol{v}\boldsymbol{u}). \tag{3.4}$$

Here $\alpha = \boldsymbol{u} \cdot \boldsymbol{v} \in \mathbb{R}$ is a scalar, which is of grade zero, i.e. $\alpha \in \langle \boldsymbol{\mathcal{G}}_{p,q} \rangle_0$. Besides, $\boldsymbol{B} = \boldsymbol{u} \wedge \boldsymbol{v}$ is a bivector, i.e. $\boldsymbol{B} \in \langle \boldsymbol{\mathcal{G}}_{p,q} \rangle_2$.

As extension the inner product of a $r$-blade $\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r$ with a $s$-blade $\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s$ can be defined recursively by

$$\begin{aligned} &(\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \\ &\begin{cases} ((\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot \boldsymbol{v}_1) \cdot (\boldsymbol{v}_2 \wedge \ldots \wedge \boldsymbol{v}_s) & \text{if } r \geq s \\ (\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_{r-1}) \cdot (\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s)) & \text{if } r < s, \end{cases} \end{aligned} \tag{3.5}$$

with

$$\begin{aligned} &(\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot \boldsymbol{v}_1 = \\ &\sum_{i=1}^{r} (-1)^{r-i} \boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_{i-1} \wedge (\boldsymbol{u}_i \cdot \boldsymbol{v}_1) \wedge \boldsymbol{u}_{i+1} \wedge \ldots \wedge \boldsymbol{u}_r, \end{aligned} \tag{3.6}$$

$$\begin{aligned} &\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \\ &\sum_{i=1}^{s} (-1)^{i-1} \boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_{i-1} \wedge (\boldsymbol{u}_r \cdot \boldsymbol{v}_i) \wedge \boldsymbol{v}_{i+1} \wedge \ldots \wedge \boldsymbol{v}_s. \end{aligned} \tag{3.7}$$

This will become more clear in the next subsections.

For two blades $\boldsymbol{A}_{\langle r \rangle}$ and $\boldsymbol{B}_{\langle s \rangle}$ with non-zero grade $r$ and $s \in \mathbb{N}$ the inner and outer product can be written as

$$\boldsymbol{A}_{\langle r \rangle} \cdot \boldsymbol{B}_{\langle s \rangle} = \langle \boldsymbol{A}\boldsymbol{B} \rangle_{|r-s|} \quad \text{and} \tag{3.8}$$

$$\boldsymbol{A}_{\langle r \rangle} \wedge \boldsymbol{B}_{\langle s \rangle} = \langle \boldsymbol{A}\boldsymbol{B} \rangle_{r+s}, \tag{3.9}$$

with the following additional rules:

1. If $r = 0$ or $s = 0$, the inner product is zero.

2. If $r + s > n$, the outer product is zero.

The blades of highest grade are $n$-blades, called *pseudoscalars*. Pseudoscalars differ from each other by a nonzero scalar only. For non-degenerated geometric algebras there exists two unit $n$-blades, called the *unit pseudoscalars* $\pm\boldsymbol{I}$. The unit pseudoscalars are often indexed by the generating vector spaces of the geometric algebras, for example $\boldsymbol{I}_E$, $\boldsymbol{I}_P$ and $\boldsymbol{I}_C$ represent the unit pseudoscalars of the algebras for the Euclidean, projective and conformal space, respectively.

The magnitude $[\boldsymbol{P}]$ of a pseudoscalar $\boldsymbol{P}$ is a scalar. It is called *bracket* of $\boldsymbol{P}$ and is defined by

$$[\boldsymbol{P}] \;:=\; \boldsymbol{P}\boldsymbol{I}^{-1}. \tag{3.10}$$

For the bracket determined by $n$ vectors, it is convenient to write

$$\begin{aligned}[\boldsymbol{v}_1 \ldots \boldsymbol{v}_n] &= [\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n] \\ &= (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n)\boldsymbol{I}^{-1}. \end{aligned} \tag{3.11}$$

This can also be taken as a definition of a determinant, well known from matrix calculus.

The *dual* $\boldsymbol{X}^\star$ of a $r$-blade $\boldsymbol{X}$ is defined by

$$\boldsymbol{X}^\star \;:=\; \boldsymbol{X}\boldsymbol{I}^{-1}. \tag{3.12}$$

It follows that the dual of a $r$-blade is a $(n-r)$-blade.

The *reverse* $\widetilde{\boldsymbol{A}_{\langle s\rangle}}$ of a $s$-blade $\boldsymbol{A}_{\langle s\rangle} = \boldsymbol{a}_1 \wedge \ldots \wedge \boldsymbol{a}_s$ is defined as the reverse outer product of the vectors $\boldsymbol{a}_i$,

$$\begin{aligned}\widetilde{\boldsymbol{A}_{\langle s\rangle}} &= (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \ldots \wedge \boldsymbol{a}_{s-1} \wedge \boldsymbol{a}_s)^\sim \\ &= \boldsymbol{a}_s \wedge \boldsymbol{a}_{s-1} \wedge \ldots \wedge \boldsymbol{a}_2 \wedge \boldsymbol{a}_1. \end{aligned} \tag{3.13}$$

The *join* $\boldsymbol{A}\dot{\wedge}\boldsymbol{B}$ is the pseudoscalar of the space given by the sum of spaces spanned by $\boldsymbol{A}$ and $\boldsymbol{B}$.

For blades $\boldsymbol{A}$ and $\boldsymbol{B}$ the *dual shuffle* product $\boldsymbol{A} \vee \boldsymbol{B}$ is defined by the DeMorgan rule

$$(\boldsymbol{A} \vee \boldsymbol{B})^\star \;:=\; \boldsymbol{A}^\star \dot{\wedge} \boldsymbol{B}^\star. \tag{3.14}$$

For blades $\boldsymbol{A}$ and $\boldsymbol{B}$ it is possible to use the join to express *meet* operations:

**Definition 3.1** *Let* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *be two arbitrary blades and let* $\boldsymbol{J} = \boldsymbol{A} \dot{\wedge} \boldsymbol{B}$*, then*

$$\boldsymbol{A} \vee \boldsymbol{B} \;:=\; \left(\boldsymbol{A}\boldsymbol{J}^{-1} \wedge \boldsymbol{B}\boldsymbol{J}^{-1}\right)\boldsymbol{J}. \tag{3.15}$$

The meet $\vee$, also called the shuffle product, is a common factor of $\boldsymbol{A}$ and $\boldsymbol{B}$ with the highest grade. The meet will be used in chapter 3.2 for incidence estimation of points, lines and planes.

For later computations, the commutator $\underline{\times}$ product and the anticommutator $\overline{\times}$ product for any two multivectors is used,

$$\boldsymbol{A}\boldsymbol{B} \;=\; \frac{1}{2}(\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A}) + \frac{1}{2}(\boldsymbol{A}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}) =: \boldsymbol{A}\overline{\times}\boldsymbol{B} + \boldsymbol{A}\underline{\times}\boldsymbol{B}. \tag{3.16}$$

The reader should consult [133] to become more familiar with the commutator and anticommutator product. Their role is to separate the symmetric part of the geometric product from the antisymmetric one.

Now follows an introduction to algebras for the Euclidean, projective and conformal space.

## 3.1 The Euclidean geometric algebra

The algebra $\boldsymbol{\mathcal{G}}_3$, which is derived from $\mathbb{R}^3$, i.e. $n = p = 3$, is the smallest and simplest one in this thesis. This algebra is suited to represent entities and operations in 3D Euclidean space. Therefore it is called EGA as abbreviation for Euclidean geometric algebra. Let $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ be three orthonormal basis vectors of the 3D Euclidean space. Following e.g. [20], such a basis can always be found[2]. The geometric algebra of the 3D Euclidean space consists of $2^3 = 8$ basis elements, derived from the three basis vectors:

$$\boldsymbol{\mathcal{G}}_3 = span\{1\ ,\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3,\ \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12},\ \mathbf{e}_{123}\}. \tag{3.17}$$

The elements $\mathbf{e}_{ij} = \mathbf{e}_i\mathbf{e}_j = \mathbf{e}_i \wedge \mathbf{e}_j$ are the unit bivectors and the element

$$\mathbf{e}_{123} \;=\; \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \tag{3.18}$$

$$=:\; \boldsymbol{I}_E \tag{3.19}$$

is a trivector, called Euclidean unit pseudoscalar, which squares to $-1$ and commutes with scalars, vectors and bivectors. As visualized in figure 3.1, each bivector represents the plane spanned by two vectors and the trivector represents the unit volume. To enlighten the rules of the geometric prod-

[2] This follows from the orthonormalization theorem of E. Schmidt.

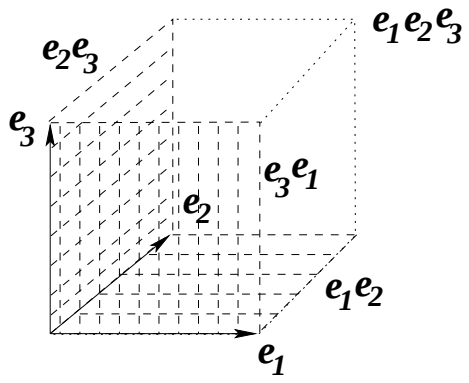



Fig. 3.1: A basis of the 3D Euclidean geometric algebra contains three basis vectors, three bivectors, one scalar and one pseudoscalar.

uct introduced in equation (3.1), the geometric product of two vectors is calculated as an example:

$$\begin{aligned}
\boldsymbol{uv} &= (u_1\mathbf{e}_1 + u_2\mathbf{e}_2 + u_3\mathbf{e}_3)(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) \\
&= u_1\mathbf{e}_1(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) + u_2\mathbf{e}_2(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) + \\
&\quad + u_3\mathbf{e}_3(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) \\
&= u_1v_1 + u_2v_2 + u_3v_3 + (u_1v_2 - u_2v_1)\mathbf{e}_{12} + (u_3v_1 - u_1v_3)\mathbf{e}_{31} + \\
&\quad + (u_2v_3 - u_3v_2)\mathbf{e}_{23} \\
&= \boldsymbol{u}\cdot\boldsymbol{v} + \boldsymbol{u}\wedge\boldsymbol{v}. \qquad (3.20)
\end{aligned}$$

Thus the geometric product of two vectors leads to a scalar representing the inner product of the two vectors (corresponding to the scalar product of these vectors in matrix calculus), and to a bivector representing the outer product of two vectors. The dual of the bivector corresponds to the cross product of the two vectors. The inner product of a bivector ($\boldsymbol{a}\wedge\boldsymbol{b}$) with a vector $\boldsymbol{c}$ leads to another vector,

$$\begin{aligned}
(\boldsymbol{a}\wedge\boldsymbol{b})\cdot\boldsymbol{c} &\overset{3.6}{=} -(\boldsymbol{a}\cdot\boldsymbol{c})\wedge\boldsymbol{b} + \boldsymbol{a}\wedge(\boldsymbol{b}\cdot\boldsymbol{c}) \\
&= -(\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b} + (\boldsymbol{b}\cdot\boldsymbol{c})\boldsymbol{a}, \qquad (3.21)
\end{aligned}$$

and is the equivalent formulation of the cross product rule for the 3D case[3],

$$(\boldsymbol{a}\times\boldsymbol{b})\times\boldsymbol{c} = \langle\boldsymbol{a},\boldsymbol{c}\rangle\boldsymbol{b} - \langle\boldsymbol{b},\boldsymbol{c}\rangle\boldsymbol{a}. \qquad (3.22)$$

[3] In this example, $\langle,\rangle$ denotes the scalar product of vectors, and $\times$ denotes the cross product.

The inner product of two bivectors leads to a scalar,

$$\begin{aligned}(\boldsymbol{a}\wedge\boldsymbol{b})\cdot(\boldsymbol{c}\wedge\boldsymbol{d}) &\overset{3.5}{=} ((\boldsymbol{a}\wedge\boldsymbol{b})\cdot\boldsymbol{c})\cdot\boldsymbol{d}\\ &\overset{3.21}{=} (-(\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b}+(\boldsymbol{b}\cdot\boldsymbol{c})\boldsymbol{a})\cdot\boldsymbol{d}\\ &= -(\boldsymbol{a}\cdot\boldsymbol{c})(\boldsymbol{b}\cdot\boldsymbol{d})+(\boldsymbol{b}\cdot\boldsymbol{c})(\boldsymbol{a}\cdot\boldsymbol{d}) \qquad (3.23)\end{aligned}$$

and results (for the 3D case) in an equivalent expression of the *Lagrange identity* for the cross products of 3D vectors,

$$\langle(\boldsymbol{a}\times\boldsymbol{b}),(\boldsymbol{c}\times\boldsymbol{d})\rangle = \langle\boldsymbol{a},\boldsymbol{c}\rangle\langle\boldsymbol{b},\boldsymbol{d}\rangle-\langle\boldsymbol{b},\boldsymbol{c}\rangle\langle\boldsymbol{a},\boldsymbol{d}\rangle. \qquad (3.24)$$

Note that the outer product is more general than the cross product, since it can be applied to vector spaces of any dimension and any signature.

### 3.1.1 Representation of points, lines and planes in the Euclidean geometric algebra

Points, lines and planes of the 3D space can all be modeled in the algebra $\boldsymbol{\mathcal{G}}_3$. A point, representing a position in the 3D space, can simply be expressed by a linear combination of the three basis vectors,

$$\boldsymbol{u} = u_1\mathbf{e}_1+u_2\mathbf{e}_2+u_3\mathbf{e}_3. \qquad (3.25)$$

A line can be represented as an inhomogeneous multivector by using a vector $\boldsymbol{r}$ for the direction and a bivector $\boldsymbol{m}$ containing the moment, as outer product of a point $\boldsymbol{x}$ on the line and the direction $\boldsymbol{r}$ of the line. This means a line is represented by its Plücker coordinates [21],

$$\begin{aligned}\boldsymbol{l} &= \boldsymbol{r}+\boldsymbol{x}\wedge\boldsymbol{r}\\ &= \boldsymbol{r}+\boldsymbol{m}. \qquad (3.26)\end{aligned}$$

Incidence of a point with a line can be expressed by the kernel of a function $\mathcal{F}_{XL}$ as follows,

$$\begin{aligned}\boldsymbol{p}\in\boldsymbol{l} &\Leftrightarrow \mathcal{F}_{XL}(\boldsymbol{p},\boldsymbol{l})=0\\ &\Leftrightarrow (\boldsymbol{p}\wedge\boldsymbol{r})-\boldsymbol{m}=0. \qquad (3.27)\end{aligned}$$

A plane can be represented by an entity one grade higher than a line. In terms of the Hesse distance $d$ from the origin to the plane (coded by the

Euclidean pseudoscalar) and the unit bivector direction $\boldsymbol{n}$ from the origin to the plane, a plane is represented by

$$\boldsymbol{p} \;=\; \boldsymbol{n} + \boldsymbol{I}_E d. \tag{3.28}$$

Thus a plane is an inhomogeneous multivector consisting of a bivector and a trivector. The incidence of a point with a plane can be expressed in the following way,

$$\begin{aligned} \boldsymbol{x} \in \boldsymbol{p} \;&\Leftrightarrow\; \mathcal{F}_{XP}(\boldsymbol{x}, \boldsymbol{p}) = 0 \\ &\Leftrightarrow\; (\boldsymbol{x} \wedge \boldsymbol{n}) - \boldsymbol{I}_E d = 0. \end{aligned} \tag{3.29}$$

It is easy to recognize that the representation of lines and planes is more complicated than that of points. Also the constraint equations expressing the incidence relation are not compact or simple. This has its reason in the fact, that so far no origin of the vector space is modeled within the geometric algebra. In vector calculus this can formally be done by introducing an additional (or *homogeneous*) coordinate. Such an extension will also be done in section 3.2 for modeling the projective space in a Clifford algebra.

### 3.1.2 Rotations and translations in the Euclidean space

Multiplication of the three basis vectors $\mathbf{e}_i$ with $\boldsymbol{I}_E$ results in the three basis bivectors $\boldsymbol{I}_E\mathbf{e}_i$. These bivectors rotate vectors in their own plane by $90^o$, e.g.

$$(\boldsymbol{I}_E\mathbf{e}_3)\mathbf{e}_2 \;=\; \mathbf{e}_{123}\mathbf{e}_3\mathbf{e}_2 = \mathbf{e}_{12}\mathbf{e}_2 = \mathbf{e}_1, \tag{3.30}$$

or

$$(\boldsymbol{I}_E\mathbf{e}_1)\mathbf{e}_2 \;=\; \mathbf{e}_{123}\mathbf{e}_1\mathbf{e}_2 = \mathbf{e}_{23}\mathbf{e}_2 = -\mathbf{e}_1. \tag{3.31}$$

Note that since the basis vectors are orthonormal, it is equivalent to write $\mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j$ for $i \neq j$. The basis bivectors square to $-1$, and can be identified with the unit vectors $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ of the quaternion algebra $\mathbb{H}$ with the famous Hamilton relations [21]

$$\boldsymbol{i}^2 \;=\; \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1. \tag{3.32}$$

The bivectors of the geometric algebra can be used to represent rotations of points in the 3D space. A *rotor* $\boldsymbol{R}$ is an even grade element of the algebra

$\boldsymbol{\mathcal{G}}_3$ which satisfies $\boldsymbol{R}\widetilde{\boldsymbol{R}} = 1$. Since the even grade elements of $\boldsymbol{\mathcal{G}}_3$ are scalars and bivectors, a rotor $\boldsymbol{R}$ and its reverse $\widetilde{\boldsymbol{R}}$ is given by

$$\boldsymbol{R} = \underbrace{u_0}_{scalar} + \underbrace{u_1\mathbf{e}_{23} + u_2\mathbf{e}_{31} + u_3\mathbf{e}_{12}}_{bivectors}, \tag{3.33}$$

$$\widetilde{\boldsymbol{R}} = \underbrace{u_0}_{scalar} - \underbrace{u_1\mathbf{e}_{23} - u_2\mathbf{e}_{31} - u_3\mathbf{e}_{12}}_{bivectors}. \tag{3.34}$$

By using the Euler representation of a rotor,

$$\begin{aligned} \boldsymbol{R} &= \exp\left(-\frac{\theta}{2}\boldsymbol{n}\right) \\ &= \cos\left(\frac{\theta}{2}\right) - \boldsymbol{n}\sin\left(\frac{\theta}{2}\right), \end{aligned} \tag{3.35}$$

this takes on geometric significance. Here $\boldsymbol{n}$ is a unit bivector representing the plane of the rotation (its dual $\boldsymbol{n}^\star$ corresponds to the rotation axis) and $\theta \in \mathbb{R}$ represents the amount of rotation. The negative value $-\theta$ in the rotor is used to gain a counter clockwise rotation and therewith a mathematically positive rotation. The rotation of a point, represented by its vector $\boldsymbol{x}$, can be carried out by multiplying the rotor $\boldsymbol{R}$ from the left and its reverse from the right to the point $\boldsymbol{x}$,

$$\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}}. \tag{3.36}$$

Such a multiplication is also called *rotor product*. The rotor product is a special case of a *versor product* [79]. A rotor represents the group $SO(3)$ in EGA. Thus the geometric product of two rotors $\boldsymbol{R} = \boldsymbol{R}_2\boldsymbol{R}_1$ results in a new rotor. From this follows

$$\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}} = (\boldsymbol{R}_2\boldsymbol{R}_1)\,\boldsymbol{x}\left(\widetilde{\boldsymbol{R_1}}\widetilde{\boldsymbol{R_2}}\right). \tag{3.37}$$

In contrast to rotation matrices in $\mathbb{R}^3$, rotors can not only be applied to points, but to all types of geometric objects, independent of the grade and the dimension of the space.

The exponential function of multivectors $\boldsymbol{m}$ can be expressed by the series expansion,

$$\exp(\boldsymbol{m}) = \sum_{k=0}^{\infty} \frac{\boldsymbol{m}^k}{k!}. \tag{3.38}$$

Derivating the rotor $\boldsymbol{R}$ with respect to $\theta$ leads to

$$\begin{aligned}\frac{\partial \boldsymbol{R}}{\partial \theta} &= \frac{\partial \exp(-\frac{\theta}{2}\boldsymbol{n})}{\partial \theta} = -\frac{1}{2}\boldsymbol{n}\exp\left(-\frac{\theta}{2}\boldsymbol{n}\right) \\ &= -\frac{1}{2}\sin\left(\frac{\theta}{2}\right) - \frac{1}{2}\boldsymbol{n}\cos\left(\frac{\theta}{2}\right). \end{aligned} \tag{3.39}$$

The derivation of the rotation applied on $\boldsymbol{x}$ leads to

$$\frac{\partial \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}}}{\partial \theta} = \frac{\partial \boldsymbol{R}}{\partial \theta}\boldsymbol{x}\widetilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x}\frac{\partial \widetilde{\boldsymbol{R}}}{\partial \theta}. \tag{3.40}$$

In contrast to rotations there exists no multiplicative way to formalize translations in the Euclidean geometric algebra. The only possibility is to express translations in an additive way, e.g. a point $\boldsymbol{x}$ is translated with a translation vector $\boldsymbol{t}$, by

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}. \tag{3.41}$$

This results from the fact that translations in $\mathbb{R}^3 = \langle \boldsymbol{\mathcal{G}}_3 \rangle_1$ constitute the additive group $\mathbb{R}^3$. Therefore composite translation follows the rule $\boldsymbol{t} = \boldsymbol{t}_1 + \boldsymbol{t}_2$. Another problem concerns the linearity of both operations. A rotation is a linear operation: Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be any multivectors of $\boldsymbol{\mathcal{G}}_3$, then

$$\mathcal{R}\{\boldsymbol{x} + \boldsymbol{y}\} = \mathcal{R}\{\boldsymbol{x}\} + \mathcal{R}\{\boldsymbol{y}\}. \tag{3.42}$$

Yet translation does not have this linearity property: For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ representing points of $\langle \boldsymbol{\mathcal{G}}_3 \rangle_1$, it follows

$$\mathcal{T}\{\boldsymbol{x} + \boldsymbol{y}\} \neq \mathcal{T}\{\boldsymbol{x}\} + \mathcal{T}\{\boldsymbol{y}\}. \tag{3.43}$$

These different behaviors cause problems in representing the rigid body motion of an object in EGA as linear operation. In general the movement of a rigid body, also called a rigid displacement, may include both rotation and translation in the following way: Let be $\boldsymbol{x}', \boldsymbol{x} \in \langle \boldsymbol{\mathcal{G}}_3 \rangle_1$, then

$$\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}} + \boldsymbol{t}. \tag{3.44}$$

A spatial displacement $\mathcal{D} = (\boldsymbol{R}, \boldsymbol{t})$ belongs to the special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$. Thus a composite displacement $\mathcal{D} = \mathcal{D}_2\mathcal{D}_1$ exists with

$$\mathcal{D} = (\boldsymbol{R}, \boldsymbol{t}) = (\boldsymbol{R_2}, \boldsymbol{t_2})(\boldsymbol{R_1}, \boldsymbol{t_1}) = (\boldsymbol{R}_2\boldsymbol{R}_1, \boldsymbol{R}_2\boldsymbol{t}_1 + \boldsymbol{t_2}). \tag{3.45}$$

But regrettably, because of the non-linear behavior of the translation displacement it is no linear operation in $\boldsymbol{\mathcal{G}}_3$, neither for points nor for any other entities. Fortunately, there are other algebraic embeddings which result in linearization with respect to points and other entities. While so far either point or line based transformations for rigid displacements have been distinguished [141], in this thesis a third category is introduced which is based on spheres [110], see section 3.3.

## 3.2 The projective geometric algebra

By using homogeneous coordinates the dimension of the vector space increases by one and the corresponding geometric algebra is of dimension $2^4 = 16$. The elements are now scalars, vectors, bivectors, trivectors and pseudoscalars. To model 3D projective geometry in a geometric algebra four basis vectors are needed. The signature of the derived vector space will be unimportant, therefore it is free to choose. Since I will later use $\boldsymbol{\mathcal{G}}_{3,1}$ (chapter 4), the geometric algebra $\boldsymbol{\mathcal{G}}_{3,1}$ is introduced to represent the projective space. Here the additional basis vector $\mathbf{e}_-$ denotes the homogeneous component. Because $\mathbf{e}_-^2 = -1$, this basis vector induces a Minkowski metric. The algebra $\boldsymbol{\mathcal{G}}_{3,1}$ contains the following elements

$$\begin{aligned}\boldsymbol{\mathcal{G}}_{3,1} \;=\; span\{1,\;\; \mathbf{e}_-, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3,\;\; \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12},\quad \mathbf{e}_{-1}, \mathbf{e}_{-2}, \mathbf{e}_{-3},\\ \mathbf{e}_{123}, \mathbf{e}_{-23}, \mathbf{e}_{-31}, \mathbf{e}_{-12},\;\; \mathbf{e}_{-123}\}.\end{aligned} \tag{3.46}$$

Note that

$$\mathbf{e}_{-123} \;=\; \mathbf{e}_- \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 =: \boldsymbol{I}_P, \tag{3.47}$$

and

$$\mathbf{e}_{-123}^2 = -1. \tag{3.48}$$

### 3.2.1 Representation of points, lines and planes in the projective geometric algebra

In contrast to the Euclidean geometric algebra $\boldsymbol{\mathcal{G}}_3$, the projective geometric algebra $\boldsymbol{\mathcal{G}}_{3,1}$ (PGA) can be used to model projective geometry. In PGA points, lines and planes can be represented as $r$-blades, i.e. homogeneous multivectors of grade $r$. Thus the duality operator defined in equation (3.12) is of special importance since it transforms geometric entities to their duals.

A point can be represented by a 1-blade. The basis vector $\mathbf{e}_-$ represents the homogeneous component of the point. Thus the point $\boldsymbol{x}$ given in $\boldsymbol{\mathcal{G}}_3$ can be represented in $\boldsymbol{\mathcal{G}}_{3,1}$ by

$$\boldsymbol{X} \;=\; \boldsymbol{x} + \mathbf{e}_-. \tag{3.49}$$

Since $\boldsymbol{X} \wedge \boldsymbol{X} = 0$, it is simple to get

$$\boldsymbol{X} \wedge \lambda \boldsymbol{X} \;=\; 0 \quad \forall \lambda \in \mathbb{R} \setminus \{0\}. \tag{3.50}$$

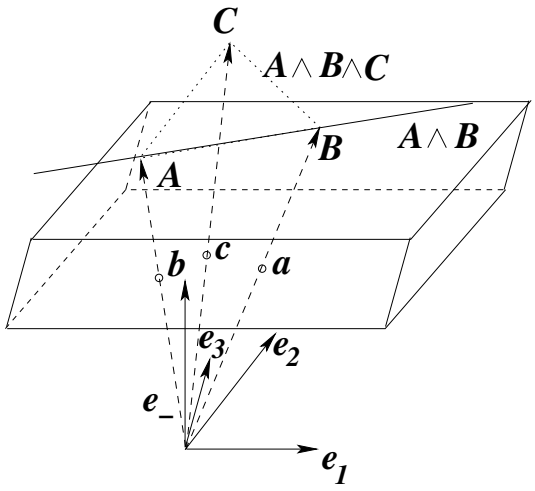



Fig. 3.2: The 3D projective space contains four basis vectors. Projective points are 1D-subspaces in the 4D space. Two projective points define a projective line and three projective points define a projective plane.

For this reason the outer product is used to define the equivalence class of points in the projective space (see A.1 of the appendix for a definition of the projective space in classical matrix calculus). All vectors $\boldsymbol{X}$ represent a point $\boldsymbol{A}$ if $\boldsymbol{A} \wedge \boldsymbol{X} = 0$. The term $\boldsymbol{A}\wedge$ can be viewed as a linear operator. This means, that the so-called *outer product null space* defines the incidence of two entities, similar to [77].

A line can be represented by the outer product of two points, leading to a 2-blade

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \\ &= (\boldsymbol{x}_1 + \mathbf{e}_-) \wedge (\boldsymbol{x}_2 + \mathbf{e}_-) \\ &= \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 + (\boldsymbol{x}_1 - \boldsymbol{x}_2)\mathbf{e}_- \\ &= \boldsymbol{m} - \boldsymbol{r}\mathbf{e}_-. \end{aligned} \tag{3.51}$$

The line $\boldsymbol{L}$ contains the moment $\boldsymbol{m}$ and the (negative) direction $\boldsymbol{r}$. Therefore it corresponds directly to the Plücker representation (up to scale factor) [21]. Being a 2-blade, the line contains 6 bivector components.

A plane can be represented by the outer product of three points, leading to a 3-blade

$$\boldsymbol{P} = \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \wedge \boldsymbol{X}_3$$

$$\begin{aligned} &= (\boldsymbol{x}_1 + \mathbf{e}_-) \wedge (\boldsymbol{x}_2 + \mathbf{e}_-) \wedge (\boldsymbol{x}_3 + \mathbf{e}_-) \\ &= \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 \wedge \boldsymbol{x}_3 + (\boldsymbol{x}_1 - \boldsymbol{x}_2) \wedge (\boldsymbol{x}_1 - \boldsymbol{x}_3)\mathbf{e}_- \\ &= d\boldsymbol{I}_E + \boldsymbol{n}\mathbf{e}_-. \end{aligned} \tag{3.52}$$

This representation corresponds to the Hesse description of planes (up to scale factor), formalizing a plane by the normal $\boldsymbol{n}$ (as bivector) of the plane, and the Hesse distance $d$ of the origin to the plane. The representation of the entities is visualized in figure 3.2: A 4D homogeneous point corresponds to a ray. Two points define a line and three points a plane.

As can be seen, the generation of higher order entities is much more natural than in the algebra of the Euclidean space.

The outer product of two blades is non-vanishing iff[4] their supports have zero intersection. This can be used to prove an incidence relation [81], e.g. a point $\boldsymbol{X}$ is on a line $\boldsymbol{L}$ iff

$$\boldsymbol{X} \wedge \boldsymbol{L} = 0. \tag{3.53}$$

To calculate intersections, the definition 3.1 is recalled: Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two arbitrary blades and let $\boldsymbol{J} = \boldsymbol{A} \dot{\wedge} \boldsymbol{B}$, then

$$(\boldsymbol{A} \vee \boldsymbol{B}) := \left(\boldsymbol{A}\boldsymbol{J}^{-1} \wedge \boldsymbol{B}\boldsymbol{J}^{-1}\right)\boldsymbol{J}. \tag{3.54}$$

Thus the meet and the join operators provide the desired operations in the algebra of subspaces of a vector space. The join can be used to determine the *union* of subspaces and the meet product can be used to determine the *intersection* of subspaces. Note that the incidence operations always lead to entities in the projective space. To transform e.g. a projective representation of a point to a Euclidean one, the entities have to be rescaled in their homogeneous component, as shown in chapter 4.

To make clear the meet and join operators, in the following example the intersection of two lines will be calculated for the 2D projective case: Let

$$\begin{aligned} \boldsymbol{a}_1 &= \mathbf{e}_-, & \boldsymbol{b}_1 &= 2\mathbf{e}_1 + 2\mathbf{e}_2 + \mathbf{e}_-, \\ \boldsymbol{a}_2 &= 2\mathbf{e}_2 + \mathbf{e}_-, & \boldsymbol{b}_2 &= 2\mathbf{e}_1 + \mathbf{e}_-, \end{aligned} \tag{3.55}$$

be four vectors, defining two lines as shown in figure 3.3. The lines are given as

$$\boldsymbol{l}_1 = \boldsymbol{a}_1 \wedge \boldsymbol{b}_1 = 2\mathbf{e}_{-1} + 2\mathbf{e}_{-2}, \tag{3.56}$$

$$\boldsymbol{l}_2 = \boldsymbol{a}_2 \wedge \boldsymbol{b}_2 = 4\mathbf{e}_{12} + 2\mathbf{e}_{2-} + 2\mathbf{e}_{-1}. \tag{3.57}$$

[4] if and only if

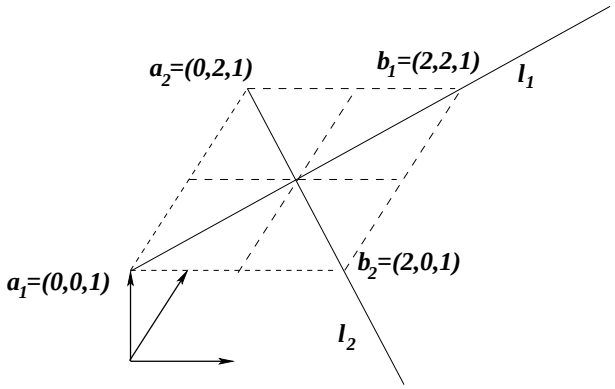



Fig. 3.3: Visualization of the intersection of two lines.

The join of these two lines is

$$\boldsymbol{J} \;=\; \mathbf{e}_{-12}. \tag{3.58}$$

Since $\boldsymbol{J}^2 = 1$, its inverse is $\boldsymbol{J}$ itself. Then the geometric product of the lines with the join $\boldsymbol{J}$ leads to

$$\boldsymbol{l}_1\boldsymbol{J} = 2\mathbf{e}_2 - 2\mathbf{e}_1, \qquad \boldsymbol{l}_2\boldsymbol{J} = 2\mathbf{e}_2 + 2\mathbf{e}_1 + 3\mathbf{e}_-. \tag{3.59}$$

Computation of their outer product multiplied with the join $\boldsymbol{J}$ leads to

$$((\boldsymbol{l}_1\boldsymbol{J}) \wedge (\boldsymbol{l}_2\boldsymbol{J}))\boldsymbol{J} \;=\; 8\mathbf{e}_- + 8\mathbf{e}_1 + 8\mathbf{e}_2. \tag{3.60}$$

Rescaling with the homogeneous component leads to

$$\boldsymbol{l}_1 \vee \boldsymbol{l}_2 \;\simeq\; \mathbf{e}_1 + \mathbf{e}_2 \tag{3.61}$$

as Euclidean intersecting point. This is consistent with the scenario visualized in figure 3.3.

The advantage of the algebra $\boldsymbol{\mathcal{G}}_{3,1}$ for the projective space in comparison to the algebra $\boldsymbol{\mathcal{G}}_3$ for the Euclidean space is that the representation of the entities is much more natural and is provided by the subspace concepts. This leads to a simple formulation of the duality concept in projective geometry and to compact descriptions of joins and meets of subspaces, just by applying a suitable operator. These concepts are for example used in Ch. Perwass' thesis [131] to formalize stereo geometry, fundamental matrices and trifocal tensors.

Projective transformations are more general than Euclidean transformations, since they also include other transformations like scaling or shearing.

For this thesis only Euclidean transformations (rigid body motions) are of interest. Therefore it is necessary to restrict the projective transformations in a second processing step. This leads to numerical problems and so there is need for an algebraic embedding which enables the restriction of the transformations on a rigid body motion in a better way. The commonly used algebra so far is the *dual quaternion* algebra, which is isomorphic to the motor algebra $\mathcal{G}^{+}_{3,0,1}$ [14]. But since it contains null spaces, the duality concepts cannot be applied any more[5]. The aim is now to proceed to the conformal algebra, which can handle these problems. One important property of the conformal geometric algebra is that it is non-degenerate, but contains an artificially generated null space. The algebra for projective geometry is furthermore a subset of this (extended) algebra. Since the null space is constructed from a Minkowski subspace, it is possible to switch between null spaces and non-null spaces, an important fact for the next sections.

## 3.3 The conformal geometric algebra

In PGA projective transformations can be expressed, but the restriction to the Euclidean transformations leads to problems. Therefore there is need to introduce another geometric embedding. The use of the conformal geometric algebra [110, 111, 80] is motivated by introducing stereographic projections [62].

### 3.3.1 Stereographic projections

Simply speaking, a stereographic projection is one way to generate a flat map of the earth. Taking the earth as a 3D sphere, any map must distort shapes or sizes to some degree. The rule for a stereographic projection has a clear geometric description and is visualized for the 1D case in figure 3.4: Think of the earth as a transparent sphere, intersected on the equator by an *equatorial plane*. Now imagine a light bulb at the *north pole* $\boldsymbol{n}$, which shines through the sphere. Each point on the sphere casts a shadow on the paper and that is where it is drawn on the map.

Before introducing a formalization in terms of geometric algebra, the basic properties of stereographic projections in classical vector calculus are derived.

[5] The inverse pseudoscalar does not exist, since $\boldsymbol{I}^2 = 0$.

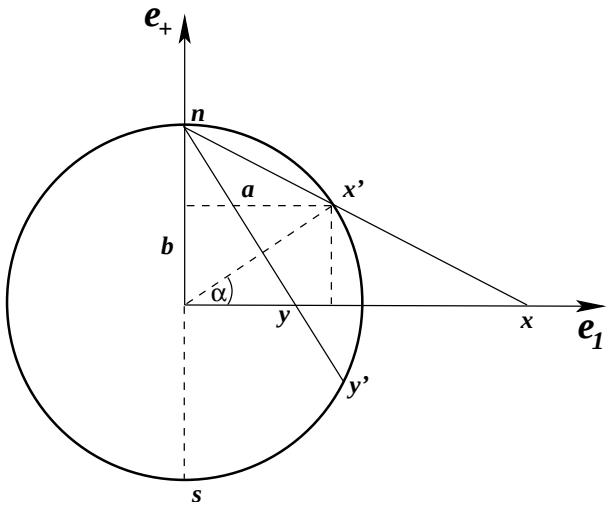



Fig. 3.4: Visualization of a stereographic projection for the 1D case: Points on the circle are projected onto the line. Note that the north pole $\boldsymbol{n}$ projects to the points at infinity, and the south pole $\boldsymbol{s}$ projects to the origin.

The main questions are: (a) How to project a point on the sphere to the plane? And vice versa: (b) How to project a point on the plane to the sphere?

To simplify the calculations for answering these questions they are restricted to the 1D case as shown in figure 3.4. Let there be two orthonormal basis vectors $\{\mathbf{e}_1, \mathbf{e}_+\}$ and let the radius of the circle be $\rho = 1$. Note that $\mathbf{e}_+$ is an additional vector to the one-dimensional vector space $\mathbf{e}_1$ with $\mathbf{e}_+^2 = \mathbf{e}_1^2 = 1$.

(a) A point $\boldsymbol{x}'$ on the circle is given by its angle $\alpha$:

$$\begin{aligned} \boldsymbol{x}' &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \cos(\alpha)\mathbf{e}_1 + \sin(\alpha)\mathbf{e}_+. \end{aligned} \tag{3.62}$$

To project the point $\boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+$ on the circle to a point on the $\mathbf{e}_1$-axis the intercept theorems can be applied to gain

$$\frac{x}{1} = \frac{a}{1-b} = \left(\frac{\cos(\alpha)}{1-\sin(\alpha)}\right). \tag{3.63}$$

The point on the line has the coordinates

$$\boldsymbol{x} = \left(\frac{\cos(\alpha)}{1-\sin(\alpha)}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{3.64}$$

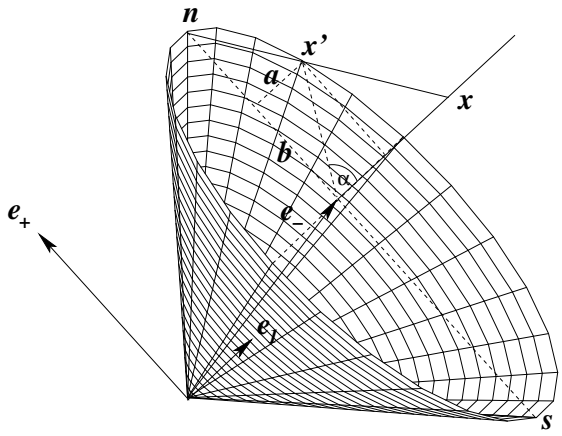



Fig. 3.5: Visualization of the homogeneous model for stereographic projections for the 1D case. All stereographically projected points lie on a cone, which is a null-cone in the Minkowski space. Note that in comparison to figure 3.4 the coordinate axes are rotated and drawn perspectively.

(b) To project a point $x\mathbf{e}_1$ ($x \in \mathbb{R}$) onto the circle means to calculate appropriate factors $a, b \in [0, \ldots, 1]$. Therefore the following equalities apply:

$$x = \frac{a}{1-b}, \tag{3.65}$$

$$a^2 + b^2 = 1. \tag{3.66}$$

Now it is possible to calculate

$$(3.65) \rightarrow \quad a = x(1-b), \tag{3.67}$$

$$(3.66) \rightarrow \quad a^2 = (1+b)(1-b). \tag{3.68}$$

This leads to

$$\begin{aligned} x^2(1-b)^2 &= (1+b)(1-b) \\ \Leftrightarrow x^2(1-b) &= (1+b) \\ \Leftrightarrow b &= \frac{x^2-1}{x^2+1}. \end{aligned} \tag{3.69}$$

Resubstituting in (3.67) results in

$$a = x(1-b) = \frac{2x}{x^2+1}. \tag{3.70}$$

Therefore the projection on the circle can be written as

$$\begin{aligned} \boldsymbol{x}' &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+. \end{aligned} \tag{3.71}$$

Using homogeneous coordinates leads to a homogeneous representation of the point on the circle as

$$\boldsymbol{x}' = x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}_3. \tag{3.72}$$

The vector $\boldsymbol{x}$ is mapped to

$$\boldsymbol{x} \Rightarrow \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_3. \tag{3.73}$$

In [110] $\mathbf{e}_3$ is defined to have a negative signature, and therefore $\mathbf{e}_3$ is replaced with $\mathbf{e}_-$, whereby $\mathbf{e}_-^2 = -1$. This has the advantage that in addition to using a homogeneous representation of points, they are further embedded in a Minkowski space. Euclidean points, stereographically projected onto the circle in figure 3.5, are then represented by the set of null vectors in the new space. A Euclidean point is mapped to the conformal space by

$$\boldsymbol{x} \Rightarrow \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_-, \tag{3.74}$$

with

$$(\boldsymbol{x}')^2 = a^2 + b^2 - 1 = 0. \tag{3.75}$$

The coordinates $(a, b)$ are the coordinates of a point on the unit circle. Note that each point in Euclidean space is in fact represented by a line of null vectors in the new space: the scaled versions of the null vector on the unit sphere.

In [110] it is shown that the conformal group of $n$-dimensional Euclidean space $\mathbb{R}^n$ is isomorphic to the Lorentz group of $\mathbb{R}^{n+1,1}$. Furthermore, the geometric algebra $\boldsymbol{\mathcal{G}}_{n+1,1}$ of $\mathbb{R}^{n+1,1}$ has a spinor representation of the Lorentz group. Therefore, any conformal transformation of the $n$-dimensional Euclidean space is represented by a spinor in $\boldsymbol{\mathcal{G}}_{n+1,1}$, the conformal geometric algebra. Figure 3.5 visualizes the homogeneous model for stereographic projections for the 1D case.

This homogeneous representation of a point is used as *point* in the conformal geometric algebra. This will be shown in the next section. Note that the stereographic projection from a plane leads to points on a sphere.

Therefore it is possible to use (special) rotations on this sphere to model e.g. translations in the originate space. Since also a homogeneous embedding is used, it is further possible to model projective geometry. The fusion of stereographic projections within geometric algebras is also presented in [132] and more details on the geometric properties are given.

### 3.3.2 Definition of the conformal geometric algebra

Following [110] a *Minkowski plane* is used to introduce CGA. Its vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the properties

$$\mathbf{e}_+^2 = 1, \quad \mathbf{e}_-^2 = -1, \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0. \tag{3.76}$$

In addition, a *null basis* can now be introduced by the vectors

$$\mathbf{e}_0 := \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} := \mathbf{e}_- + \mathbf{e}_+. \tag{3.77}$$

These vectors can be interpreted as the origin $\mathbf{e}_0$ of the coordinate system and the point at infinity $\mathbf{e}$ respectively. Furthermore, $\boldsymbol{E}$ is defined as $\boldsymbol{E} := \mathbf{e} \wedge \mathbf{e}_0 = \mathbf{e}_+ \wedge \mathbf{e}_-$.
For these elements the following straightforwardly proved properties can be summarized as

$$\begin{aligned} \mathbf{e}_0^2 = \mathbf{e}^2 = 0, \quad \mathbf{e} \cdot \mathbf{e}_0 = -1, \quad \boldsymbol{E} = \mathbf{e}_+\mathbf{e}_-, \\ \boldsymbol{E}^2 = 1, \quad \boldsymbol{E}\mathbf{e} = -\mathbf{e}, \quad \boldsymbol{E}\mathbf{e}_0 = \mathbf{e}_0, \\ \mathbf{e}_+\boldsymbol{E} = \mathbf{e}_-, \quad \mathbf{e}_-\boldsymbol{E} = \mathbf{e}_+, \quad \mathbf{e}_+\mathbf{e} = \boldsymbol{E} + 1, \\ \mathbf{e}_-\mathbf{e} = -(\boldsymbol{E} + 1), \quad \mathbf{e} \wedge \mathbf{e}_- = \boldsymbol{E}, \quad \mathbf{e}_+ \cdot \mathbf{e} = 1. \end{aligned} \tag{3.78}$$

The role of the Minkowski plane is to generate null vectors, and so to extend an Euclidean vector space $\mathbb{R}^n$ to $\mathbb{R}^{n+1,1} = \mathbb{R}^n \oplus \mathbb{R}^{1,1}$ and thus resulting in the conformal geometric algebra $\boldsymbol{\mathcal{G}}_{n+1,1}$. The conformal vector space derived from $\mathbb{R}^3$ is denoted as $\mathbb{R}^{4,1}$. A basis is given by $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_+, \mathbf{e}_-\}$. The corresponding algebra $\boldsymbol{\mathcal{G}}_{4,1}$ contains $2^5 = 32$ elements. The conformal unit pseudoscalar is denoted as

$$\boldsymbol{I}_C \quad = \quad \mathbf{e}_{+-123} = \boldsymbol{E}\boldsymbol{I}_E. \tag{3.79}$$

In this algebra points are considered as elements of the so-called null cone,

$$\{\underline{\boldsymbol{x}} \in \mathbb{R}^{4,1} | \underline{\boldsymbol{x}}^2 = 0, \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1\}. \tag{3.80}$$

The points of the null cone are related to those of the Euclidean space by

$$\underline{\boldsymbol{x}} \;=\; \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0. \tag{3.81}$$

Evaluating $\underline{\boldsymbol{x}}$ leads to

$$\begin{aligned}\underline{\boldsymbol{x}} &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0 \\ &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2(\mathbf{e}_+ + \mathbf{e}_-) + \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \\ &= \boldsymbol{x} + \left(\frac{1}{2}\boldsymbol{x}^2 - \frac{1}{2}\right)\mathbf{e}_+ + \left(\frac{1}{2}\boldsymbol{x}^2 + \frac{1}{2}\right)\mathbf{e}_-. \end{aligned} \tag{3.82}$$

This is exactly the homogeneous representation of a stereographically projected point onto the circle, given in equation (3.72). A point is only more compactly written using $\{\mathbf{e}, \mathbf{e}_0\}$, instead of $\{\mathbf{e}_+, \mathbf{e}_-\}$.

### 3.3.3 Geometric entities in conformal geometric algebra

The use of a certain geometric algebra induces a *basis geometric entity* from which the other entities are derived. In $\boldsymbol{\mathcal{G}}_3$, the algebra of the Euclidean space, the basis entities are points, whereby lines and planes are formulated as certain sets of points. In the motor algebra $\boldsymbol{\mathcal{G}}^+_{3,0,1}$, an algebra to model kinematics [14], the basis entities are lines, expressed in terms of the Plücker coordinates [21], and points and planes are written in these terms. This comes along with the fact, that the motor algebra is built from bivectors as basis entities and this is the reason, why it is in contrast to $\boldsymbol{\mathcal{G}}_3$ no universal Clifford algebra. In conformal geometric algebra $\boldsymbol{\mathcal{G}}_{4,1}$, spheres can be interpreted as the basis entities [126] from which the other entities are derived. It turns out that the above introduced point representation is nothing else than a degenerate sphere.

To introduce primitive geometric entities in CGA, firstly the representation of spheres in CGA is introduced. Then further entities will be discussed. A more detailed introduction can be found in [110]. A useful tool to visualize the entities and their geometric transformations in CGA can be found in [36].

There is no direct way to describe spheres as compact entities in $\boldsymbol{\mathcal{G}}_3$. The only possibility to define them is given by formulating a constraint equation. The equation for a point, $\boldsymbol{x} \in \boldsymbol{\mathcal{G}}_3$, on a sphere with center $\boldsymbol{p} \in \boldsymbol{\mathcal{G}}_3$ and radius $\rho \in \mathbb{R}$, $\rho \geq 0$, can be written as

$$\begin{aligned}(\boldsymbol{x} - \boldsymbol{p})^2 &= \rho^2 \\ \Leftrightarrow \boldsymbol{x}^2 - (\boldsymbol{x}\boldsymbol{p} + \boldsymbol{p}\boldsymbol{x}) + \boldsymbol{p}^2 &= \rho^2. \end{aligned} \tag{3.83}$$

The basis entities of the 3D conformal space are spheres $\underline{\boldsymbol{s}}$, containing the center $\boldsymbol{p}$ and the radius $\rho$, $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0$. The point $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0$ is nothing more than a degenerate sphere with radius $\rho = 0$, which can easily be seen from the representation of a sphere. In $\boldsymbol{\mathcal{G}}_{4,1}$ the equation (3.83) can therefore be represented more compactly:

$$\begin{aligned} (\boldsymbol{x} - \boldsymbol{p})^2 &= \rho^2 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} &= 0. \end{aligned} \tag{3.84}$$

This can easily be verified by evaluating

$$\begin{aligned} \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} &= (\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0) \cdot (\boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0) \\ &= -\frac{1}{2}(\boldsymbol{x}^2 + \boldsymbol{p}^2 - \rho^2) + \boldsymbol{x} \cdot \boldsymbol{p} \\ &= -\frac{1}{2}((\boldsymbol{x} - \boldsymbol{p})^2 - \rho^2). \end{aligned} \tag{3.85}$$

The dual form for a sphere is $\underline{\boldsymbol{s}}^\star$. The advantage of the dual form is that $\underline{\boldsymbol{s}}^\star$ can be calculated directly from points on the sphere: For four points on the sphere, $\underline{\boldsymbol{s}}^\star$ can be written as

$$\underline{\boldsymbol{s}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} \wedge \underline{\boldsymbol{d}}, \tag{3.86}$$

and a point $\underline{\boldsymbol{x}}$ is on a sphere $\underline{\boldsymbol{s}}$ iff $\underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star = 0$. Note: The incidence of a point with an entity can be expressed by the *inner product null-space* or *outer product null-space*, depending on the representation or dual representation of the entity. This follows from the easy relationship (see e.g. [77])

$$\begin{aligned} \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star &= 0. \end{aligned} \tag{3.87}$$

So far the description of the first two entities, points and spheres, is introduced.

Geometrically, a circle $\underline{\boldsymbol{z}}$ can be described by the intersection of two spheres. This means:

$$\underline{\boldsymbol{x}} \in \underline{\boldsymbol{z}} \;\Leftrightarrow\; \underline{\boldsymbol{x}} \in \underline{\boldsymbol{s}_1} \;\text{ and }\; \underline{\boldsymbol{x}} \in \underline{\boldsymbol{s}_2}. \tag{3.88}$$

Since $\underline{\boldsymbol{s}_1}$ and $\underline{\boldsymbol{s}_2}$ can be assumed as linearly independent, it is possible to write

$$\begin{aligned} \underline{\boldsymbol{x}} &\in \underline{\boldsymbol{z}} \\ \Leftrightarrow (\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}}_1)\underline{\boldsymbol{s}}_2 - (\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}}_2)\underline{\boldsymbol{s}}_1 &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underbrace{(\underline{\boldsymbol{s}}_1 \wedge \underline{\boldsymbol{s}}_2)}_{\underline{\boldsymbol{z}}} &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{z}} &= 0. \end{aligned} \tag{3.89}$$

This means, that algebraically a circle can be expressed as the outer product of two spheres. Figure 3.6[6] visualizes the generation of a circle as intersection

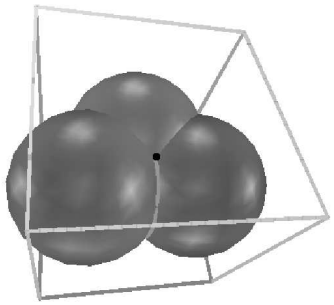


Fig. 3.6: A circle can be expressed as intersection of two spheres. Intersecting the circle with a third sphere leads to two points (only one of these two points is visible).

of two spheres. The intersection of the circle with a third sphere leads to a point pair. Indeed a point pair can be expressed as outer product of three spheres and is a three-blade entity.

In the dual form a circle is geometrically defined by three points on it,

$$\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}. \tag{3.90}$$

Evaluating the outer products of three points leads to

$$\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} = A + A^- \mathbf{e} + A^+ \mathbf{e}_0 + A^\pm \boldsymbol{E}, \tag{3.91}$$

with

[6] Figure 3.6 is taken from the visualization tool for Clifford algebra, CLUDraw [36].

| Entity | Representation | G. | Dual representation | G. |
|---|---|---|---|---|
| Sphere | $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0$ | 1 | $\underline{\boldsymbol{s}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} \wedge \underline{\boldsymbol{d}}$ | 4 |
| Point | $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0$ | 1 | $\underline{\boldsymbol{x}}^\star = (-\boldsymbol{E}\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0)\boldsymbol{I}_E$ | 4 |
| Plane | $\underline{\boldsymbol{P}} = \boldsymbol{n}\boldsymbol{I}_E - d\mathbf{e}$<br>$\boldsymbol{n} = (\boldsymbol{a} - \boldsymbol{b}) \wedge (\boldsymbol{a} - \boldsymbol{c})$<br>$d = (\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c})\boldsymbol{I}_E$ | 1 | $\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$ | 4 |
| Line | $\underline{\boldsymbol{L}} = \boldsymbol{r}\boldsymbol{I}_E + \mathbf{e}\boldsymbol{m}\boldsymbol{I}_E$<br>$\boldsymbol{r} = \boldsymbol{a} - \boldsymbol{b}$<br>$\boldsymbol{m} = \boldsymbol{a} \wedge \boldsymbol{b}$ | 2 | $\underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$ | 3 |
| Circle | $\underline{\boldsymbol{z}} = \underline{\boldsymbol{s}}_1 \wedge \underline{\boldsymbol{s}}_2$<br>$\underline{\boldsymbol{P}}_z = \underline{\boldsymbol{z}} \cdot \mathbf{e}$, $\underline{\boldsymbol{L}}_z^\star = \underline{\boldsymbol{z}} \wedge \mathbf{e}$<br>$\underline{\boldsymbol{p}}_z = \underline{\boldsymbol{P}}_z \vee \underline{\boldsymbol{L}}_z$, $\rho = \frac{\underline{\boldsymbol{z}}^2}{(\mathbf{e} \wedge \underline{\boldsymbol{z}})^2}$ | 2 | $\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$ | 3 |
| Point Pair | $\underline{\boldsymbol{P}\boldsymbol{P}} = \underline{\boldsymbol{s}}_1 \wedge \underline{\boldsymbol{s}}_2 \wedge \underline{\boldsymbol{s}}_3$ | 3 | $\underline{\boldsymbol{P}\boldsymbol{P}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$, $\underline{\boldsymbol{X}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{x}}$ | 2 |

Tab. 3.1: The entities and their dual representations in CGA

$$A = \boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c} \qquad A^- = \tfrac{1}{2}(\boldsymbol{c}^2(\boldsymbol{a} \wedge \boldsymbol{b}) - \boldsymbol{b}^2(\boldsymbol{a} \wedge \boldsymbol{c}) + \boldsymbol{a}^2(\boldsymbol{b} \wedge \boldsymbol{c}))$$
$$A^+ = \boldsymbol{a} \wedge \boldsymbol{b} + \boldsymbol{b} \wedge \boldsymbol{c} - \boldsymbol{a} \wedge \boldsymbol{c} \qquad A^\pm = \tfrac{1}{2}(\boldsymbol{a}(\boldsymbol{b}^2 - \boldsymbol{c}^2) + \boldsymbol{b}(\boldsymbol{c}^2 - \boldsymbol{a}^2) + \boldsymbol{c}(\boldsymbol{a}^2 - \boldsymbol{b}^2)).$$

The dual form of lines is represented by the outer product of two points on the line and the point at infinity (see [126]),

$$\underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}. \tag{3.92}$$

Since the outer product of three points determines a circle [110], the line can be interpreted as a circle passing through the point at infinity.

Evaluating the line $\underline{\boldsymbol{L}}^\star$ leads to

$$\begin{aligned} \underline{\boldsymbol{L}}^\star &= \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \\ &= (\mathbf{e} \wedge \boldsymbol{a} + \boldsymbol{E}) \wedge \underline{\boldsymbol{b}} \\ &= \mathbf{e} \wedge \boldsymbol{a} \wedge \boldsymbol{b} - \boldsymbol{a} \wedge \boldsymbol{E} + \boldsymbol{b} \wedge \boldsymbol{E} \\ &= \mathbf{e} \wedge \boldsymbol{a} \wedge \boldsymbol{b} + (\boldsymbol{b} - \boldsymbol{a})\boldsymbol{E} \\ &= \mathbf{e}\boldsymbol{m} + \boldsymbol{r}\boldsymbol{E}. \end{aligned} \tag{3.93}$$

From chapter 3.2 can be seen that again the line is given in its Plücker coordinates, containing the direction $\boldsymbol{r}$ and the moment $\boldsymbol{m}$.

Similar to lines, dual planes can then be defined by the outer product of three points on the plane and the point at infinity,

$$\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}. \tag{3.94}$$

A plane is a degenerate sphere, containing the point at infinity. Evaluating the plane $\underline{\boldsymbol{P}}^\star$ leads to

$$\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$$

$$\begin{aligned} &= (\mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}) \wedge \underline{\boldsymbol{c}} \\ &= \mathbf{e} \wedge \boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c} + \boldsymbol{E}(\boldsymbol{b} - \boldsymbol{a}) \wedge (\boldsymbol{c} - \boldsymbol{a}) \\ &= \mathbf{e}\boldsymbol{I}_E d + \boldsymbol{E}\boldsymbol{n}. \end{aligned} \tag{3.95}$$

Similar to lines it is easy to recognize, that the plane is given in its Hesse form of a plane, containing the normal $\boldsymbol{n}$ and the distance $d$.

An overview of the definitions of the entities, their dual representations and their grades is given in table 3.1. Since the outer product of 3 spheres leads to a point pair, it is a 2-blade in its dual space. Using the point at infinity leads to another representation of a pure point $\underline{\boldsymbol{X}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{x}}$ in the dual space and is called the affine representation of a point in [110].

The entities now have the following grades: points, spheres and planes are 1-blades, lines and circles are 2-blades and point pairs are 3-blades. Due to the fact that lines and planes are mostly generated by points on these entities, in the next sections the dual representations of points, lines and planes will be used.

### 3.3.4 Conformal transformations

In CGA, any conformal transformation can be expressed in the form

$$\sigma\underline{\boldsymbol{x}}' = G\underline{\boldsymbol{x}}G^{-1}, \tag{3.96}$$

where $G$ is a versor and $\sigma$ a scalar. Since the null cone is invariant under $G$, i.e. $(\underline{\boldsymbol{x}}')^2 = \underline{\boldsymbol{x}}^2 = 0$, it is convenient to apply a scale factor $\sigma$ to ensure $\underline{\boldsymbol{x}}' \cdot \mathbf{e} = \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1$. Table 3.2, taken from [110], summarizes the conformal transformations. The first column shows the type of operation performed with the versor product. The second column shows as example the result of a transformation acting on a point. The third column shows the versor which has to be applied and the last column shows the scaling parameter $\sigma$ which is (sometimes) needed, to result in a homogeneous point and ensure the scaling $\underline{\boldsymbol{x}}' \cdot \mathbf{e} = \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1$. As can be seen, any conformal transformation covers several more simple geometric transformations. In table 3.2, a reflection is expressed with respect to a hyper-plane with unit normal $\boldsymbol{n}$ and signed distance $\delta$. The inversion is expressed for a circle of radius $\rho$ centered at point $\boldsymbol{c}$. A transversion can be written down as an inversion followed by a translation and another inversion. The other transformations are self-explanatory. More explanations of the conformal group can also be found in [126, 64].

| Type | $G(\boldsymbol{x})$ on $\mathbb{R}^n$ | Versor in $\boldsymbol{\mathcal{G}}_{n+1,1}$ | $\sigma$ |
|---|---|---|---|
| Reflection | $-\boldsymbol{n}\boldsymbol{x}\boldsymbol{n} + 2\boldsymbol{n}\delta$ | $\boldsymbol{V} = \boldsymbol{n} + \mathbf{e}\delta$ | 1 |
| Inversion | $\frac{\rho^2}{\boldsymbol{x}-\boldsymbol{c}} + \boldsymbol{c}$ | $\boldsymbol{V} = \boldsymbol{c} - \frac{1}{2}\rho^2\mathbf{e}$ | $\left(\frac{\boldsymbol{x}-\boldsymbol{c}}{\rho}\right)^2$ |
| Rotation | $\boldsymbol{R}\boldsymbol{x}\boldsymbol{R}^{-1}$ | $\boldsymbol{R} = \exp\left(-\frac{\theta}{2}\boldsymbol{n}\right)$ | 1 |
| Translation | $\boldsymbol{x} - \boldsymbol{t}$ | $\boldsymbol{T}_t = 1 + \frac{1}{2}\boldsymbol{t}\mathbf{e}$ | 1 |
| Transversion | $\frac{\boldsymbol{x}-\boldsymbol{x}^2\boldsymbol{t}}{\sigma(\boldsymbol{x})}$ | $\boldsymbol{K}_t = 1 + \boldsymbol{t}\mathbf{e}_0$ | $1 - 2\boldsymbol{t}\cdot\boldsymbol{x} + \boldsymbol{x}^2\boldsymbol{t}^2$ |
| Dilation | $\lambda\boldsymbol{x}$ | $\boldsymbol{D}_\lambda = \exp(-\frac{1}{2}\boldsymbol{E}(\ln\lambda))$ | $\lambda^{-1}$ |
| Involution | $\boldsymbol{x}^\star = -\boldsymbol{x}$ | $\boldsymbol{E}$ | $-1$ |

Tab. 3.2: Table of conformal transformations, versors and scaling parameters.

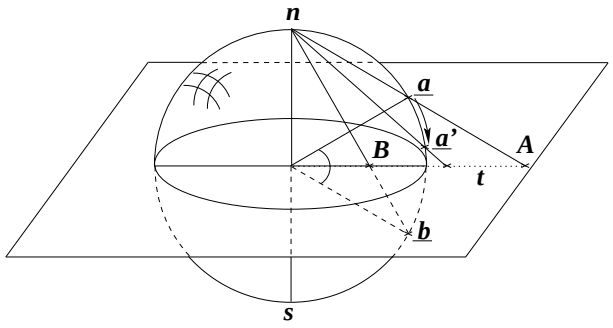



Fig. 3.7: Visualization of an inversion and translation for a stereographically projected point in the 2D case.

It is shown in e.g. [78], that in $\boldsymbol{\mathcal{G}}_3$ the transformations are generated by reflections as basic operation. The question is what a reflection does mean for the stereographically projected point on the sphere. This is visualized in figure 3.7 for the 2D case: A reflection of a point $\underline{\boldsymbol{a}}$ on the sphere with respect to a 2D (base) plane leads to a new point $\underline{\boldsymbol{b}}$ on the sphere, which corresponds to the inverse $\boldsymbol{B}$ of the point $\boldsymbol{A}$ on the 2D plane. This means, that the basic operation in $\boldsymbol{\mathcal{G}}_{4,1}$ is an inversion, and the other operations are derived from it. In figure 3.7 it is also shown, what a translation $\boldsymbol{t}$ of a point $\boldsymbol{A}$ on the 2D plane means for a corresponding point $\underline{\boldsymbol{a}}$ on the sphere. A translation $\boldsymbol{t}$ corresponds to a special rotation $\underline{\boldsymbol{a}} \rightarrow \underline{\boldsymbol{a}}'$. It is also easy to imagine that a rotation of a point in the 2D plane is exactly the same for its stereographically projected point on the sphere. This means, that a rotation can be calculated in the same manner as in $\boldsymbol{\mathcal{G}}_2$ or $\boldsymbol{\mathcal{G}}_3$ and a translation is a special rotation in $\boldsymbol{\mathcal{G}}_{3,1}$ or $\boldsymbol{\mathcal{G}}_{4,1}$, respectively. This is the reason why kinematics can be described in this model in a linear manner.

Since the main topic of this thesis concentrates on rigid motions, it will

now be proceeded with expressing rotations and translations in CGA.

### 3.3.5 Rigid motions in CGA

This section deals with the formulation of *rigid body motions*. As mentioned previously, a rigid body motion corresponds to the Euclidean transformation group $SE(3)$. Although being a transformation by itself, it subsumes rotation and translation. This makes it necessary to represent the Euclidean transformation as a linear one, with a multiplicative coupling of rotation and translation and to have access to both as linear operations. Since the conformal transformation contains the Euclidean transformation, it is possible to use the conformal group to express rigid body motions. Note that although the conformal group is more general than the Euclidean group, for the pose estimation scenario it is sufficient to concentrate only on this subset of transformations.

So far, rotors as elements of $\boldsymbol{\mathcal{G}}_3^+$ can be used to formalize a pure rotation, but indeed it is not possible to describe general rigid body motions in this algebra in a multiplicative manner. As well as in $\boldsymbol{\mathcal{G}}_3$ (see section 3.1.2), a rotation in $\boldsymbol{\mathcal{G}}_{4,1}$ is represented by a rotor,

$$\boldsymbol{R} \;=\; \exp\left(-\frac{\theta}{2}\boldsymbol{l}\right). \tag{3.97}$$

The components of the rotor $\boldsymbol{R} \in \boldsymbol{\mathcal{G}}_{4,1}^+$ are, similar to section 3.1.2, the unit bivector $\boldsymbol{l} \in \langle \boldsymbol{\mathcal{G}}_3 \rangle_2 \subseteq \boldsymbol{\mathcal{G}}_{4,1}$ which represents the rotation plane, and the angle $\theta$, which represents the amount of the rotation. The rotation of an entity can be performed just by multiplying the entity from the left with the rotor $\boldsymbol{R}$ and from the right with its reverse $\widetilde{\boldsymbol{R}}$. For example, a rotation of a point can be written as

$$\underline{\boldsymbol{x}}' \;=\; \boldsymbol{R}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{R}}. \tag{3.98}$$

That a rotation of a point in $\boldsymbol{\mathcal{G}}_{4,1}$ can be expressed by the same rotor as in $\boldsymbol{\mathcal{G}}_3$ has already been clarified in section 3.3.4. That this also holds for any blade (and thus for lines, planes, circles, spheres, etc.) can be seen from the easy relation

$$\boldsymbol{R}(\underline{\boldsymbol{x}}_1 \wedge \underline{\boldsymbol{x}}_2 \wedge \ldots \wedge \underline{\boldsymbol{x}}_n)\widetilde{\boldsymbol{R}} \;=\; (\boldsymbol{R}\underline{\boldsymbol{x}}_1\widetilde{\boldsymbol{R}}) \wedge (\boldsymbol{R}\underline{\boldsymbol{x}}_2\widetilde{\boldsymbol{R}}) \wedge \ldots \wedge (\boldsymbol{R}\underline{\boldsymbol{x}}_n\widetilde{\boldsymbol{R}}). \tag{3.99}$$

To translate an entity with respect to a translation vector $\boldsymbol{t} \in \langle \boldsymbol{\mathcal{G}}_3 \rangle_1$, it is possible to use a so called *translator*, $\boldsymbol{T} \in \boldsymbol{\mathcal{G}}_{4,1}$,

$$\boldsymbol{T} \;=\; (1 + \frac{\mathbf{e}\boldsymbol{t}}{2}) = \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right). \tag{3.100}$$

A translator is a special rotor acting at infinity by using the null vector $\mathbf{e}$. Similar to a rotation, an entity can be translated by multiplying the entity from the left with the translator $\boldsymbol{T}$ and its reverse $\widetilde{\boldsymbol{T}}$ from the right,

$$\underline{\boldsymbol{x}}' \;=\; \boldsymbol{T}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{T}}. \tag{3.101}$$

To express a rigid body motion, the consecutive application of a rotor and translator can be written as their product. Such an operator is denoted as $\boldsymbol{M}$,

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}. \tag{3.102}$$

It is a special even grade multivector, called a motor, which is an abbreviation of *moment and vector*[13, 141]. The rigid body motion of e.g. a point $\underline{\boldsymbol{X}}$ can be written as

$$\underline{\boldsymbol{X}}' \;=\; \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}, \tag{3.103}$$

see also [79].

This formalization of a rigid displacement can not only be applied to points or lines (see [141]), but to all entities contained in table 3.1. Furthermore, the transformation rule is the same for all entities of table 3.1. This is in contrast to a former definition of motors in the frame of motor algebra [13, 14], the algebra $\boldsymbol{\mathcal{G}}^+_{3,0,1}$, which is formulating kinematics in a space composed of lines and which is isomorphic to the dual quaternion algebra. Although equation (3.102) is a valid definition of a motor in both the motor algebra and CGA, its behavior with respect to different entities is quite different. Compared with the motor algebra, in CGA there is no need to make any sign changes depending on the entity the motor is acting on. This makes several case decisions in the previous formalizations of kinematics unnecessary and thus the calculations will become easier. The reason for this increased symmetry of a motor action lies in the chosen algebraic embedding.

### 3.3.6 Twists and screw transformations

Now follows a further definition of a motor in CGA based on the so-called *twists*. Every rigid body motion can be expressed as a twist or screw motion

[124], which is a rotation around a line in space (in general not passing through the origin)[7] combined with a translation along this line. In CGA it is possible to use the rotors and translators to express screw motions in space. I will start with the formalization of general rotations and then continue with screw motions. It will turn out that a general rotation is a special case of a screw motion and its generator is directly connected to the representation of a 3D line.

To model a rotation of a point $\underline{\boldsymbol{X}}$ around an arbitrary line $\underline{\boldsymbol{L}}$ in the space, the general idea is to translate the point $\underline{\boldsymbol{X}}$ with the distance vector between the line $\underline{\boldsymbol{L}}$ and the origin, to perform a rotation and to translate the transformed point back. So a motor $\boldsymbol{M} \in \boldsymbol{\mathcal{G}}^{+}_{4,1}$ describing a general rotation has the form

$$\boldsymbol{M} \;=\; \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}}, \tag{3.104}$$

denoting the inverse translation, rotation and back translation, respectively. Using the exponential form of the translator and rotor leads to[8]

$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}} \\
&= \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\exp\left(-\frac{\theta}{2}\boldsymbol{l}\right)\exp\left(-\frac{\mathbf{e}\boldsymbol{t}}{2}\right) \\
&= \left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\exp\left(-\frac{\theta}{2}\boldsymbol{l}\right)\left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right) \\
&= \exp\left(\left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\left(-\frac{\theta}{2}\boldsymbol{l}\right)\left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\right) \\
&= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right).
\end{aligned} \tag{3.105}$$

This formulation corresponds to the one for a general rotation given in [110]. Merely an exponential representation of the motor is used since then it is more easy to calculate its derivative.

It is interesting to mention that the exponential part of the motor $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}}$ consists directly of the line components to rotate the entities around. To show this property, firstly the description of a dual line $\underline{\boldsymbol{L}}^{\star}$ is recalled,

$$\begin{aligned}
\underline{\boldsymbol{L}}^{\star} &= \mathbf{e}\wedge\underline{\boldsymbol{a}}\wedge\underline{\boldsymbol{b}} \\
&= \mathbf{e}(\boldsymbol{a}\wedge\boldsymbol{b})+(\boldsymbol{b}-\boldsymbol{a})\boldsymbol{E}.
\end{aligned} \tag{3.106}$$

[7] Such an operation is also called a *general rotation*.

[8] In the fourth equation is made use of the property $\boldsymbol{g}\ \exp(\xi)\ \widetilde{\boldsymbol{g}} = \exp(\boldsymbol{g}\xi\widetilde{\boldsymbol{g}})$ for $\boldsymbol{g}\widetilde{\boldsymbol{g}} = 1$. This property can be proven by an induction on the series expression of the exponential function.

Using the unit direction vector $\boldsymbol{n}$ and the plumb point $\boldsymbol{t}$ of the origin to the line leads to the line representation

$$\underline{\boldsymbol{L}}^{\star} = \mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n}) + \boldsymbol{n}\boldsymbol{E}. \tag{3.107}$$

Multiplying the dual line $\underline{\boldsymbol{L}}^{\star}$ with $\boldsymbol{I}_C$ (from the right) results in

$$\begin{aligned}(\mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n}) + \boldsymbol{n}\boldsymbol{E})\boldsymbol{I}_C &= (\mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n})\boldsymbol{E}\boldsymbol{I}_E + \boldsymbol{n}\boldsymbol{E})\boldsymbol{E}\boldsymbol{I}_E \\ &= \mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n})\boldsymbol{I}_E + \boldsymbol{n}\boldsymbol{I}_E \\ &= \mathbf{e}(\boldsymbol{t} \cdot (\boldsymbol{n}\boldsymbol{I}_E)) + \boldsymbol{n}\boldsymbol{I}_E \\ &= \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l}) + \boldsymbol{l}, \end{aligned} \tag{3.108}$$

since in 3D the direction $\boldsymbol{n}$ of the line corresponds to the dual of the rotation plane $\boldsymbol{l}$, $\boldsymbol{n} = \boldsymbol{l}^{\star}$.

Vice versa: Given the dual line $\underline{\boldsymbol{L}}^{\star}$ (with unit direction) in the space, the corresponding motor describing a general rotation around this line is given by

$$\begin{aligned}\boldsymbol{M} &= \exp\left(-\frac{\theta}{2}\underline{\boldsymbol{L}}^{\star}\boldsymbol{I}_C\right) \\ &= \exp\left(-\frac{\theta}{2}\underline{\boldsymbol{L}}\right). \end{aligned} \tag{3.109}$$

Note that the line $\underline{\boldsymbol{L}}$ must be scaled with respect to the direction, $\|\boldsymbol{n}\| = 1$, since the scaling of the line is directly connected to the amount of the rotation $\theta$. This shows that a line is a generator of a general rotation. Now will be continued with screw motions.

Screw motions can be used to describe rigid body motions. Already as early as 1830 Chasles proved that every rigid body motion can be realized by a rotation around an axis combined with a translation parallel to that axis, see also [141, 124]. This is called a *screw motion*. The infinitesimal version of a screw motion is called a twist and it provides a description of the instantaneous velocity of a rigid body in terms of its linear and angular components. A screw motion is defined by an axis $\boldsymbol{l}$, a pitch $h$ and a magnitude $\theta$. The *pitch* of the screw is the ratio of translation to rotation, $h := \frac{d}{\theta}$ ($d, \theta \in \mathbb{R}$, $\theta \neq 0$). If $h \to \infty$, then the corresponding screw motion consists of a pure translation along the axis of the screw. The principle of a screw motion is visualized in figure 3.8. To model a screw motion, the entity has to be translated during a general rotation with respect to the rotation axis.

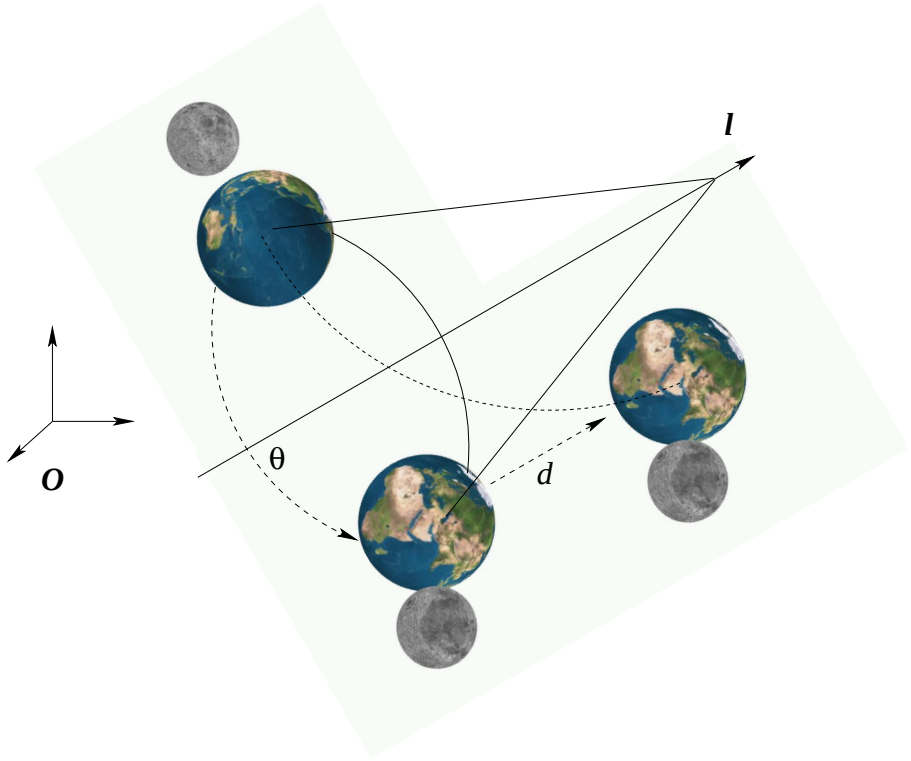



Fig. 3.8: Visualization of a screw motion along $\boldsymbol{l}$.

The resulting motor can be calculated in the following way,

$$
\begin{aligned}
\boldsymbol{M} &= \boldsymbol{T}_{d\boldsymbol{n}}\boldsymbol{T}\boldsymbol{R}\tilde{\boldsymbol{T}} \\
&= \exp\left(\frac{\mathbf{e}d\boldsymbol{n}}{2}\right)\exp\left(-\frac{\theta}{2}\left(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right) \\
&= \exp\left(\frac{\mathbf{e}d\boldsymbol{n}}{2}-\frac{\theta}{2}\left(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right) \\
&= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l}+\mathbf{e}(\underbrace{\boldsymbol{t}\cdot\boldsymbol{l}-\frac{d}{\theta}\boldsymbol{n}}_{\boldsymbol{m}})\right)\right)
\end{aligned}
$$

$$= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}\boldsymbol{m}\right)\right). \tag{3.110}$$

The bivector in the exponential part, $-\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}\boldsymbol{m})$, is a twist (see also chapter A.2). The vector $\boldsymbol{m}$ is a vector in $\mathbb{R}^3$ which can be decomposed in an orthogonal and parallel part with respect to $\boldsymbol{n} = \boldsymbol{l}^\star$. If $\boldsymbol{m}$ is zero, the motor $\boldsymbol{M}$ gives a pure rotation and if $\boldsymbol{l}$ is zero, the motor gives a pure translation. For $\boldsymbol{m} \perp \boldsymbol{l}^\star$, the motor gives a general rotation and for $\boldsymbol{m} \not\perp \boldsymbol{l}^\star$, the motor gives a screw motion.

For the pose estimation algorithm, I prefer the interpretation of a motor $\boldsymbol{M}$ as exponential of a twist, since the exponential form enables me to linearize the rigid motion in the later introduced constraint equations more easily.

## Chapter 4

# STRATIFICATION OF THE POSE ESTIMATION PROBLEM

Figure 4.1 recalls the 2D-3D pose estimation problem for points, lines and planes: Assumed is the knowledge of a 3D object model and its observation

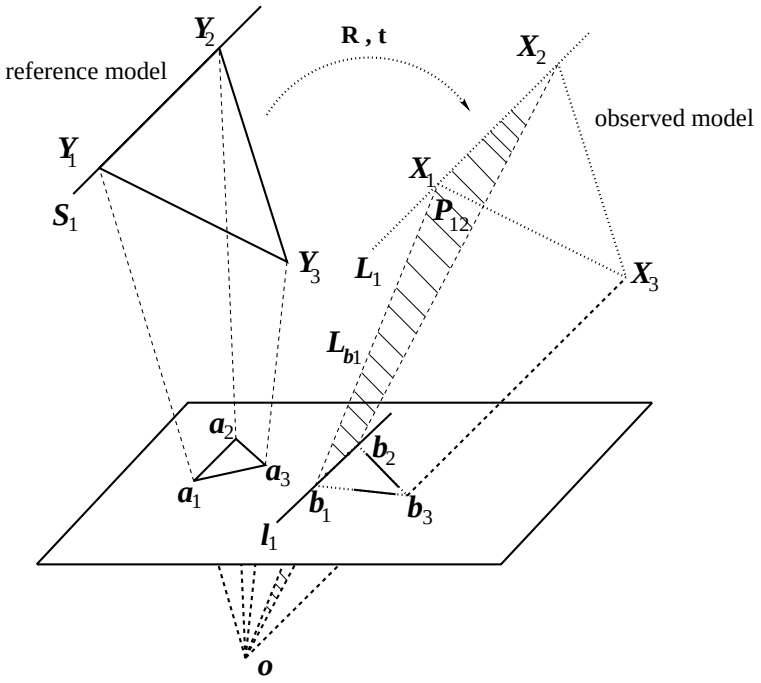



Fig. 4.1: The scenario. The solid lines describe the assumptions: the camera model (visualized by the image plane and the optical center $\boldsymbol{o}$), the model of the object (consisting of points and lines $\boldsymbol{Y}_i$ and $\boldsymbol{S}_i$) and corresponding extracted entities on the image plane (points $\boldsymbol{b}_i$ and lines $\boldsymbol{l}_i$). The dashed lines describe the pose of the model, which leads to the best fit of the object with the actually extracted entities.

in an image of a calibrated camera. The aim is to find the rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ of the object, which lead to the best fit of the reference model with the actually extracted entities.

## 4.1 Relating entities in Euclidean, projective and conformal geometry

So far it is possible to use CGA for the formalization of involved entities and their rigid motions. To formalize the scenario of figure 4.1 in a suitable way, now it will be examined how to describe the interaction of projective and conformal geometry. As mentioned earlier, the interaction of the different strata of the stratification hierarchy [48] has only been poorly lit during the last years. E.g. Ruf [157] deals with this problem, but only for point features in the framework of matrix calculus. Instead, I want to extend the problem to more general object features and use the conformal geometric algebra in this context. To reach interaction between the strata, algebras for the projective and Euclidean space are interpreted as subalgebras of the CGA. It turns out that it is possible to switch representations of entities between these algebras by multiplicative operators.

The main strategy to estimate the pose of the rigid object in figure 2.1 is very simple. It is summarized in figure 4.2 for the case of points: Compute the projection rays as projective reconstruction of the image points and compare them (in the Euclidean space) with the object model points after the movement. But in detail several algebraic transformations have to be performed: Firstly, the image entities are projectively reconstructed and converted to a conformal representation. Then the model features are transformed into the conformal space. To get a distance measure in the Euclidean space, the transformed model entities and reconstructed image features are compared by suitable scaled constraint equations in the last step.

The involved mathematical spaces and their corresponding geometric algebras can be summarized in the following manner: The Euclidean framework can be represented by using the algebra $\boldsymbol{\mathcal{G}}_{3,0}$ (chapter 3.1), and $\boldsymbol{\mathcal{G}}_{3,1}$ (chapter 3.2) can be used to represent the projective space [81]. The projective plane is represented by the algebra $\boldsymbol{\mathcal{G}}_{2,1}$. One way of defining a kinematic space is given by the motor algebra $\boldsymbol{\mathcal{G}}^{+}_{3,0,1}$ [13, 14]. Another way is given by embedding the kinematics into the 3D conformal space represented by $\boldsymbol{\mathcal{G}}_{4,1}$ (chapter 3.3) [110]. Table 4.1 gives an overview of representations of points

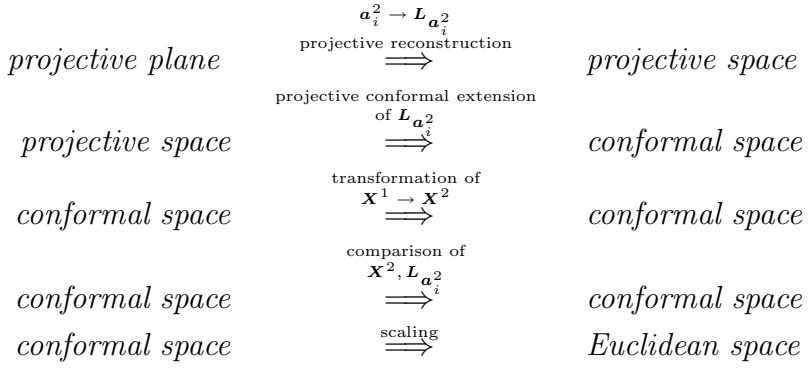



Fig. 4.2: Involved geometric spaces of the 2D-3D pose estimation problem.

| Space | Algebra | Point Representation |
|---|---|---|
| 3D Euclidean | $\boldsymbol{\mathcal{G}}_{3,0}$ | $\boldsymbol{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$ |
| 2D projective | $\boldsymbol{\mathcal{G}}_{2,1}$ | $\boldsymbol{x_{p_2}} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \mathbf{e}_-$ |
| 3D projective | $\boldsymbol{\mathcal{G}}_{3,1}$ | $\boldsymbol{X_{p_3}} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 + \mathbf{e}_-$ |
| 3D kinematic | $\boldsymbol{\mathcal{G}}^+_{3,0,1}$ | $\boldsymbol{X} = 1 + \boldsymbol{I}(x_1\mathbf{e}_{23} + x_2\mathbf{e}_{31} + x_3\mathbf{e}_{12})$ |
| 3D conformal | $\boldsymbol{\mathcal{G}}_{4,1}$ | $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0$ |
| | | $\underline{\boldsymbol{X}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{x}}$ |

Tab. 4.1: Different mathematical spaces with their corresponding geometric algebras and point representations.

using different algebras. As can be seen, the relation

$$\boldsymbol{\mathcal{G}}_{4,1} \supseteq \boldsymbol{\mathcal{G}}_{3,1} \supseteq \boldsymbol{\mathcal{G}}_{3,0} \tag{4.1}$$

is valid, but only for $\boldsymbol{\mathcal{G}}_{3,0}$ limited to points. Both algebras for the projective and Euclidean space constitute subspaces of the linear space of the conformal geometric algebra. Since only points are modeled in $\boldsymbol{\mathcal{G}}_{3,0}$ the *direction* of modeling the pose problem is consistent with the increasing possibilities through using higher geometric algebras: Reconstruct from the projective plane one dimensional higher entities and work in the projective or conformal space respectively. These spaces provide more possibilities for expressing geometry. Thus the modeling of the pose problem follows the direction

$$\boldsymbol{\mathcal{G}}_{3,0}, \boldsymbol{\mathcal{G}}_{2,1} \Rightarrow \boldsymbol{\mathcal{G}}_{3,1} \Rightarrow \boldsymbol{\mathcal{G}}_{4,1}. \tag{4.2}$$

In the following operators will be introduced which not only relate linear spaces of the considered algebras but guarantee the mapping between the

algebraic properties. This means that operators are defined which transform the representation of the entities of the conformal space into equivalent entities in the projective space, and vice versa. The possibility to change the representation of an entity allows one to pick up the advantages of each algebra, and thus to use the best suited algebra for each subproblem.

### 4.1.1 Change of representations of geometric entities

In this section it will be shown, how these different representations can interact. The operators between the conformal and projective space will be denoted as *conformal projective split* and *projective conformal extension*, according to the *projective split* [81] which allows for a change between the projective and the Euclidean space and the *conformal split* [79, 81] which allows for a change between the Euclidean and conformal space. By using these different splits and extensions, it is possible to describe the whole stratification hierarchy. This will lead to a compact formulation of the 2D-3D pose estimation problem.

Now the interaction of the first two spaces, the conformal space to describe kinematics and the projective space to describe the pinhole camera system will be considered. The two operators which are used to switch entities between the geometric algebras representing these spaces are summarized in the following theorems:

**Theorem 4.1** *To change an entity $\Theta$ given in the projective representation, $\Theta_p$, to the conformal representation, $\Theta_c$, $\Theta_p \in \{\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{P} \in \boldsymbol{\mathcal{G}}_{3,1}\} \rightarrow \Theta_c \in \{\underline{\boldsymbol{X}}^\star, \underline{\boldsymbol{L}}^\star, \underline{\boldsymbol{P}}^\star \in \boldsymbol{\mathcal{G}}_{4,1}\}$, the following operator has to be applied:*

$$\Theta_c \;=\; \mathbf{e} \wedge \Theta_p. \tag{4.3}$$

Note, that circles and spheres are entities which can not be modeled by nullspaces in the projective geometric algebra. Therefore it is not possible to switch them between the projective and conformal geometric algebra. This is in consistency with the *direction* of modeling the pose problem: Reconstruct image features to one-dimensional higher entities and make a comparison in the 3D space.

To prove the theorem 4.1 it is sufficient to show the easy relation $\mathbf{e} \wedge \mathbf{e}_- = \boldsymbol{E}$,

$$\begin{aligned} \mathbf{e} \wedge \mathbf{e}_- &= (\mathbf{e}_- + \mathbf{e}_+) \wedge \mathbf{e}_- \\ &= \mathbf{e}_+ \wedge \mathbf{e}_- = \boldsymbol{E}. \end{aligned} \tag{4.4}$$

To clarify the involved geometry, the representation changes of points, lines and planes will be computed explicitly:

$$\begin{aligned} \boldsymbol{X} \in \boldsymbol{\mathcal{G}}_{3,1} &= \boldsymbol{x} + \mathbf{e}_- \\ &\rightarrow \mathbf{e} \wedge (\boldsymbol{x} + \mathbf{e}_-) \\ &= \mathbf{e} \wedge \boldsymbol{x} + \mathbf{e} \wedge \mathbf{e}_- \\ &= \mathbf{e}\boldsymbol{x} + \boldsymbol{E} = \underline{\boldsymbol{X}}^\star \in \boldsymbol{\mathcal{G}}_{4,1}. \end{aligned} \tag{4.5}$$

$$\begin{aligned} \boldsymbol{L} \in \boldsymbol{\mathcal{G}}_{3,1} &= \mathbf{e}_- \boldsymbol{r} + \boldsymbol{m} \\ &\rightarrow \mathbf{e} \wedge (\mathbf{e}_- \boldsymbol{r} + \boldsymbol{m}) \\ &= \mathbf{e}\boldsymbol{m} + \mathbf{e} \wedge (\mathbf{e}_- \boldsymbol{r}) \\ &= \boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m} = \underline{\boldsymbol{L}}^\star \in \boldsymbol{\mathcal{G}}_{4,1}. \end{aligned} \tag{4.6}$$

$$\begin{aligned} \boldsymbol{P} \in \boldsymbol{\mathcal{G}}_{3,1} &= \mathbf{e}_- \boldsymbol{n} + d\boldsymbol{I}_E \\ &\rightarrow \mathbf{e} \wedge (\mathbf{e}_- \boldsymbol{n} + d\boldsymbol{I}_E) \\ &= \boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E = \underline{\boldsymbol{P}}^\star \in \boldsymbol{\mathcal{G}}_{4,1}. \end{aligned} \tag{4.7}$$

Now I will continue with the operator to switch representations from the conformal space into the projective space.

**Theorem 4.2** *To change an entity $\Theta$, given in the conformal representation, $\Theta_c$, to the projective representation, $\Theta_p$, $\Theta_c \in \{\underline{\boldsymbol{X}}^\star, \underline{\boldsymbol{L}}^\star, \underline{\boldsymbol{P}}^\star \in \boldsymbol{\mathcal{G}}_{4,1}\} \rightarrow \Theta_p \in \{\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{P} \in \boldsymbol{\mathcal{G}}_{3,1}\}$, the following operator has to be applied:*

$$\Theta_p = \mathbf{e}_+ \cdot \Theta_c. \tag{4.8}$$

To prove theorem 4.2 it is sufficient to show the following identity
$\Theta_p = \mathbf{e}_+ \cdot (\mathbf{e} \wedge \Theta_p)$,

$$\mathbf{e}_+ \cdot (\mathbf{e} \wedge \Theta_p) = \underbrace{(\mathbf{e}_+ \cdot \mathbf{e})}_{1} \wedge \Theta_p - \mathbf{e} \wedge \underbrace{(\mathbf{e}_+ \cdot \Theta_p)}_{0} = \Theta_p. \tag{4.9}$$

I call the operations $\mathbf{e}\wedge$ and $\mathbf{e}_+\cdot$ the *projective conformal extension* and *conformal projective split* respectively.

The interaction between the algebras for the projective and Euclidean space is much simpler. Lines and planes can be represented in the Euclidean space, but as mentioned before, this is only an artificially generated representation and not generated by the algebra itself. As a consequence the

transformation can be described in a suitable way for points only. The transformations are based on Hestenes' formalization in [81] and can be expressed in the following way,

$$\begin{array}{rcccl} \boldsymbol{X} & \rightarrow & \frac{(\boldsymbol{X}\wedge\mathbf{e}_-)\cdot\mathbf{e}_-}{\boldsymbol{X}\cdot\mathbf{e}_-} & = & \boldsymbol{x} \quad \in \boldsymbol{\mathcal{G}}_{3,0} \\ \boldsymbol{x} & \rightarrow & \boldsymbol{x}+\mathbf{e}_- & = & \boldsymbol{X} \quad \in \boldsymbol{\mathcal{G}}_{3,1}. \end{array}$$

Table 4.2 gives an overview of the three mainly involved spaces and their interaction.

| Euclidean space | | projective space | | conformal space |
|---|---|---|---|---|
| $\boldsymbol{\mathcal{G}}_{3,0}$ | $\subseteq$ | $\boldsymbol{\mathcal{G}}_{3,1}$ | $\subseteq$ | $\boldsymbol{\mathcal{G}}_{4,1}$ |
| $\Theta_e$ | $\begin{array}{c}\boldsymbol{x}+\mathbf{e}_- \\ \longrightarrow \\ -\frac{\boldsymbol{X}}{\boldsymbol{X}\cdot\mathbf{e}_-} \\ \longleftarrow\end{array}$ | $\Theta_p$ | $\begin{array}{c}\mathbf{e}\wedge\Theta_p \\ \longrightarrow \\ \mathbf{e}_+\cdot\Theta_c \\ \longleftarrow\end{array}$ | $\Theta_c$ |
| $\Theta_e$ | $\begin{array}{c}\longrightarrow \\ \longleftarrow\end{array}$ | $\begin{array}{c}\mathbf{e}\wedge(\boldsymbol{x}+\mathbf{e}_-) \\ \frac{((\mathbf{e}_+\cdot\underline{\boldsymbol{X}})\wedge\mathbf{e}_-)\cdot\mathbf{e}_-}{(\mathbf{e}_+\cdot\underline{\boldsymbol{X}})\cdot\mathbf{e}_-}\end{array}$ | $\begin{array}{c}\longrightarrow \\ \longleftarrow\end{array}$ | $\Theta_c$ |

Tab. 4.2: Interaction between algebras of the Euclidean, projective and conformal space.

To describe a rigid body motion of an entity given in projective geometry, it is possible to change its representation in a conformal one, compute the rigid body motion and return to the projective space: Let $\Theta_p$ be an entity given in the projective space, and $\boldsymbol{t}$ a translation vector in Euclidean space. In conformal geometric algebra, the translator has the following structure,

$$\boldsymbol{T} = \left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right) \tag{4.10}$$

$$\text{and} \quad \widetilde{\boldsymbol{T}} = \left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right). \tag{4.11}$$

Then a multiplicative formulation of the translated entity in the projective

space is given by

$$\Theta'_p = \underbrace{\mathbf{e}_+ \cdot \underbrace{(\boldsymbol{T}(\mathbf{e} \wedge \underbrace{\Theta_p}_{\text{projective}})\widetilde{\boldsymbol{T}})}_{\text{conformal}}}_{\text{projective}}. \tag{4.12}$$

To compute joins and meets of entities given in conformal geometric algebra, I change their representations to the projective space, perform the incidence operation and return to the conformal space. For example, the intersection (denoted with the operator $\vee_c$) of a line $\underline{\boldsymbol{L}}^\star$ with a plane $\underline{\boldsymbol{P}}^\star$ is given by

$$\underline{\boldsymbol{L}}^\star \vee_c \underline{\boldsymbol{P}}^\star = \underbrace{\mathbf{e} \wedge \underbrace{((\mathbf{e}_+ \cdot \underbrace{\underline{\boldsymbol{L}}^\star}_{\text{conformal}}) \vee (\mathbf{e}_+ \cdot \underbrace{\underline{\boldsymbol{P}}^\star}_{\text{conformal}}))}_{\text{projective}}}_{\text{conformal}}. \tag{4.13}$$

To explicitly compute the Euclidean intersection point of two lines, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, given in the projective space, the lines are intersected in the projective space and then the projective split is applied to get the intersection point in the geometric algebra of the Euclidean space,

$$\boldsymbol{x} = \underbrace{\frac{1}{\underbrace{\boldsymbol{L}_1 \vee \boldsymbol{L}_2}_{projective} \cdot \mathbf{e}_-} \left( ((\underbrace{\boldsymbol{L}_1 \vee \boldsymbol{L}_2}_{projective}) \wedge \mathbf{e}_-) \cdot \mathbf{e}_- \right)}_{Euclidean}. \tag{4.14}$$

These examples show the possibility of interaction between the Euclidean, projective and conformal framework.

### 4.1.2 Pose constraints in conformal geometric algebra

This section gives a brief preview how the interaction of entities in geometric algebras will be applied on the pose problem. As mentioned earlier, the main problem in the pose scenario is how to compare 2D image features with 3D Euclidean object features. The constraint equations will lead to equations of the following structure (here just for point correspondences),

$$\lambda((\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \cdot \mathbf{e}_+ = 0. \tag{4.15}$$

The interpretation of the equation is simple as the equation can be separated in the following manner,

$$\lambda((\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\substack{\text{object point}\\ \text{in conformal space}}}\ \widetilde{\boldsymbol{M}})\ \underline{\times}\ \mathbf{e}\wedge(\ \underbrace{\boldsymbol{O}}_{\substack{\text{optical}\\ \text{center}}}\ \wedge\ \underbrace{\boldsymbol{x}}_{\substack{\text{image}\\ \text{point}}}\ )\ )\cdot\mathbf{e}_+\ =\ 0. \tag{4.16}$$

rigid motion of the object point

projection ray, reconstructed from the image point in conformal space

collinearity of the transformed object point with the reconstructed line

Euclidean distance measure between 3D line and 3D point

As can be recognized, the strategy of expressing the pose problem can directly be seen from the equation, all geometric aspects are considered and the equation is compact and easy to interpret.

The mathematical spaces involved here are

$$\lambda\,((\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{CS}\ \widetilde{\boldsymbol{M}})\ \underline{\times}\ \mathbf{e}\wedge(\underbrace{\underbrace{\boldsymbol{O}}_{PS}\wedge\underbrace{\boldsymbol{x}}_{PP}}_{PS}))\cdot\mathbf{e}_+\ =\ 0. \tag{4.17}$$

$CS$ (over $\boldsymbol{M}\,\underline{\boldsymbol{X}}\,\widetilde{\boldsymbol{M}}$)

$CS$ (over $\boldsymbol{O}\wedge\boldsymbol{x}$)

$CS$ (over the whole left factor)

$ES$ (over the whole expression)

Here $PP$ abbreviates *projective plane*, $PS$ *projective space*, $CS$ *conformal space* and $ES$ the *Euclidean space*. Furthermore it will be shown in the next section, that the commutator product ($\underline{\times}$) and anti-commutator product ($\overline{\times}$) can be used to model a distance measure to ensure well conditioned equations in the presence of noise.

The main advantages of the constraint equations are the following: Firstly, the constraints are expressed in a multiplicative manner, they are concise and easy to interpret (see equation (4.16)). This is the basis for further extensions, like kinematic chains and other higher order algebraic entities. Secondly, the whole geometry within the scenario is concerned and strictly modeled. This ensures an optimal treating of the geometry and the knowledge that no geometric aspects have been neglected or approximated which is sometimes done in the literature by e.g. using orthographic camera models [25] etc.

## Chapter 5

# POSE CONSTRAINTS FOR POINT, LINE AND PLANE CORRESPONDENCES

So far I have introduced the entities, their transformations and the interaction of Euclidean, projective and conformal geometry. Furthermore, the modeling of the pose problem has been clarified and a small preview of the proposed constraint equations has been given. Still missing is the analysis of the used operators to express collinearity and tangentiality of points, lines and planes. This will be done in this section.

Note: As can be seen from chapter 4, the transformation from an entity given in the projective geometric algebra to the conformal geometric algebra always leads to a dual representation of the entity, since

$$\begin{aligned} \mathbf{e} \wedge \boldsymbol{X} &= \underline{\boldsymbol{X}}^{\star} \\ \mathbf{e} \wedge \boldsymbol{L} &= \underline{\boldsymbol{L}}^{\star} \\ \mathbf{e} \wedge \boldsymbol{P} &= \underline{\boldsymbol{P}}^{\star}. \end{aligned} \tag{5.1}$$

In the next chapters I will work only with the dual representation of the entities and therefore from now on the $\star$-sign will be neglected in the equations.

In this chapter constraints will be derived to express collinearity and coplanarity for points, lines and planes. The constraints will be given in the conformal space. They are then translated to an error measure of the Euclidean space. While this chapter only deals with the relation of points, lines and planes, the following chapters will regard the constraints to relate other entities like circles, spheres, cycloidal curves, etc.

Table 5.1 gives an overview of the constraints for collinearity and coplanarity of points, lines and planes in conformal geometric algebra. Indeed

| Entities | Constraint in conformal geometric algebra |
|---|---|
| point-line | $\underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{L}} = 0$ |
| point-plane | $\underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{P}} = 0$ |
| line-plane | $\underline{\boldsymbol{L}} \overline{\times} \underline{\boldsymbol{P}} = 0$ |

Tab. 5.1: The geometric constraints for collinearity and coplanarity of points, lines and planes expressed in conformal geometric algebra.

there is no unique representation to model incidence of entities. Therefore I searched for an expression of collinearity and coplanarity which is not only compact and linear, but also contains a distance measure which can numerically stable and fast applied to the pose problem. The reason why I use these equations is the fact, that they express such a distance measure without introducing non-linearities within the unknowns. The constraints are inspired by W. Blaschke [21] who formalized three constraint equations for incidence of points, lines and planes in the dual-quaternions. I then translated the equations in [167, 145] to the motor algebra and the conformal geometric algebra, respectively. So far I found no better equations for the pose scenario which are compact, linear, contain a distance measure, can be applied to a perspective camera model and are suited for the use of different entities simultaneously. Now it will be continued with a geometric analysis of the constraint equations introduced in table 5.1.

## 5.1 Point-line constraint

Evaluating the point-line constraint of a point $\underline{\boldsymbol{X}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{X}} = \boldsymbol{E} + \mathbf{e}\boldsymbol{x}$, collinear with a line $\underline{\boldsymbol{L}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{L}} = \boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m}$, leads to

$$
\begin{aligned}
0 &= \underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{L}} \\
&= \frac{1}{2}\left(\underline{\boldsymbol{X}\boldsymbol{L}} - \underline{\boldsymbol{L}\boldsymbol{X}}\right) \\
&= \frac{1}{2}\left((\boldsymbol{E} + \mathbf{e}\boldsymbol{x})(\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m}) - (\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})(\boldsymbol{E} + \mathbf{e}\boldsymbol{x})\right) \\
&= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{E}\boldsymbol{r} + \boldsymbol{E}\mathbf{e}\boldsymbol{m} + \boldsymbol{E}^2\boldsymbol{r} - \mathbf{e}\boldsymbol{m}\boldsymbol{E} - \boldsymbol{r}\boldsymbol{E}\mathbf{e}\boldsymbol{x} - \boldsymbol{r}\right) \\
&= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{r} - \boldsymbol{m}\mathbf{e} - \boldsymbol{m}\mathbf{e} - \boldsymbol{r}\boldsymbol{x}\mathbf{e}\right) \\
&= \frac{1}{2}\left(-(2\boldsymbol{m} - (\boldsymbol{x}\boldsymbol{r} - \boldsymbol{r}\boldsymbol{x}))\mathbf{e}\right) \\
&= -(\boldsymbol{m} - \boldsymbol{x} \underline{\times} \boldsymbol{r})\mathbf{e}
\end{aligned}
$$

$$\Leftrightarrow 0 \quad = \quad (\boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r})\mathbf{e}\cdot\mathbf{e}_+ = \boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r}. \tag{5.2}$$

The product $\underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{L}}$ is an element of the null space, since $\mathbf{e}^2 = 0$. By computing the inner product with $\mathbf{e}_+$, this expression can be changed to an equation in the non-null space. Note, that this is consistent with table 4.2: Computing the inner product with $\mathbf{e}_+$ leads to an error direction in the projective space and since the homogeneous component is zero it is simultaneously a vector expression in $\boldsymbol{\mathcal{G}}_{3,0}$, the algebra of the 3D Euclidean space.

The term $\boldsymbol{m}-\boldsymbol{x}\underline{\times}\boldsymbol{r}$ means that the moment $\boldsymbol{m}$ of a line, which is generated by the outer product of the direction $\boldsymbol{r}$ of the line with a point $\boldsymbol{x}$ on the line, is independent of the chosen point of the line. This is a clear fact from Plücker representation of lines [21].

So far the constraint equation is given unscaled. Following section 2.2, a scaling parameter $\lambda \in \mathbb{R}$ has to be applied to express a distance measure in the Euclidean space. In this case let $\lambda = \frac{1}{\|\boldsymbol{r}\|}$. That means that the equation is scaled with the inverse norm of the direction of the line. This leads to

$$\begin{aligned} 0 &= \boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r} \\ \Leftrightarrow 0 &= \lambda(\boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r}) \\ \Leftrightarrow 0 &= \lambda\boldsymbol{m} - \boldsymbol{x}\underline{\times}(\lambda\boldsymbol{r}) \\ \Leftrightarrow 0 &= \boldsymbol{m}' - \boldsymbol{x}\underline{\times}\boldsymbol{r}'. \end{aligned} \tag{5.3}$$

The aim is to analyze the bivector $\boldsymbol{m}' - \boldsymbol{x}\underline{\times}\boldsymbol{r}'$. Suppose $\underline{\boldsymbol{X}} \notin \underline{\boldsymbol{L}}'$. Then, nonetheless, there exists a decomposition $\boldsymbol{x} = \boldsymbol{x_1} + \boldsymbol{x_2}$ with $\underline{\boldsymbol{X_1}} \in \underline{\boldsymbol{L}}'$, $\underline{\boldsymbol{X_1}} = (\boldsymbol{E} + \mathbf{e}\boldsymbol{x_1})$ and $\underline{\boldsymbol{X_2}} \perp \underline{\boldsymbol{L}}'$, $\underline{\boldsymbol{X_2}} = (\boldsymbol{E} + \mathbf{e}\boldsymbol{x_2})$. Figure 5.1 shows the scenario.

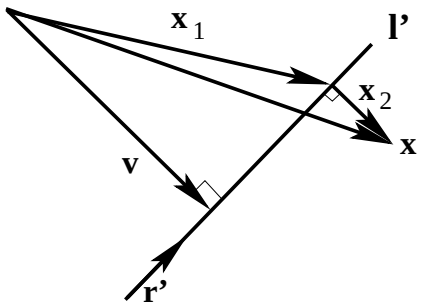



Fig. 5.1: The Euclidean line $\boldsymbol{l}'$ consists of the direction $\boldsymbol{r}'$ and the moment $\boldsymbol{m}' = \boldsymbol{v}\underline{\times}\boldsymbol{r}'$. Furthermore there exists a decomposition $\boldsymbol{x} = \boldsymbol{x_1} + \boldsymbol{x_2}$ with $\boldsymbol{x_1} \in \boldsymbol{l}'$ and $\boldsymbol{x_2} \perp \boldsymbol{r}'$, so that $\boldsymbol{m}' = \boldsymbol{v}\underline{\times}\boldsymbol{r}' = \boldsymbol{x}_1\underline{\times}\boldsymbol{r}'$.

Then can be calculated

$$\|\boldsymbol{m}' - \boldsymbol{x}\underline{\times}\boldsymbol{r}'\| \quad = \quad \|\boldsymbol{m}' - (\boldsymbol{x_1} + \boldsymbol{x_2})\underline{\times}\boldsymbol{r}'\|$$

$$\begin{aligned} &= \|\boldsymbol{m}' - \boldsymbol{x_1}\underline{\times}\boldsymbol{r}' - \boldsymbol{x_2}\underline{\times}\boldsymbol{r}'\| \\ &= \|\boldsymbol{x_2}\underline{\times}\boldsymbol{r}'\| = \|\boldsymbol{x_2}\|. \end{aligned} \tag{5.4}$$

Thus satisfying the scaled point-line constraint means to equate the bivectors $\boldsymbol{m}'$ and $\boldsymbol{x} \times \boldsymbol{r}'$, respectively making the Hesse distance $\|\boldsymbol{x_2}\|$ of the Euclidean point $\boldsymbol{x}$ to the Euclidean line $\boldsymbol{l}'$ to zero.

## 5.2 Point-plane constraint

Evaluating the point-plane constraint of a point $\underline{\boldsymbol{X}} = \boldsymbol{E} + \mathbf{e}\boldsymbol{x}$ coplanar to a plane $\underline{\boldsymbol{P}} = \boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E$ leads to

$$\begin{aligned} 0 &= \underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{P}} \\ &= \frac{1}{2}\left(\underline{\boldsymbol{X}\boldsymbol{P}} - \underline{\boldsymbol{P}\boldsymbol{X}}\right) \\ &= \frac{1}{2}\left((\boldsymbol{E} + \mathbf{e}\boldsymbol{x})(\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E) - (\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E)(\boldsymbol{E} + \mathbf{e}\boldsymbol{x})\right) \\ &= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{E}\boldsymbol{n} + \boldsymbol{E}\mathbf{e}d\boldsymbol{I}_E + \boldsymbol{n} - \mathbf{e}d\boldsymbol{I}_E\boldsymbol{E} - \boldsymbol{E}\boldsymbol{n}\mathbf{e}\boldsymbol{x} - \boldsymbol{n}\right) \\ &= \frac{1}{2}\left(-\boldsymbol{x}\boldsymbol{n}\mathbf{e} + d\boldsymbol{I}_E\mathbf{e} + \boldsymbol{n} + d\boldsymbol{I}_E\mathbf{e} - \boldsymbol{n}\boldsymbol{x}\mathbf{e} - \boldsymbol{n}\right) \\ &= \frac{1}{2}\left(2d\boldsymbol{I}_E - (\boldsymbol{x}\boldsymbol{n} + \boldsymbol{n}\boldsymbol{x})\right)\mathbf{e} \\ &= (d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n})\mathbf{e} \\ \Leftrightarrow 0 &= (d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n}))\mathbf{e} \cdot \mathbf{e}_+ = d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n}). \end{aligned} \tag{5.5}$$

Note here that the anticommutator product of the bivector $\boldsymbol{n}$ and the vector $\boldsymbol{x}$ lead to a trivector, which is subtracted from $d\boldsymbol{I}_E$. Again the constraint equation is given in the null space which is then transformed to the non-null space by computing the dot-product with $\mathbf{e}_+$. This leads directly to a scalar value as element of the Euclidean geometric algebra. To express a distance measure in the Euclidean space, let $\lambda = \frac{1}{\|\boldsymbol{n}\|}$, the inverse of the norm of the normal. This leads to

$$\begin{aligned} 0 &= d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n}) \\ \Leftrightarrow 0 &= \lambda(d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}) \\ \Leftrightarrow 0 &= \lambda d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}(\lambda\boldsymbol{n}) \\ \Leftrightarrow 0 &= d'\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}'. \end{aligned} \tag{5.6}$$

Suppose $\underline{\boldsymbol{X}} \notin \underline{\boldsymbol{P}}'$. The value $d'$ can be interpreted as the sum of distances, so that $d' = d'_{01} + d'_{02}$ and $d'_{01}\boldsymbol{n}'$ is the orthogonal projection of $\boldsymbol{x}$ onto $\boldsymbol{n}'$. Figure 5.2 shows the scenario. Then the following equation holds,

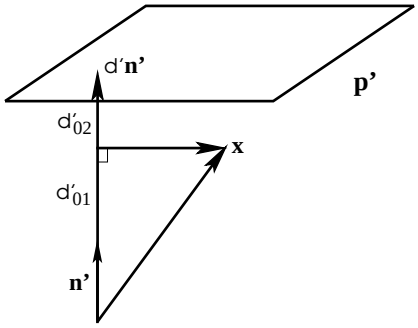



Fig. 5.2: The Euclidean plane $\boldsymbol{p}'$ is represented by the normal $\boldsymbol{n}'$, and the Hesse distance $d'$. The value $d'$ can be interpreted as a sum $d' = d'_{01} + d'_{02}$ so that $d'_{01}\boldsymbol{n}'$ corresponds to the orthogonal projection of $\boldsymbol{x}$ onto $\boldsymbol{n}'$.

$$\begin{aligned} d'\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}' &= (d'_{01} + d'_{02})\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}' \\ &= d'_{02}\boldsymbol{I}_E. \end{aligned} \tag{5.7}$$

The value of the expression $d'\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}'$ corresponds to the Hesse distance of the Euclidean point $\boldsymbol{x}$ to the Euclidean plane $\boldsymbol{p}'$.

## 5.3 Line-plane constraint

Evaluating the line-plane constraint of a line $\underline{\boldsymbol{L}} = \boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m}$ coplanar to a plane $\underline{\boldsymbol{P}} = \boldsymbol{E}\boldsymbol{n} + \mathbf{e}\boldsymbol{I}_E d$ leads to

$$\begin{aligned} 0 &= \underline{\boldsymbol{L}}\,\overline{\times}\,\underline{\boldsymbol{P}} \\ &= \frac{1}{2}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{P}} + \underline{\boldsymbol{P}}\underline{\boldsymbol{L}}\right) \\ &= \frac{1}{2}\left((\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})(\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E) + (\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E)(\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})\right) \\ &= \frac{1}{2}\left(\mathbf{e}\boldsymbol{m}\boldsymbol{E}\boldsymbol{n} + \boldsymbol{r}\boldsymbol{E}\mathbf{e}\boldsymbol{I}_E d + \boldsymbol{r}\boldsymbol{n} + \mathbf{e}\boldsymbol{I}_E d\boldsymbol{r}\boldsymbol{E} + \boldsymbol{E}\boldsymbol{n}\mathbf{e}\boldsymbol{m} + \boldsymbol{E}\boldsymbol{n}\boldsymbol{r}\boldsymbol{E}\right) \\ &= \frac{1}{2}\left(\boldsymbol{m}\boldsymbol{n}\mathbf{e} + \boldsymbol{r}\boldsymbol{I}_E d\mathbf{e} + \boldsymbol{r}\boldsymbol{n} + \boldsymbol{I}_E d\boldsymbol{r}\mathbf{e} - \boldsymbol{n}\boldsymbol{m}\mathbf{e} + \boldsymbol{n}\boldsymbol{r}\right) \\ &= \frac{1}{2}\left((\boldsymbol{r}\boldsymbol{n} + \boldsymbol{n}\boldsymbol{r}) + (2\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\boldsymbol{n} - \boldsymbol{n}\boldsymbol{m})\mathbf{e}\right) \end{aligned}$$

$$\begin{aligned} &= \frac{1}{2}\left((\boldsymbol{r}\boldsymbol{n}+\boldsymbol{n}\boldsymbol{r})+2(\boldsymbol{r}\boldsymbol{I}_E d+\boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e}\right) \\ &= \boldsymbol{r}\overline{\times}\boldsymbol{n}+(\boldsymbol{r}\boldsymbol{I}_E d+\boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e}. \end{aligned} \tag{5.8}$$

Thus the constraint for coplanarity of a line to a plane can be partitioned into a constraint on the non-null part of the motor and a constraint on the null part of the motor. This can directly be seen in equation (5.8), since $\mathbf{e}^2 = 0$.

Again the constraint equation is given unscaled. Let be $\lambda = \frac{1}{\|\boldsymbol{n}\|\|\boldsymbol{r}\|}$. Inserting $\lambda$ leads to

$$\begin{aligned} 0 &= \boldsymbol{r}\overline{\times}\boldsymbol{n}+(\boldsymbol{r}\boldsymbol{I}_E d+\boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e} \\ \Leftrightarrow 0 &= \lambda\left(\boldsymbol{r}\overline{\times}\boldsymbol{n}+(\boldsymbol{r}\boldsymbol{I}_E d+\boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e}\right) \\ \Leftrightarrow 0 &= \boldsymbol{r'}\overline{\times}\boldsymbol{n'}+(\boldsymbol{r'}\boldsymbol{I}_E d'+\boldsymbol{m'}\underline{\times}\boldsymbol{n'})\mathbf{e}, \end{aligned} \tag{5.9}$$

with

$$\boldsymbol{r'} = \tfrac{1}{\|\boldsymbol{r}\|}\boldsymbol{r} \quad \boldsymbol{n'} = \tfrac{1}{\|\boldsymbol{n}\|}\boldsymbol{n} \quad \boldsymbol{m'} = \tfrac{1}{\|\boldsymbol{r}\|}\boldsymbol{m} \quad d' = \tfrac{1}{\|\boldsymbol{n}\|}d.$$

Suppose $\underline{\boldsymbol{L}'} \notin \underline{\boldsymbol{P}'}$. If $\boldsymbol{r'} \not\perp \boldsymbol{n'^{\star}}$, the non-null part results in

$$\boldsymbol{r'}\overline{\times}\boldsymbol{n'} = \|\boldsymbol{r'}\|\|\boldsymbol{n'}\|\cos(\alpha) = \cos(\alpha), \tag{5.10}$$

where $\alpha$ is the angle between $\underline{\boldsymbol{L}'}$ and $\underline{\boldsymbol{P}'}$, see figure 5.3. If $\boldsymbol{r'} \perp \boldsymbol{n'^{\star}}$, it leads to $\boldsymbol{r'}\overline{\times}\boldsymbol{n'} = 0$. Since the direction of the line is independent of the translation of the rigid body motion, the constraint on the non-null part can be used to generate equations with the parameters of the rotation as the only unknowns. The constraint on the null part can then be used to determine the unknown translation. In other words, since the motor which is to be estimated, $\boldsymbol{M} = \boldsymbol{R}'_1 + \mathbf{e}\boldsymbol{R}'_2$, is determined in its non-null part only by rotation, the non-null part of the constraint allows to estimate the rotor $\boldsymbol{R}'_1$, while the null part of the constraint allows to estimate the rotor $\boldsymbol{R}'_2$. So it is possible to sequentially separate equations on the unknown rotation from equations on the unknown translation without the limitations, known from the embedding of the problem in Euclidean space [41]. This is useful, since the two smaller equation systems are easier to solve than one larger equation system. To analyze the null part of the constraint, let the moment $\boldsymbol{m'}$ of the line representation $\underline{\boldsymbol{L}'} = \boldsymbol{E}\boldsymbol{r'} + \mathbf{e}\boldsymbol{m'}$ be interpreted as $\boldsymbol{m'} = \boldsymbol{s}\underline{\times}\boldsymbol{r'}$ by choosing a vector $\boldsymbol{s}$ with $\boldsymbol{s} \in \boldsymbol{l'}$ and $\boldsymbol{s} \perp \boldsymbol{r'}$. Following [133], this leads to

$$\begin{aligned} \boldsymbol{n'}\underline{\times}\boldsymbol{m'} &= -(\boldsymbol{s}\underline{\times}\boldsymbol{r'})\underline{\times}\boldsymbol{n'} \\ &= (\boldsymbol{s}\overline{\times}\boldsymbol{n'})\overline{\times}\boldsymbol{r'} - \boldsymbol{s}\overline{\times}(\boldsymbol{r'}\overline{\times}\boldsymbol{n'}). \end{aligned} \tag{5.11}$$

Now can be evaluated

$$d\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}') \;=\; d\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}' + \boldsymbol{s}\overline{\times}(\boldsymbol{r}'\overline{\times}\boldsymbol{n}'). \qquad (5.12)$$

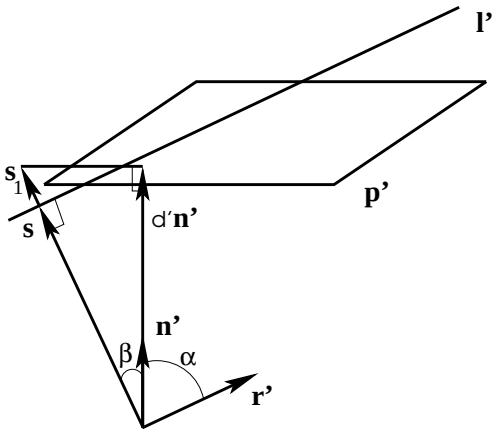



Fig. 5.3: The Euclidean plane $\boldsymbol{p}'$ is represented by its normal $\boldsymbol{n}'$ (as bivector) and the Hesse distance $d'$. Furthermore $\boldsymbol{s} \in \boldsymbol{l}'$ is chosen with $\boldsymbol{s} \perp \boldsymbol{r}'$. The angle of $\boldsymbol{r}'$ and $\boldsymbol{n}'^{\star}$ is $\alpha$ and the angle of $\boldsymbol{s}$ and $\boldsymbol{n}'^{\star}$ is $\beta$. The vector $\boldsymbol{s_1}$ with $\boldsymbol{s} \parallel \boldsymbol{s_1}$ is chosen so that $d\boldsymbol{n}'^{\star}$ is the orthogonal projection of $(\boldsymbol{s} + \boldsymbol{s_1})$ onto $\boldsymbol{n}'^{\star}$.

Figure 5.3 shows the scenario. Furthermore it is possible to find a vector $\boldsymbol{s_1} \parallel \boldsymbol{s}$ with $0 = d' - (\|\boldsymbol{s}\| + \|\boldsymbol{s_1}\|)\cos(\beta)$. The vector $\boldsymbol{s_1}$ might also be antiparallel to $\boldsymbol{s}$. This leads to a change of the sign, but does not affect the constraint itself. Now can be evaluated

$$\begin{aligned} d'\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}') &= d'\boldsymbol{I}_E\boldsymbol{r}' - \|\boldsymbol{s}\|\cos(\beta)\boldsymbol{r}' + \cos(\alpha)\boldsymbol{s} \\ &= \|\boldsymbol{s}_1\|\cos(\beta)\boldsymbol{r}' + \cos(\alpha)\boldsymbol{s}. \end{aligned} \qquad (5.13)$$

Thus, the error of the null part of the motor is constituted by the sum of the vector $\boldsymbol{s}$, scaled by the angle $\alpha$, and the direction vector $\boldsymbol{r}'$, scaled by the norm of $\boldsymbol{s_1}$ and the angle $\beta$.
If $\boldsymbol{r}' \perp \boldsymbol{n}'^{\star}$, then $\boldsymbol{n}'^{\star} \parallel \boldsymbol{s}$ and thus,

$$\begin{aligned} \|d'\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}')\| &= \|d'\boldsymbol{I}_E\boldsymbol{r}' + \boldsymbol{s}\overline{\times}(\boldsymbol{r}'\overline{\times}\boldsymbol{n}') - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}'\| \\ &= \|d'\boldsymbol{I}_E\overline{\times}\boldsymbol{r}' - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}'\| \\ &= \|d'\boldsymbol{I}_E - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\|. \end{aligned} \qquad (5.14)$$

This means, in agreement with the point-plane constraint, that the above distance measure corresponds to the Hesse distance of the line to the plane. Since equation 5.13 contains the error vector $\boldsymbol{s}$, its error value is dependent on the chosen origin of the vector space. This effect is indeed unwanted and can lead to bad conditioned equations, but if $\boldsymbol{n}'^{\star}$ is nearly parallel to $\boldsymbol{s}$ good conditioned equations can be assured since they are then related to the point-plane constraint.

This analysis shows that the considered constraints are not only qualitative constraints, but also quantitative ones. This is very important, since I want to measure the extend of fulfillment of these constraints in the case of noisy data.

## 5.4 Constraint equations for pose estimation

Now it is possible to express the 2D-3D pose estimation problem in a quantitative manner. The aim is to express that *a transformed object entity has to lie on a projective reconstructed image entity* in the conformal geometric algebra. Let $\underline{\boldsymbol{X}} \in \boldsymbol{\mathcal{G}}_{4,1}$ be an object point and $\underline{\boldsymbol{L}} \in \boldsymbol{\mathcal{G}}_{4,1}$ be an object line. The (unknown) transformed entities can be written as

$$\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \quad \text{and} \tag{5.15}$$

$$\underline{\boldsymbol{L}}' = \boldsymbol{M}\underline{\boldsymbol{L}}\widetilde{\boldsymbol{M}}. \tag{5.16}$$

Let $\boldsymbol{x} \in \boldsymbol{\mathcal{G}}_{2,1}$ be an image point and $\boldsymbol{l} \in \boldsymbol{\mathcal{G}}_{2,1}$ be an image line. Note, that I denote the 2D projective image features also with small bold letters, similar to 3D Euclidean points. The reason is, that both algebras are built from three basis vectors, they can not be confounded in the scenario and it avoids extra fonds. The projective reconstruction of these entities can be written as

$$\boldsymbol{L}_x = \boldsymbol{O} \wedge \boldsymbol{x} \in \boldsymbol{\mathcal{G}}_{3,1} \quad \text{and} \tag{5.17}$$

$$\boldsymbol{P}_l = \boldsymbol{O} \wedge \boldsymbol{l} \in \boldsymbol{\mathcal{G}}_{3,1}. \tag{5.18}$$

The vector $\boldsymbol{O} \in \boldsymbol{\mathcal{G}}_{3,1}$ denotes the optical center of the camera. Then the $\mathbf{e}\wedge$-operator can be applied to change the representations from the projective to the conformal space, and combined with the commutator and anticommutator products to express the collinearity and coplanarity of the involved entities.

Thus, the constraint equations of pose estimation can be read in the following manner:

Point-line constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\ \underbrace{\boldsymbol{O}}_{\text{optical center}}\ \wedge\ \underbrace{\boldsymbol{x}}_{\text{image point}}\ )}_{\substack{\text{projection ray,}\\ \text{reconstructed from the image point}}}}_{\substack{\text{collinearity of the transformed object}\\ \text{point with the reconstructed line}}}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (5.19)$$

$$\text{distance measure between 3D point and 3D line}$$

Point-plane constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\ \underbrace{\boldsymbol{O}}_{\text{optical center}}\ \wedge\ \underbrace{\boldsymbol{l}}_{\text{image line}}\ )}_{\substack{\text{3D plane,}\\ \text{reconstructed from the image line}}}}_{\substack{\text{coplanarity of the transformed object}\\ \text{point with the reconstructed plane}}}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (5.20)$$

$$\text{distance measure between 3D point and 3D plane}$$

Line-plane constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{L}}}_{\text{object line}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object line}}\ \overline{\times}\ \underbrace{\mathbf{e}\wedge(\ \underbrace{\boldsymbol{O}}_{\text{optical center}}\ \wedge\ \underbrace{\boldsymbol{l}}_{\text{image line}}\ )}_{\substack{\text{3D plane,}\\ \text{reconstructed from the image line}}}}_{\substack{\text{coplanarity of the transformed object}\\ \text{line with the reconstructed plane}}}\Big) \;=\; 0. \quad (5.21)$$

$$\text{distance measure between 3D line and 3D plane}$$

The involved mathematical spaces are (again) exemplarily shown for the point-line constraint,

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{CS}\ \widetilde{\boldsymbol{M}})}_{CS}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\underbrace{\underbrace{\boldsymbol{O}}_{PS}\wedge\underbrace{\boldsymbol{x}}_{PP}}_{PS})}_{CS}}_{ES}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (5.22)$$

Here $PP$ abbreviates *projective plane*, $PS$ *projective space*, $CS$ *conformal space* and $ES$ the *Euclidean space*. These compact equations subsume the pose estimation problem at hand: find the best motor $\boldsymbol{M}$ which satisfies the constraint. The 2D-3D pose estimation problem is modeled in terms of constraint equations. Note, that the stratification hierarchy of the involved entities is strictly kept within these equations. Furthermore the equations are

compact and therefore easy to interpret. Additionally the geometric analysis of the constraints assures well conditioned equations and helps to interpret effects of the constraints discussed in the experiments. The constraints behave robust in case of noisy data and linearization and iteration allow for the design of fast (real-time capable) algorithms. In contrast to other approaches, where the minimization of errors has to be computed directly on the manifold of geometric transformations [35, 177], in this approach a distance in the Euclidean space constitutes the error measure.

## 5.5 Numerical estimation of pose parameters

In the last sections, several constraint equations to relate object information to image information are derived. In these equations the object, camera and image information is assumed to be known and the motor $\boldsymbol{M}$ expressing the motion is assumed to be unknown. The main question is now, how to solve a set of constraint equations for multiple (different) features with respect to the unknown motor $\boldsymbol{M}$. Since a motor is a polynomial of infinite degree, see e.g. its series expression in equation (3.35) and (3.38), this is a non-trivial task, especially in the case of real-time estimations.

The idea is to gain linear equations with respect to the generators of the motor. The exponential representation of motors is used and the Taylor series expansion of first order is applied for approximation. This leads to a mapping of the above mentioned global motion transformation to a twist representation, which allows for incremental changes of pose. That means, that there is done no search for the parameters of the Lie group $SE(3)$ to describe the rigid body motion [62], but for the parameters which generate their Lie algebra $se(3)$ [124]. This results in linear equations in the generators of the unknown 3D rigid body motion. In this section the linearization of the motor is derived. For the sake of simplicity it will be done for the case of point transformations.

The Euclidean transformation of a point $\underline{\boldsymbol{X}}$ caused by the motor $\boldsymbol{M}$ is approximated in the following way:

$$\begin{aligned}
\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right)\underline{\boldsymbol{X}}\exp\left(\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right) \\
&\approx (1-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'}))\underline{\boldsymbol{X}}(1+\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})) \\
&\approx \boldsymbol{E}+\mathbf{e}(\boldsymbol{x}-\theta(\boldsymbol{l'}\cdot\boldsymbol{x})-\theta\boldsymbol{m'}). \qquad (5.23)
\end{aligned}$$

Setting $\boldsymbol{l} := \theta \boldsymbol{l'}$ and $\boldsymbol{m} := \theta \boldsymbol{m'}$ results in

$$\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \;\approx\; \boldsymbol{E} + \mathbf{e}(\boldsymbol{x} - \boldsymbol{l} \cdot \boldsymbol{x} - \boldsymbol{m}). \tag{5.24}$$

By combining this approximation of the motion with the previously derived constraints (e.g. the point-line constraint) it leads to

$$\begin{aligned} 0 &= \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \times \underline{\boldsymbol{L}} \\ \Leftrightarrow 0 &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l'} + \mathbf{e}\boldsymbol{m'})\right) \underline{\boldsymbol{X}} \exp\left(\frac{\theta}{2}(\boldsymbol{l'} + \mathbf{e}\boldsymbol{m'})\right) \times \underline{\boldsymbol{L}} \\ \Leftarrow\approx\Rightarrow 0 &= (\boldsymbol{E} + \mathbf{e}(\boldsymbol{x} - \boldsymbol{l} \cdot \boldsymbol{x} - \boldsymbol{m})) \times \underline{\boldsymbol{L}} \\ \Leftrightarrow 0 &= \lambda(\boldsymbol{E} + \mathbf{e}(\boldsymbol{x} - \boldsymbol{l} \cdot \boldsymbol{x} - \boldsymbol{m})) \times \underline{\boldsymbol{L}}. \end{aligned} \tag{5.25}$$

Because of the approximation ($\Leftarrow\approx\Rightarrow$) the unknown motion parameters $\boldsymbol{l}$

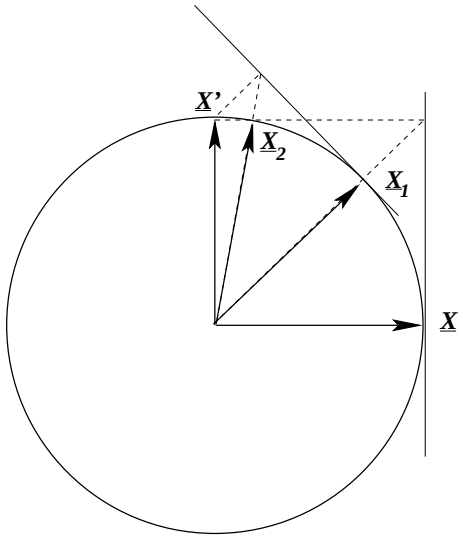



Fig. 5.4: Principle of convergence for the iteration of a point $\underline{\boldsymbol{X}}$ rotated around 90 degrees to a point $\underline{\boldsymbol{X}}'$. $\underline{\boldsymbol{X}}_1$ is the result of the first iteration and $\underline{\boldsymbol{X}}_2$ is the result of the second iteration.

and $\boldsymbol{m}$ are linear. This equation contains six unknown parameters for the rigid body motion. The unknowns are the unknown twist parameters for the motion. In the last step the linearized constraints are scaled with a suitable factor $\lambda$ to express an Euclidean distance measure as explained in section 5. This means, everything so far happens in the conformal space and only in the very last step the constraint equations are scaled to transform to the Euclidean space, as one of the strata of the hierarchy described in section 2.2.

The linear equations can be solved for a set of correspondences by applying e.g. the Householder method [136]. From the solution of the system of equations, the motion parameters $\boldsymbol{R}, \boldsymbol{t}$ can easily be recovered by evaluating $\theta := \|\boldsymbol{l}\|$, $\boldsymbol{l'} := \frac{\boldsymbol{l}}{\theta}$ and $\boldsymbol{m'} := \frac{\boldsymbol{m}}{\theta}$. The Motor $\boldsymbol{M}$ can be evaluated by applying the Rodrigues' formula [124, 62], see chapter A.2.

Figure 5.4 visualizes the principle of this approximation: The aim is to rotate a point $\underline{\boldsymbol{X}}$ by 90 degrees to a point $\underline{\boldsymbol{X}}'$. The first order approximation of the rotation leads to the tangent of the circle passing through $\underline{\boldsymbol{X}}$. Normalizing the tangent line to $\underline{\boldsymbol{X}}'$ (denoted by dashed lines) $\underline{\boldsymbol{X}}_1$ is gained as the first order approximation of the required point $\underline{\boldsymbol{X}}'$. By repeating this procedure the points $\underline{\boldsymbol{X}}_2, \ldots, \underline{\boldsymbol{X}}_n$ will be estimated, which converge to the point $\underline{\boldsymbol{X}}'$. It is clear from figure 5.4 that the convergence rate of a rotation is dependent on the amount of the expected rotation. An analysis of the convergence rate for general angles is given in figure 5.5 more detailed explained in the next section. I call this gradient descent method the *twist approach* for pose estimation since only the generators of the rigid body motion (the twists) are estimated and not the rigid body motion (as group action) itself.

### 5.5.1 Generating an example system of equations

In this section a system of equations will be derived for point, line and plane correspondences. The aim is to visualize the type of equations which are obtained. In this example are assumed two points $\boldsymbol{P}_i$,

$$\boldsymbol{P}_1 = (p_{11}, p_{12}, p_{13}) \tag{5.26}$$
$$\boldsymbol{P}_2 = (p_{21}, p_{22}, p_{23}), \tag{5.27}$$

one corresponding (Plücker) line $\boldsymbol{L}$ (containing a (unit) direction $\boldsymbol{L}_{d1}$ and a moment $\boldsymbol{L}_{m1}$),

$$\boldsymbol{L} = \{\boldsymbol{L}_{d1} = (Ld_{11}, Ld_{12}, Ld_{13}), \tag{5.28}$$
$$\boldsymbol{L}_{m1} = (Lm_{11}, Lm_{12}, Lm_{13})\} \tag{5.29}$$

and one plane $\boldsymbol{P}$ (containing a (unit) normal $\boldsymbol{P}_{d1}$ and Hesse distance $h_{d1}$),

$$\boldsymbol{P} = \{\boldsymbol{P}_{d1} = (Pd_{11}, Pd_{12}, Pd_{13}), h_{d1}\}. \tag{5.30}$$

Further let be assumed that $\boldsymbol{L}$ and $\boldsymbol{P}$ are reconstructed from image entities corresponding to $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. So that $\boldsymbol{P}_1$ is related to $\boldsymbol{L}$ and $\boldsymbol{P}_2$ is related to $\boldsymbol{P}$. Then the matrix for the system of equations takes the form

$$Ax = b, \tag{5.31}$$

with

$$A = \begin{pmatrix} 0 & Ld_{13} & -Ld_{12} \\ -Ld_{13} & 0 & Ld_{11} \\ Ld_{12} & -Ld_{11} & 0 \\ -Pd_{11} & -Pd_{12} & -Pd_{13} \end{pmatrix}$$

$$\begin{pmatrix} -p_{13}Ld_{13} - p_{12}Ld_{12} & p_{11}Ld_{12} & p_{11}Ld_{13} \\ p_{12}Ld_{11} & -p_{11}Ld_{11} - p_{13}Ld_{13} & p_{12}Ld_{13} \\ p_{13}Ld_{11} & p_{13}Ld_{12} & -p_{12}Ld_{12} - p_{11}Ld_{11} \\ -Pd_{13}p_{22} + Pd_{12}p_{23} & Pd_{13}p_{21} - Pd_{11}p_{23} & -Pd_{12}p_{21} + Pd_{11}p_{22} \end{pmatrix}.$$

The first three rows contain the components for a point-line constraint and the fourth row the components for a point-plane constraint. The solution vector $b$ has the form

$$\begin{aligned} b \;=\; & (-p_{12}Ld_{13} + p_{13}Ld_{12} + Lm_{11}, -p_{13}Ld_{11} + p_{11}Ld_{13} + Lm_{12}, \\ & -p_{11}Ld_{12} + p_{12}Ld_{11} + Lm_{13}, \\ & -h_{d1} + Pd_{11}p_{21} + Pd_{12}p_{22} + Pd_{13}p_{23})^T. \end{aligned} \tag{5.32}$$

The system of equations contains as unknowns $x$ the six twist parameters for which the equations are solved for.

Note, that though the point correspondences give three equations their rank is just two. This shows the well-known fact, that at least three point correspondences are necessary to solve the 2D-3D pose estimation problem. Furthermore every point-plane constraint gives exactly one equation, so that at least six correspondences are necessary to get a unique solution.

Once the six twist parameters $\theta$, $\boldsymbol{l'}$ and $\boldsymbol{m'}$ are determined from $\boldsymbol{l}$ and $\boldsymbol{m}$ (see equation 5.25), the group action can be recovered by applying the famous Rodrigues' formula (1840) [62].

## 5.6 Experiments with pose estimation of rigid objects

First of all, the convergence rate of a rotation by the angle $\theta$ during iterations will be studied. The result is demonstrated in figure 5.5. The $x$-axis represents the wanted angle $\theta$, the $y$-axis shows the estimated angle $\hat{\theta}$. Four iterations are overlaid. The functions are very characteristic and it can be

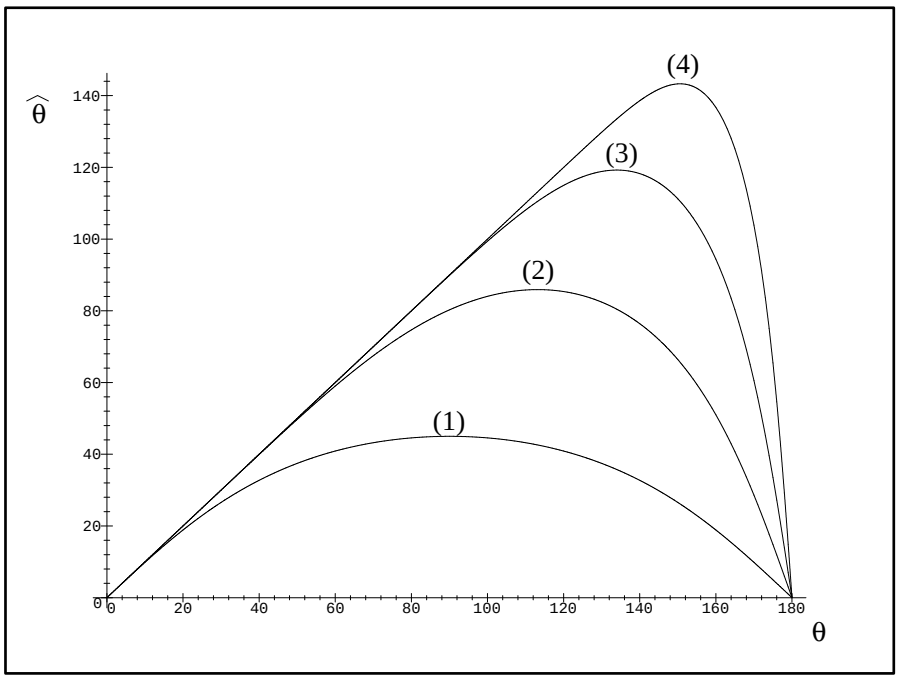



Fig. 5.5: Convergence rate of iterations for arbitrary angles between 0 and 180 degrees. The expected angles $\theta$ are on the $x$-axis and the estimated angles $\hat{\theta}$ are on the $y$-axis. The iterations $(1) \ldots (4)$ are overlaid.

seen that the contribution of the first iteration to gain a 90 degree rotation is 45 degree. This becomes clear by comparing the situation with figure 5.4.

All angles except that of 180 degree converge during the iteration. The reason is, that for a 180 degree rotation the nearest point on the line to the target point is the starting point itself. Therefor a 180 degree rotation is a degenerate case which is avoided in the experimental setup. For the most cases just a few iterations are sufficient to get a good approximation. In situations where only small rotations are assumed, two or three iterations are sufficient in most cases.

There exist several ways to estimate the motion parameters. A comparison of four approaches for pose estimation is made by Lorusso et al. in [115]. The algorithms deal with 3D point based pose estimation and are based on a SVD decomposition, unit quaternion (UQ), dual quaternion and eigensystem (OM) computation. Their results are not in agreement with results

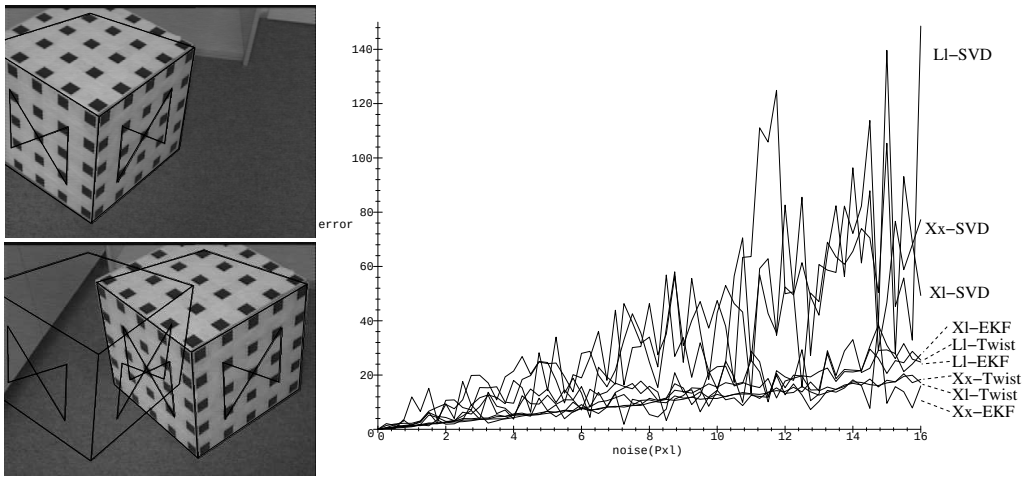



Fig. 5.6: The scenario of the first experiment. In the first image the calibration is performed and the 3D object model is projected on the image. Then the camera moves and corresponding line segments are extracted. For comparison reasons, the initial pose is overlaid. The diagram shows the performance comparison of different methods in case of noisy data.

presented in [179] and they figured out, that the SVD and UQ methods are very similar and usually the most stable. The OM method is not as stable for planar data sets, but superior for large degenerate data sets. The DQ algorithm was never the most stable and usually broke down before the others. Unfortunately they do not compare a gradient descent method within this context. In the next experiment, the noise sensivity of three methods for pose estimation is compared, with respect to the three constraint equations, relating 3D points to 2D points (Xx), 3D points to 2D lines (Xl), or 3D lines to 2D lines (Ll). The three estimating procedures are a simple SVD-approach, a Kalman filter [167] and the previously introduced *twist approach*. For comparing the noise sensivity, a Gaussian noise is added on extracted image points in a virtual scenario (see figure 5.6). Then the rigid body motion is estimated and the translational error between the ground truth and the distorted values is used as error measure. The result is depicted in figure 5.6. It is easy to see that the results obtained with the SVD approach are the worst ones. Instead, the Kalman filter and the twist approach have a more stable and comparable error behavior. It is obvious that the results of the

experiments are not much affected by the used constraints themselves. This occurs because certain points are selected by hand and from these the line subspaces are estimated. So the quality of the line subspaces is directly connected to the quality of the point extraction. The result of this investigation is that in case of systematic noise the Kalman filter or twist approach for pose estimation should be used. There are two main reasons, why I further prefer the twist approach for pose estimation instead of the EKF: Firstly, the (extended) Kalman filter is sensitive to outliers, leading to non-converging results. Secondly, a Kalman filter must be designed for special situations or scenarios. So the design of a general Kalman filter, dealing with different entities in a weighted manner or e.g. general kinematic chains is hard to implement. Instead, this can be done very easily in the twist approach since the linearized constraint equations can just be scaled and put in one system of equations.

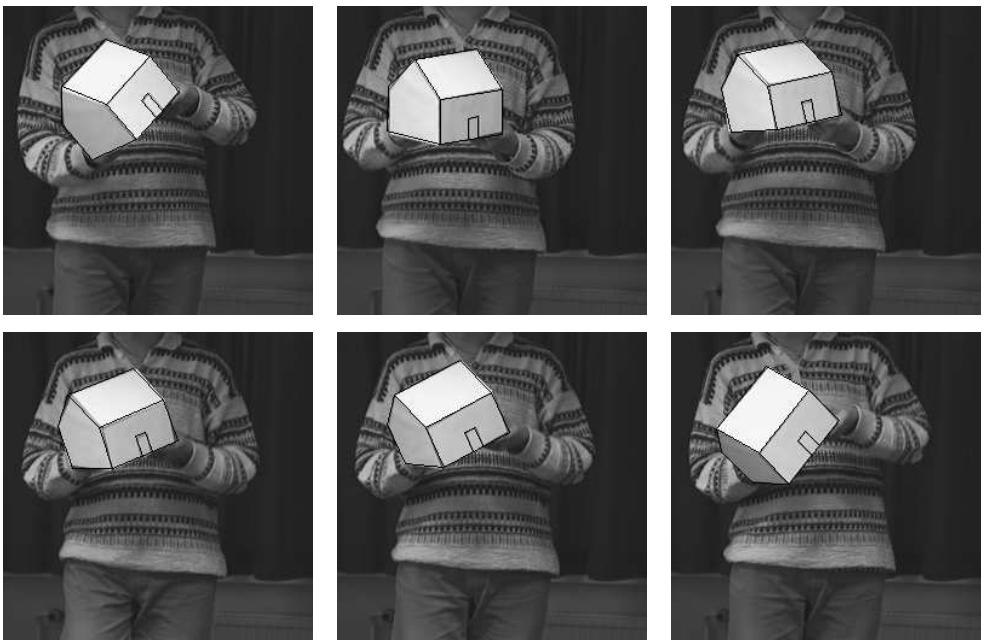


Fig. 5.7: Tracking a model house consisting of points and lines.

Figure 5.7 shows results of an automatic tracking algorithm developed and analyzed in [144] which is also explained in chapter B.1.2. The tracking algorithm is a heuristic which relies upon a combination of iterative improvement and random sampling. Iterative improvement refers to a repeated generate-and-test principle by which the algorithm moves from an initial state to its local optimum, see also [18].

### 5.6.1 Adaptive use of pose estimation constraints

Image preprocessing algorithms sometimes allow for a characterization of the quality of extracted image data (see e.g. [55]). The resulting question is how to deal with different extracted noisy image data. The idea to cope with

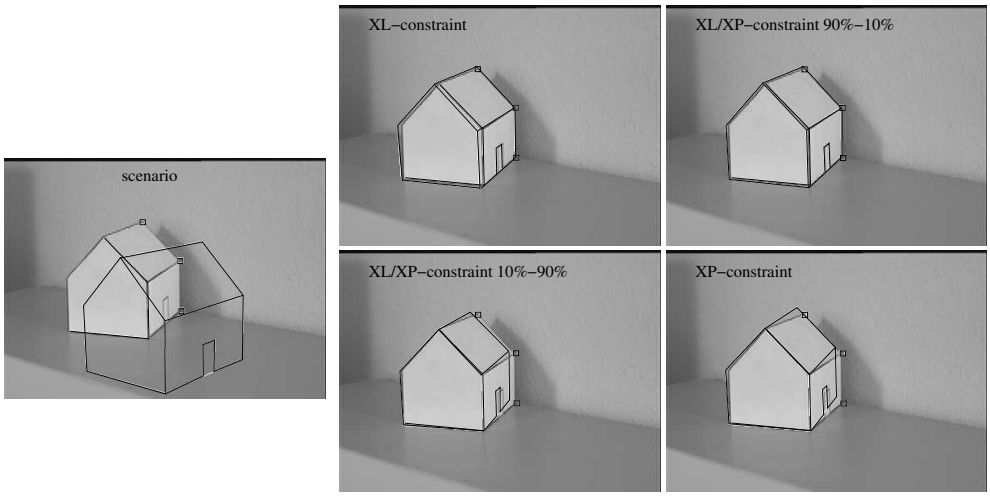



Fig. 5.8: Different weights of constraints for pose estimation. The upper left image shows the pose result for using point-line correspondences. The lower right image shows the pose result for using point-plane correspondences. The other images show pose results obtained with simultaneously used and weighted point-line and point-plane correspondences.

this problem in the context of pose estimation is simple: Every constraint equation of an image feature describes a distance measure of the involved entity. This constraint equation can be scaled by a factor $\lambda \in \mathbb{R}$ and so it is possible to individually scale the weights of the equations within the whole system of equations of an observed object. Figure 5.8 shows an example: Three extracted image points and three extracted image lines are given (see left image). It is possible to use both types of information separately to estimate the pose of the object. Since only little information for each type of correspondence is given, the object itself is not very well fitted to the image data, see e.g. the upper left or lower right images. On the other hand, it is possible to put both constraint equations in one single system of equations and solve the unknowns by using all available image information simultaneously. Furthermore, different weights of the constraints can be chosen. The change of the estimated pose is visualized in the other images of figure 5.8. This experiment demonstrates that the presented approach

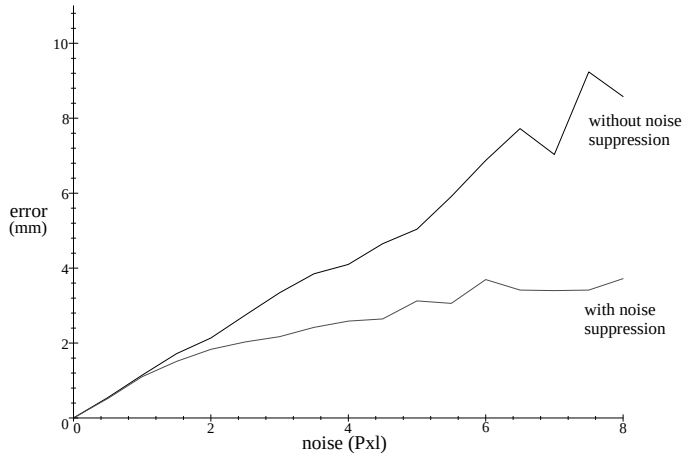



Fig. 5.9: Comparison of the results with and without noise suppression.

enables one to model adaptive observer behavior in a cognitive manner with respect to both the choice of image features at hand and the trustworthiness of the data.

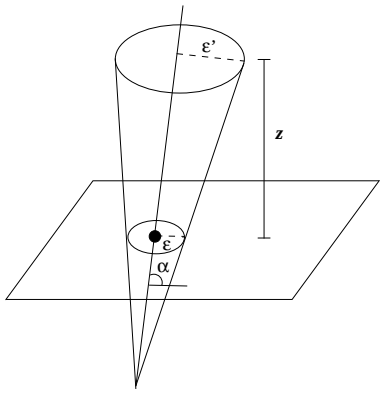



Fig. 5.10: The difference between building constraints in the image plane or in the 3D space. The 3D error measure $\epsilon'$ is geometrically dependent on the 2D image error $\epsilon$, the relative depth $\boldsymbol{z}$ and the angle $\alpha$ between the projection ray and the image plane.

In another experiment the possibility of noise adaptive use of the pose estimation constraints is simulated. For this a Gaussian noise is added on

some of the extracted image points. In this experiment six image points are used and onto two of them the Gaussian noise is added. Then the constraint equations are solved with and without weighting the constraints, depending on the noise level. The weights are chosen inverse proportional to the noise level. This means that the more noisy correspondences influence the whole result to a lesser extend. To compare the pose estimation results, the pose result without noise is used as ground truth. This experiment is repeated several times for every noise level to get a smooth error function and the mean value for every noise level is taken. The result is visualized in figure 5.9. It is easy to see that the constraints can be used in a noise adaptive manner.

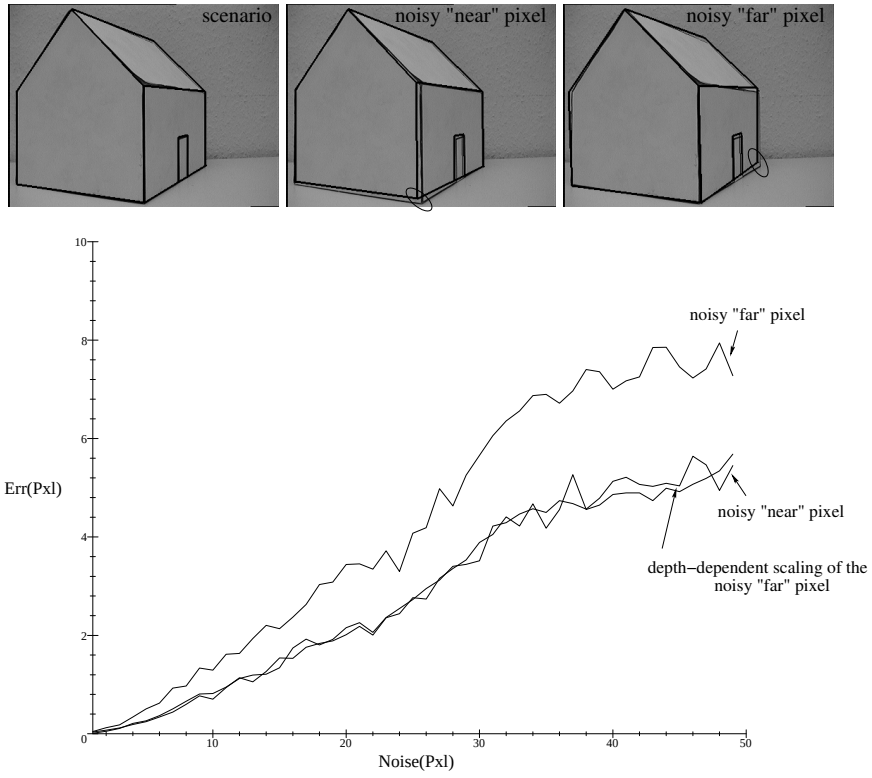



Fig. 5.11: Depth dependence of the 3D constraints with respect to the image plane.

As discussed in chapter 2.1, there is a difference between building constraints in the 3D space or in the 2D image plane: The noise in an image leads to a noise cone in the 3D space. This is visualized in figure 5.10: The

3D error measure $\epsilon'$ is geometrically dependent on the 2D image error $\epsilon$, the relative depth $\boldsymbol{z}$ and the angle $\alpha$ between the projection ray and the image plane. This property and its correction is analyzed in the scenario of figure 5.11. For this experiment, a calibrated scene with a model house is used. Then two points on the model are picked out and a Gaussian noise is put on their corresponding image points. Then their pose is estimated separately. The two chosen points differ in their relative depth with respect to the image plane as can be seen in figure 5.11. Then the absolute image error is taken. This means, the error measure is now connected to the observation of the pose in the image plane. The diagram in figure 5.11 shows the influence of the distorted image points to the estimated pose and their effect on the image plane. The pixel noise of the image point is given on the $x$-axis and on the $y$-axis the absolute pixel error of the transformed projected object model compared with the ground truth is shown. It can be seen that, though the image points are distorted in an equal manner, the result of the noisy *far* pixel is worse than the result of the noisy *near* pixel. This effect is often discussed as disadvantage of the 3D approach. But the possibility of noise adaptive use of the constraints is often neglected in this context. Since the constraint equations to formalize the pose problem contain a distance measure, the constraints can be scaled with respect to their relative depth. This is shown in the third error curve of figure 5.11.

## Chapter 6

# POSE ESTIMATION FOR EXTENDED OBJECT CONCEPTS

This chapter deals with the use of extended object concepts for pose estimation. With *extended object concepts* I mean kinematic chains, circles, spheres and cycloidal curves as entities, which extend the pose estimation scenario for point, line and plane features to more complex ones.

## 6.1 Pose estimation of kinematic chains

So far 3D pose constraints for points, lines and planes as features of a rigid object are parameterized. Assume that a second rigid body is attached to the first one by a joint. The joint can be formalized as an axis of rotation and/or translation in the object frame. If the joint $j$ is only dependent on a variable angle $\theta_j$, it is called a *revolute* joint, and it is called a *prismatic* joint if the degree of freedom is only a variable length $d_j$. This parameterization of joints is also called the Denavit-Hartenberg parameterization [42]. Each joint defines a new coordinate system, and the coordinate transformations between joints can be expressed by suitable motors $\boldsymbol{M}_j$. This means that an entity given in the coordinate system of the $j$th joint can be translated into an entity of the *base* coordinate system by transforming it with the motors $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_j$.

Such objects are also called *kinematic chains*. With kinematic chains I mean flexibly linked rigid objects which can only change their pose in mutual dependence. Examples are robot arms or human body movements, see e.g. figure 6.1. Kinematic chains can be parameterized by their including joints. Every joint defines a new coordinate system. To estimate the position of an end-effector entity of a kinematic chain in terms of the base coordinate sys-

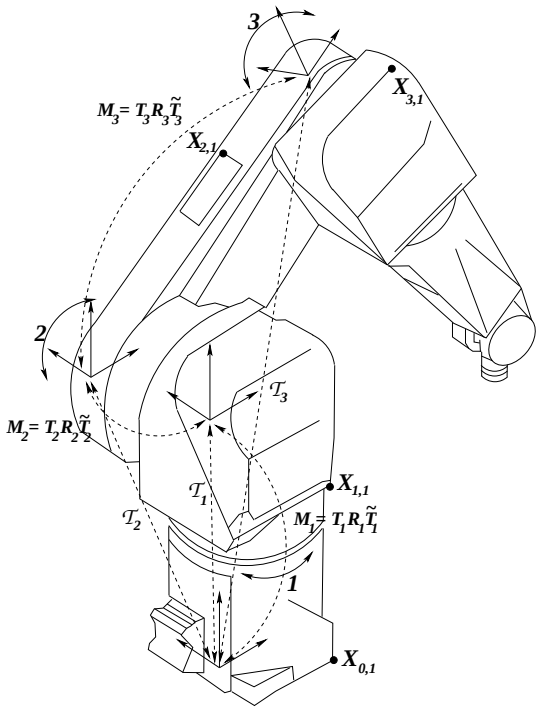



Fig. 6.1: The Stäubli RX-90 robot arm. The internal joint transformations $\boldsymbol{M}_j$ and the global transformation $\mathcal{T}_j$ are visualized.

tem, all involved joint coordinate systems must be traced. This is visualized in figure 6.1. For short notations of the single transformations between the joints I define

$$\begin{aligned} \mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\} &:= \underline{\boldsymbol{X}}_{0,i_0} \\ \mathcal{T}_j\{\underline{\boldsymbol{X}}_{j,i_j}, \boldsymbol{M}_j\} &:= \mathcal{T}_{j-1}\{\boldsymbol{M}_j \underline{\boldsymbol{X}}_{j,i_j} \widetilde{\boldsymbol{M}}_j, \boldsymbol{M}_{j-1}\} : j = 1, \ldots, n \\ &= \boldsymbol{M}_1 \ldots \boldsymbol{M}_j \underline{\boldsymbol{X}}_{j,i_j} \widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1 : j = 1, \ldots, n. \end{aligned} \quad (6.1)$$

The function $\mathcal{T}_0$ describes the identity for points which are not subject to internal transformations. I call them *base* points. The function $\mathcal{T}_j$ formalizes the transformation of an attached joint $j$ with respect to the base coordinate system in an inductive manner. In general, the transformation of a point $\underline{\boldsymbol{X}}_{j,i_j}$ on the $j$-th joint to the base coordinate system is represented by a sequence of motors $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_j$. An object model $\mathcal{O}$ of a kinematic chain with $n$ segments can now be represented by a set of $n+1$ functions $\mathcal{T}_j$,

$$\mathcal{O} = \{\mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\}, \mathcal{T}_1\{\underline{\boldsymbol{X}}_{1,i_1}, \boldsymbol{M}_1\}, \ldots, \mathcal{T}_n\{\underline{\boldsymbol{X}}_{n,i_n}, \boldsymbol{M}_n\} | n, i_0, \ldots, i_n \in \mathbb{N}\}. \quad (6.2)$$

### 6.1.1 Constraint equations of kinematic chains

Now the introduced representation of kinematic chains in CGA will be combined with the pose estimation constraints introduced in chapter 5. This is very simple now because everything is formulated in the same language. Note, that the constraints are presented unscaled, so the $\lambda(.) \cdot \mathbf{e}_+$ operation is not written extra (see e.g. equation 5.19).

The general unknown pose corresponds to a motor $\boldsymbol{M}$. For the base points $\underline{\boldsymbol{X}}_{0,i_0}$ the constraint equations reduce to

$$\begin{aligned}(\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{X}}_{0,i_0}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0}) &= 0 \end{aligned} \tag{6.3}$$

for a suitable projection ray $\underline{\boldsymbol{L}}_{0,i_0} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0})$. The general constraint equation for a point $\underline{\boldsymbol{X}}_{j,i_j}$ at the $j$-th joint leads to

$$\begin{aligned}(\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{X}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j\underline{\boldsymbol{X}}_{j,i_j}\widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{j,i_j}) &= 0. \end{aligned} \tag{6.4}$$

It is also simple to use extracted image lines $\boldsymbol{l}_{j,i_j}$ and their reconstructed projection planes $\underline{\boldsymbol{P}}_{j,i_j} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j})$. For such situations, the constraint equations reduce to

$$\begin{aligned}(\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{X}}_{0,i_0}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0, \end{aligned} \tag{6.5}$$

for the base points, and the general constraint equation for a point at the $j$th joint leads to

$$\begin{aligned}(\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{X}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j\underline{\boldsymbol{X}}_{j,i_j}\widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0. \end{aligned} \tag{6.6}$$

Kinematic chains can also be modeled by 3D lines and can be compared with reconstructed planes. For this only the lines $\underline{\boldsymbol{L}}_{j,i_j}$ and projection planes $\underline{\boldsymbol{P}}_{j,i_j} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j})$ have to be substituted and combined with the anti-commutator product. For base lines the constraint equation is

$$\begin{aligned}(\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{L}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \overline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{L}}_{0,i_0}\widetilde{\boldsymbol{M}}) \overline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0, \end{aligned} \tag{6.7}$$

and for a line on the $j$-th joint the constraint equation is

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{L}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \,\overline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j \underline{\boldsymbol{L}}_{j,i_j} \widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \,\overline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0. \quad (6.8) \end{aligned}$$

A publication about pose estimation of kinematic chains in CGA and the construction of pose estimation constraints can be found in [143]. O. Granert formalized in [66] the constraint equations for point concepts in pure matrix calculus. This was part of a student project I supervised. Note, that no hierarchical approach as in Hauck et al. [74] or Weik et al. [181] is used, instead a pose estimation based on the model of a kinematic chain. Hierarchical pose estimation means that the pose problem is separated in subtasks which are solved sequentially. So first the whole pose (the base transformation) is estimated and then each joint angle separately. To ensure that the model is not distorted after the calculations the estimated values have to be constrained to the model in a second processing step.

## 6.2 Circles and spheres

In this section constraint equations are developed to relate 3D circles to 2D ellipses and 3D spheres to 2D circles. This section starts with an analysis of the involved problems and will then continue with a suitable solution approach. The main problem is to define constraint equations in the 3D space which contain a geometric distance measure and allow to be combined with the previous formalization of constraints relating points, lines and planes.

### 6.2.1 The problem of tangentiality constraints

Since also the constraints for circles and spheres are derived in the 3D space, the aim is to reconstruct certain entities from image information and to compare the reconstructed entities with the 3D model entities. The reconstruction based on an image ellipse or an image circle (the image of a circle or sphere, respectively) leads to a cone. Indeed, it is not possible to formalize cones as single entities in conformal geometric algebra. But to make the above mentioned comparison possible, constraint equations are formalized to model tangentiality of 3D circles or spheres to projection rays, reconstructed from image points of the corresponding image entity. Figure 6.2 visualizes the idea.

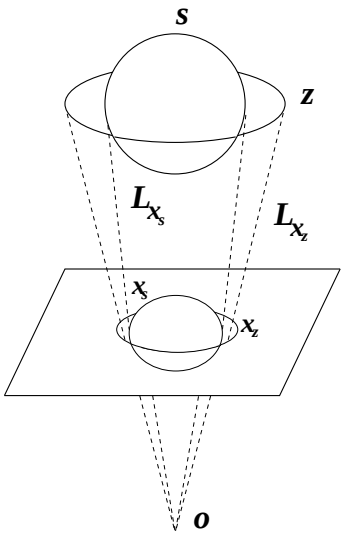



Fig. 6.2: Visualization of the 3D circle-line and sphere-line constraint.

It is easy in CGA to express e.g. tangentiality of a non-coplanar line $\underline{\boldsymbol{L}}$ to a circle $\underline{\boldsymbol{z}}$: The point $\underline{\boldsymbol{X}}_z := \underline{\boldsymbol{L}} \vee \underline{\boldsymbol{z}}$ is a null vector (this means $\underline{\boldsymbol{X}}_z^2 = 0$) iff the entities intersect. But this holds only in ideal geometry. In reality, there are several cases how a line can be related to a circle: it can intersect, be coplanar or perpendicular. The line can pass outside or inside the circle, etc. By defining a line in a parameterized manner, it is easy to see that the error function of points on a line to a circle can contain one global minimum, two global minima, one local and one global minimum or no minimum in non-degenerate and degenerate cases. In figure 6.3 three example lines are shown: Two lines are parallel to the plane in which the circle lies. One of these lines passes the circle outside, the other one inside. This leads to error functions, containing one global minimum or two global minima. The third line is passing the inside of the circle and is not parallel to the plane in which the circle lies. This leads to one global and one local minimum. From that result two possible strategies: Firstly, it is possible to make a case decision, depending on the geometric situation. This is hard to implement and to combine with the previously derived constraint equations. Secondly, it is possible to parameterize the circle in a suitable way. This will be done in the following section.

Comparing spheres with lines also is no problem in ideal geometry. One short way to formalize tangentiality is to estimate the distance from the center of the sphere to the line and to subtract the radius: Let $\underline{\boldsymbol{L}}'$ be the scaled line, as described in chapter 5. The line $\underline{\boldsymbol{L}}'$ is tangential to $\underline{\boldsymbol{s}} = \underline{\boldsymbol{p}} - \frac{1}{2}\rho^2\mathbf{e}$

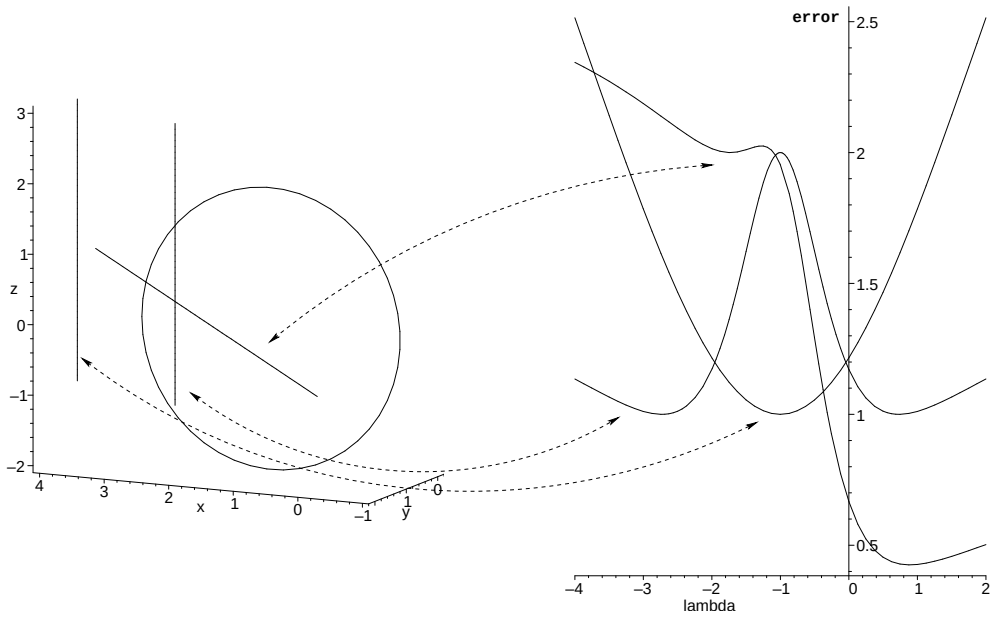



Fig. 6.3: Different geometric relations of lines to circles lead to different kinds of error functions for parameterized lines.

iff

$$\|(\underline{\boldsymbol{L}}' \underline{\times} \underline{\boldsymbol{p}}) \cdot \mathbf{e}_+\| - \rho \;=\; 0. \tag{6.9}$$

The main problem in this formulation is the square root term of the norm containing the unknowns in quadratic terms since $\|\boldsymbol{x}\| = \sqrt{\sum(x_i)^2}$. This leads to equations of the type

$$\sqrt{((\boldsymbol{M}\underline{\boldsymbol{p}}\widetilde{\boldsymbol{M}} \underline{\times} \underline{\boldsymbol{L}}') \cdot \mathbf{e}_+)^2} - \rho \;=\; 0. \tag{6.10}$$

I made experiments with these kind of equations and implemented a Newton-Raphson method to solve the equations. But there are two main problems: Firstly, the convergence rate is very slow and the algorithm often converges against a wrong minimum (the algorithm needs about 5 seconds to estimate the pose). Secondly, the possibility is lost to combine the constraints with the other constraints for simultaneous considerations in pose estimation.

The key idea to relate circles and spheres to lines is to interpret the circles and spheres as orbits generated by twist operations as introduced in the next section.

### 6.2.2 Operational definition of circles and spheres using twists

At first the general description of circles and spheres will be recapitulated, as introduced in chapter 3.3. Then an operational definition of these entities is introduced. After this follows the formalization of the circle-line and sphere-line constraints for pose estimation.

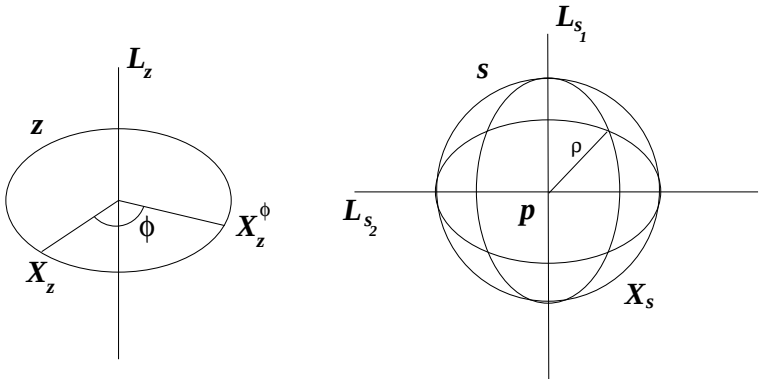



Fig. 6.4: Circles and spheres parameterized with twists.

Let $\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$ be a circle in CGA. Evaluating the outer products of three points leads to

$$\underline{\boldsymbol{z}}^\star \;=\; \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} = A + A^- \mathbf{e} + A^+ \mathbf{e}_0 + A^\pm \boldsymbol{E}, \tag{6.11}$$

with suitable multivectors $A$, $A^-$, $A^+$ and $A^\pm$, see chapter 3.3.

A circle can also be formalized by a twist (modeling a general rotation) $\underline{\boldsymbol{L}}_z$ and a point $\underline{\boldsymbol{X}}_z$ on the circle. From the dual representation of the circle, this information is easy to extract since the generating twist parameters are given directly. The twist transformation corresponds to a suitable parameterized motor $\boldsymbol{M}_\phi$,

$$\boldsymbol{M}_\phi \;=\; \exp\left(\frac{\phi}{2}\left(A^+ + \mathbf{e}A^\pm\right)\right). \tag{6.12}$$

The points on the circle are simply given by

$$\underline{\boldsymbol{X}}_z^\phi \;=\; (\boldsymbol{M}_\phi \underline{\boldsymbol{X}}_z \widetilde{\boldsymbol{M}}_\phi) \quad : \quad \phi \in [0, \ldots, 2\pi]. \tag{6.13}$$

Figure 6.4 (left) shows the geometry. The circle results as the orbit of the unique motor, which moves a certain point and is constrained by the points $\underline{\boldsymbol{a}}$, $\underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{c}}$ laying on the circle.

Now follows the formalization of a sphere. The general expression of a sphere is

$$\underline{\boldsymbol{s}} \;=\; (\underline{\boldsymbol{p}} - \frac{1}{2}\rho^2\mathbf{e}). \tag{6.14}$$

In this formulation $\underline{\boldsymbol{p}}$ is the center of the sphere and $\rho$ the radius.

The idea is to formalize spheres in an operational manner as two coupled twists (modeling general rotations). In that approach a sphere is formalized by a point $\underline{\boldsymbol{X}}_s$ on a sphere and two perpendicular twists, $\underline{\boldsymbol{L}}_{s_1}$ and $\underline{\boldsymbol{L}}_{s_2}$, intersecting in the origin of the sphere. Figure 6.4 (right) shows the idea. The corresponding motors are denoted as $\boldsymbol{M}_{\phi_1}$ and $\boldsymbol{M}_{\phi_2}$,

$$\begin{aligned} \boldsymbol{M}_{\phi_1} &= \exp\left(\frac{\phi_1}{2}\left(\mathbf{e}_{12} + \mathbf{e}(\boldsymbol{p}\cdot\mathbf{e}_{12})\right)\right), \\ \boldsymbol{M}_{\phi_2} &= \exp\left(\frac{\phi_2}{2}\left(\mathbf{e}_{31} + \mathbf{e}(\boldsymbol{p}\cdot\mathbf{e}_{31})\right)\right). \end{aligned} \tag{6.15}$$

The bivectors $\mathbf{e}_{12}$ and $\mathbf{e}_{31}$ are the two perpendicular rotation planes, belonging to the rotation axes which are connected to the center $\underline{\boldsymbol{p}}$ of the sphere. Then all points on the sphere $\underline{\boldsymbol{s}}$ result from the equation

$$\underline{\boldsymbol{X}}_s^{\phi_1,\phi_2} \;=\; (\boldsymbol{M}_{\phi_1}\boldsymbol{M}_{\phi_2}\underline{\boldsymbol{X}}_s\widetilde{\boldsymbol{M}}_{\phi_2}\widetilde{\boldsymbol{M}}_{\phi_1}) \quad : \quad \phi_1, \phi_2 \in [0, \ldots, 2\pi]. \tag{6.16}$$

This principle of coupling two motors is virtual in contrast to kinematic chains, which correspond to the coupling of physical objects. The principle of virtual coupling can be extended further to construct more complex orbits of twists and thus to enable pose estimation of more complex objects.

### 6.2.3 The constraint equations of circles and spheres to lines

So far the formalization of circles and spheres as orbits of special twists is given. This representation will be used to express tangentiality of circles $\underline{\boldsymbol{z}}$ and spheres $\underline{\boldsymbol{s}}$ to 3D lines $\underline{\boldsymbol{L}}$.

While in the constraint equations of section 5 the motors are the only unknowns to be estimated, now higher loads are given because of the parameterization of the extended entities for pose estimation.

To relate the circle $\underline{\boldsymbol{z}}$ to a line $\underline{\boldsymbol{L}} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})$, there is need to estimate the unknown angle $\phi$, which leads to collinearity of the suitable transformed point $\underline{\boldsymbol{X}}_z \in \underline{\boldsymbol{z}}$ to $\underline{\boldsymbol{L}}$. The circle-line constraint can now be written as

$$(\boldsymbol{M}_\phi \underline{\boldsymbol{X}}_z \widetilde{\boldsymbol{M}}_\phi) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;\; = \;\; 0. \tag{6.17}$$

In this equation, the angle $\phi$ is an additional unknown for each constraint equation. The pose estimation constraint equation for an unknown rigid body motion now means to estimate both the best motor $\boldsymbol{M}$ and the angle $\phi$,

$$(\boldsymbol{M}(\boldsymbol{M}_\phi \underline{\boldsymbol{X}}_z \widetilde{\boldsymbol{M}}_\phi)\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;\; = \;\; 0. \tag{6.18}$$

Note, that relating a circle to a projection ray can be seen as estimating a virtual one-parameter kinematic chain.

The sphere-line constraint, respectively expressing the incidence of a line $\underline{\boldsymbol{L}} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})$ to a sphere $\underline{\boldsymbol{s}}$ can be formalized by a point $\underline{\boldsymbol{X}}_s$ on the sphere and the two motors $\boldsymbol{M}_{\phi_1}$ and $\boldsymbol{M}_{\phi_2}$,

$$(\boldsymbol{M}_{\phi_1} \boldsymbol{M}_{\phi_2} \underline{\boldsymbol{X}}_s \widetilde{\boldsymbol{M}}_{\phi_2} \widetilde{\boldsymbol{M}}_{\phi_1}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;\; = \;\; 0. \tag{6.19}$$

In this constraint equation $\phi_1$ and $\phi_2$ are additional unknowns. The pose estimation constraint equation for an unknown rigid body motion means to estimate the best motor $\boldsymbol{M}$ and the two angles $\phi_1$ and $\phi_2$,

$$(\boldsymbol{M}(\boldsymbol{M}_{\phi_1} \boldsymbol{M}_{\phi_2} \underline{\boldsymbol{X}}_s \widetilde{\boldsymbol{M}}_{\phi_2} \widetilde{\boldsymbol{M}}_{\phi_1})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;\; = \;\; 0. \tag{6.20}$$

This approach to formalize constraint equations for circles and spheres appears surprising in the context of the algebraic embedding. The main problem with these entities is to formalize constraint equations which obtain the characteristics mentioned in chapter 2.3. For this reason I choose an operational definition of circles and spheres and linearize them in the same manner as the pose problem is linearized: The entities are formulated in their tangential space and a Lie algebra representation of these entities is chosen.

## 6.3 Cycloidal curves

Though points, lines, planes, kinematic chains, circles and spheres cover a large range to model objects, I am also interested in e.g. ellipses to model the shape of eyes, etc. This motivates to search for a more general class of

entities which can be used in the context of pose estimation. The entities I am now dealing with are *cycloidal curves*. Such curves can be generated by one, two or more *virtually coupled twists*. Examples are 3D circles, ellipses, cardioids, cycloids, spheres, quadrics, etc. This chapter will start with a general introduction into algebraic curves and will then continue with the use of twists to model cycloidal curves and how to embed them within the 2D-3D pose estimation problem.

### 6.3.1 Algebraic curves

This section gives a brief summary of algebraic curves [108]. There exist many ways to define algebraic curves. They can be defined as parametric or algebraic implicit forms or polynomial equations [33]. For example an ellipse can be defined as the set of intersection points of two projectively related pencils of lines [81]. It is also possible to define an ellipse as intersection of a cone with a plane. Parametric, cartesian or polar equations of a curve lead to quite different representations. E.g. the parametric equation of a plane cardioid is

$$(x, y) \;=\; (a(2\cos(t) - \cos(2t)), a(\sin(t) - \sin(2t))\,, \tag{6.21}$$

a cartesian equation is

$$\left(x^2 + y^2 - 2ax\right)^2 \;=\; 4a^2(x^2 + y^2) \tag{6.22}$$

and the polar equation is

$$r \;=\; 2a\,(1 + \cos(\theta))\,. \tag{6.23}$$

The resulting question is: Which representation of an algebraic curve is well suited within the pose estimation problem?

The aim is to define a curve in the 3D space and since the pose estimation algorithm uses twists to model rigid body motions, I prefer the description of algebraic curves as orbits of generating operators of the Lie algebra $se(3)$.

This section gives a brief summary of algebraic curves and their characterizations collected by Lee in [108]. More detailed information about algebraic curves can also be found in [38]. A historic introduction can be found in [52]. In this thesis, a subclass of the *roulettes*, the *cycloidal curves*, are formalized and embedded in the 2D-3D pose problem. Roughly speaking, cycloidal curves are generated from circles rolling on circles or lines.

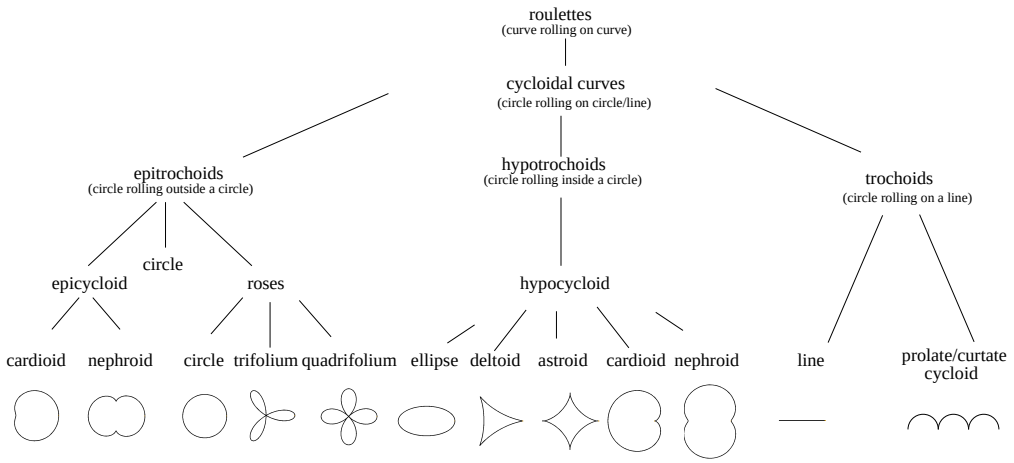



Fig. 6.5: Subtree of algebraic curves.

Figure 6.5 shows a subtree of the family of algebraic curves. Cycloidal curves can be classified as epitrochoids, hypotrochoids and trochoids, which split to other subclasses. Figure 6.5 also shows examples of these curves. The curves are distinguished by the relative position of the circles with respect to the starting point on the curve and the radius of the circles. Now it will be continued to explain some curves in more detail:

1. A **cardioid** can be defined as the trace of a point on a circle that rolls around a fixed circle of the same size without slipping. Cardioids are *double generated*, which means that a cardioid is both, an epicycloid and a hypocycloid, since it can be generated in two different ways. Cardioids were e.g. studied by Roemer (1674).

2. A **nephroid** can be defined as the trace of a point fixed on a circle of radius $\frac{1}{2}r$ that rolls around a fixed circle with radius $r$. Nephroids are also double generated. They were studied by Huygens and Tschirnhausen about 1679.

3. A **rose** is defined as a curve of epi/hypocycloids with respect to its center. The curve has loops that are symmetrically distributed around the *pole*. The loops are called petals or leafs. They were studied by Guido Grandi around 1723.

4. An **ellipse** is commonly defined as the locus of points $P$ such that the sum of the distances from $P$ to two fixed points $F_1$, $F_2$ (called foci) are constant. The ellipses seem to have been discovered by Menaechmus (a Greek, c.375-325 BC), tutor to Alexander the Great.

5. A **deltoid** can be defined as the trace of a point on a circle, rolling inside another circle 3 or $\frac{3}{2}$ times as large in radius. Deltoids were studied by Euler in 1745 in connection with a study of caustics curves.

6. An **astroid** is defined as the trace of a point on a circle of radius $r$ rolling inside a fixed circle of radius $4r$ or $\frac{4}{3}r$. The cycloidal curves, including the astroid, were also discovered by Roemer (1674).

7. A **trochoid** is defined as the trace of a point fixed on a circle that rolls along a line. This curve is sometimes called the *trace of a bike valve*.

These curves are mostly defined in the 2D plane. For the scenario of pose estimation, these curves will be extended to plane curves in the 3D space.

### 6.3.2 Cycloidal curves in conformal geometric algebra

As previously explained, cycloidal curves are circles rolling on circles or lines. In this section will be explained how to generate such curves in conformal geometric algebra. E.g. ellipses are no entities which can be directly de-

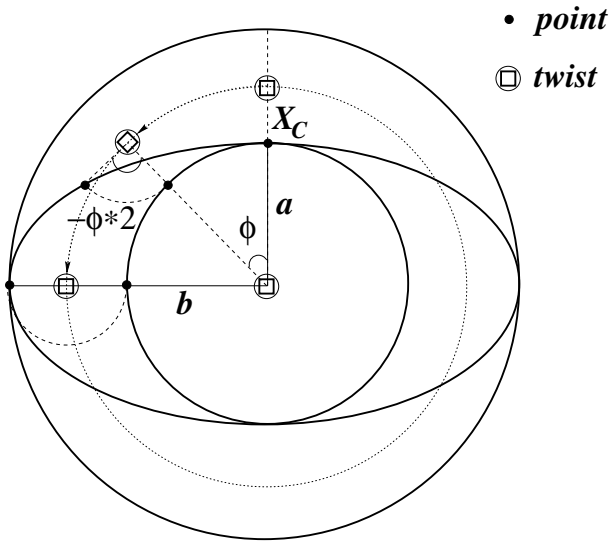



Fig. 6.6: An ellipse generated by two coupled twists.

scribed in conformal geometric algebra. The idea for modeling ellipses is visualized in figure 6.6: Two parallel twists (modeling general rotations) in the 3D space and a 3D point on the ellipse are assumed and the point is transformed around the two twists in a fixed and dependent manner. To

gain an ellipse, two coupled parallel (not collinear) twists are used to rotate a starting point by $-2\phi$ around the first twist and by $\phi$ around the second one. The set of all points for $\phi \in [0, \ldots, 2\pi]$ generates an ellipse as the orbit of the generated Lie group.

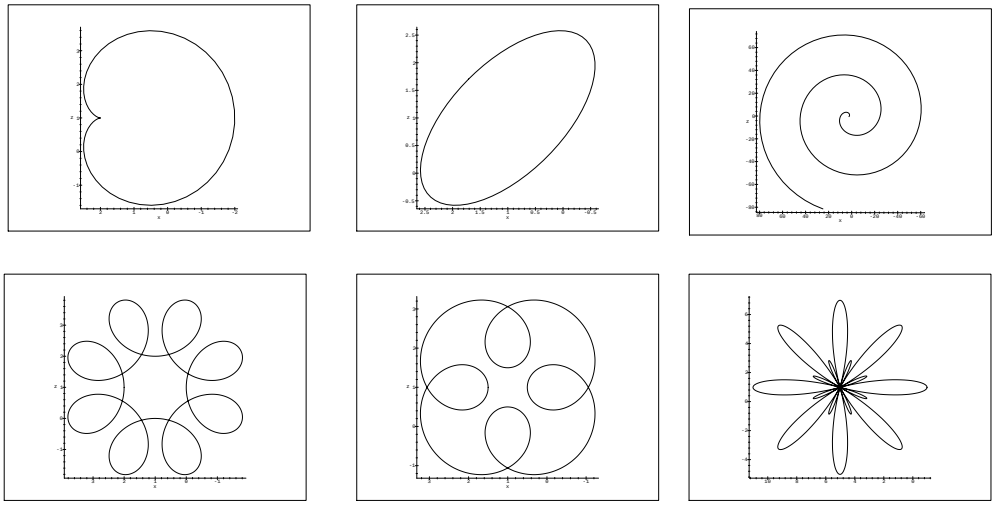


Fig. 6.7: 3D-2twist generated curves.

In general, every cycloidal curve is generated by a set of twists $\xi_i$ with frequencies $\lambda_i$ acting on one point $\underline{\boldsymbol{X}}$ on the curve. Since twists can be used to model point motions in the 2D plane or 3D space, I call the generated curves $n$D-$m$twist curves. With $n$D-$m$twist curves are meant curves given in the $n$ dimensional space, generated by $m$ twists with $n, m \in \mathbb{N}$. In the context of the 2D-3D pose estimation problem the cycloidal curves are used as 3D object entities. Furthermore 3D-$m$twist curves are abbreviated as $m$twist curves in the following.

To formalize this idea of generating curves in the CGA, I will start with easy curves. The easiest one consists of one point (a point on the curve) and one twist modeling a general rotation. Rotating the point around the twist leads to the parameterized generation of a circle: The corresponding twist transformation can be expressed as a suitable motor $\boldsymbol{M}_\phi$ and an arbitrary 3D point, $\underline{\boldsymbol{X}}_Z$, on the circle. The 3D orbit of all points on the circle is simply given by

$$\underline{\boldsymbol{X}}_Z^\phi \;=\; \boldsymbol{M}_\phi \underline{\boldsymbol{X}}_Z \widetilde{\boldsymbol{M}}_\phi \quad : \quad \phi \in [0, \ldots, 2\pi]. \tag{6.24}$$

I call a circle also a *1twist* generated curve since it is generated by one twist. This is already done in chapter 6.2.2.

This can be extended by wrapping a second twist around the first one. If the amount of rotation of each twist is dependent on each other, a more general 3D curve is obtained. This curve is firstly dependent on the relative positions and orientation of the twists with respect to each other, the (starting) point on the curve and the ratio of angular frequencies.

The form of a *2twist* generated curve is

$$\begin{aligned}\underline{\boldsymbol{X}}_C^{\phi} &= \boldsymbol{M}^2_{\lambda_2\phi}\boldsymbol{M}^1_{\lambda_1\phi}\underline{\boldsymbol{X}}_C\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi} \\ &= \exp\left(-\frac{\lambda_2\phi}{2}\Psi_2\right)\exp\left(-\frac{\lambda_1\phi}{2}\Psi_1\right)\underline{\boldsymbol{X}}_C\exp\left(\frac{\lambda_1\phi}{2}\Psi_1\right)\exp\left(\frac{\lambda_2\phi}{2}\Psi_2\right) \\ &\quad : \quad \lambda_1,\lambda_2 \in \mathbb{R},\ \ \phi \in [\alpha_1,\ldots,\alpha_2]. \quad (6.25)\end{aligned}$$

The motors $\boldsymbol{M}^i$ are the exponentials of twists, the scalars $\lambda_i \in \mathbb{R}$ determine the ratio of angular frequencies between the twists and $\underline{\boldsymbol{X}}_C$ is a point on the curve. The values $\alpha_i$ define an interval for the boundaries of the curve. Indeed it is also possible to define curve segments.

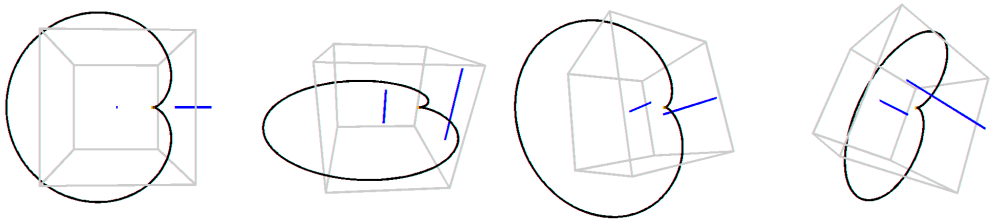


Fig. 6.8: Perspective views of a 3D-2twist generated curve. The 2twist curve and the twists axes are visualized.

The case of parallel twists modeling general rotations with $\lambda_1 = \lambda_2 = 1$ leads to cardioids, $\lambda_1 = 2, \lambda_2 = 1$ leads to nephroids, and $\lambda_1 = 3, \lambda_2 = 1$ leads to deltoids, which can be transformed (by moving the second twist) to a trifolium, etc. Ellipses can be generated with $\lambda_1 = -2, \lambda_2 = 1$. Figure 6.7 shows further examples from curves, which can be generated easily by two coupled twists. Note: Also the archimedic spiral is a 2twist generated curve. To gain an archimedic spiral, one twist has to be a translator.

Note: All these curves are given in the 3D space. In figure 6.7 only projections are shown. Figure 6.8 shows different projective views of a 3D-twist generated curve.

It is also possible to generate 3twist curves. For this, a point is transformed with respect to three twists. Examples of planar curves are shown in figure 6.9.

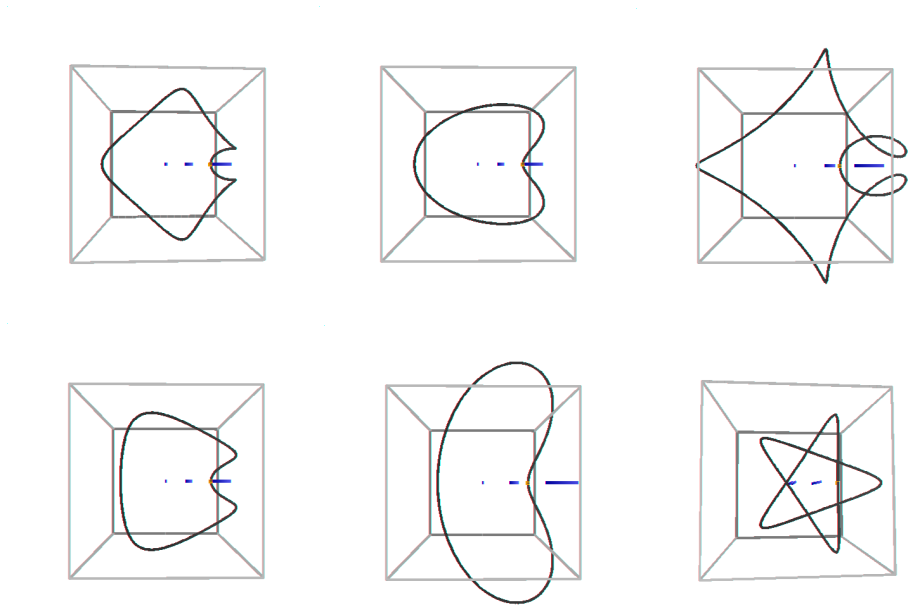


Fig. 6.9: 3D-3twist generated curves.

So far just 3D curves are formalized. Surfaces can be modeled, by rotating e.g. the 2twist generated curves around a third twist with a further independent angle $\phi_2$, leading to 3twist generated surfaces. Examples are shown in figure 6.10. In figure 6.10 no grid-plot is shown. Instead the rotated twist generated curves are shown to visualize the geometric generation of the surface. Note, that if there is only one variable angle $\phi$, the resulting entity is a 3D curve in the 3D space, whereas the case of two variable angles $\phi_1$ and $\phi_2$ leads (for non-degenerate cases) to a 3D surface in the 3D space. The case of three variable angles can lead to volumes as highest structures in the 3D space, but for the 2D-3D pose estimation scenario only curves and surfaces will be modeled.

The general form of 3twist generated surfaces is

$$\underline{\boldsymbol{X}}^{\phi_1,\phi_2} = \boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2}, \tag{6.26}$$

with $\lambda_i \in \mathbb{R}$ and $\phi_1, \phi_2 \in [\alpha_{i_1}, \ldots, \alpha_{i_2}]$.

An ellipsoid, for example, is nothing more than a rotated ellipse ($\lambda_3 = \lambda_2 = 1, \lambda_1 = -2$). Its parameterized equation can be written as

$$\underline{\boldsymbol{X}}_Q^{\phi_1,\phi_2} = \boldsymbol{M}^3_{\phi_2}\boldsymbol{M}^2_{\phi_1}\boldsymbol{M}^1_{-2\phi_1}\underline{\boldsymbol{X}}_Q\widetilde{\boldsymbol{M}}^1_{-2\phi_1}\widetilde{\boldsymbol{M}}^2_{\phi_1}\widetilde{\boldsymbol{M}}^3_{\phi_2} \quad : \quad \phi_1, \phi_2 \in [0, \ldots, 2\pi]. \tag{6.27}$$

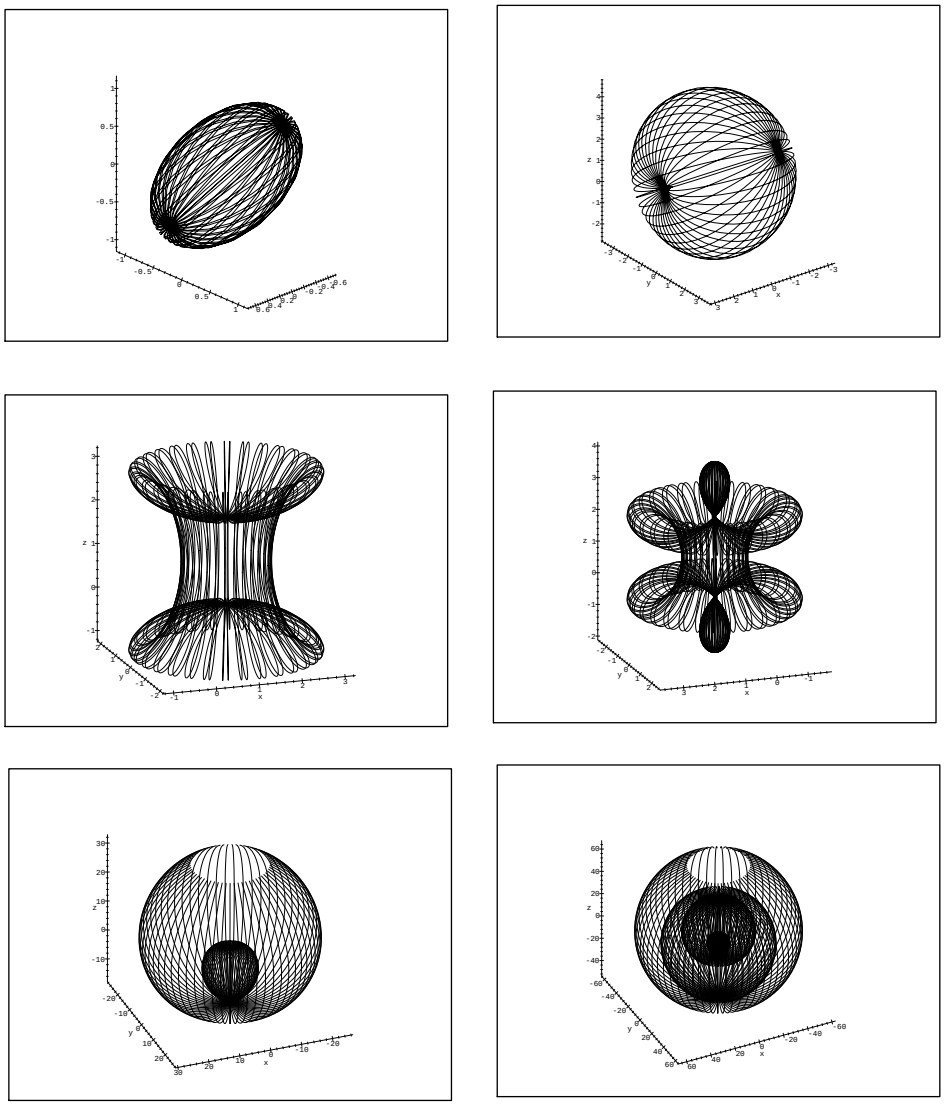


Fig. 6.10: 3D-3twist generated surfaces.

This is visualized in the first image of figure 6.10. The second image of figure 6.10 shows a horizontally rotated cardioid. The surface is comparable to a ball with a pen pressed inside the ball. The third and fourth images in the second row show rotated hypocycloids. The last row shows rotated spirals, leading to surfaces comparable to a flower. These surfaces are very easy to generate and can be represented by just a few coupled twists. Note, that also lines and planes are falling in the definition of 2twist and 3twist generated curves and surfaces. For these cases the twists contain translational

| Entity | Class | Entity | Class |
|---|---|---|---|
| point | 0twist curve | rose | 2twist curve |
| circle | 1twist curve | spiral | 2twist curve |
| line | 1twist curve | sphere | 2twist surface |
| ellipse | 2twist curve | plane | 2twist surface |
| line segment | 2twist curve | cone | 2twist surface |
| cardioid | 2twist curve | cylinder | 2twist surface |
| nephroid | 2twist curve | quadric | 3twist surface |

Tab. 6.1: Well known 3D entities as $m$twist curves or surfaces.

parts. Table 6.1 gives an overview of some well known entities interpreted as cycloidal curves.

Note, that the rigid body motions of these entities can easily be estimated, just by transforming the generating twists. The transformation of an $m$twist generated curve can be performed by transforming the $m$ twists, and the point on the curve. The description of these curves is compact and rigid transformations can be estimated very fast.

Cycloidal curves and surfaces extend already studied entities to a more general class of entities, without loosing the advantages of the previous work, see chapter 5, 6.1 and 6.2.3. Indeed, it is possible to build up a hierarchy of entities and the next chapter will concentrate on enlarging these kinds of entities to approximate free-form contours. So far the hierarchy of entities consists of the following entities:

$$\begin{array}{ccccccc} \text{points, lines} & \subseteq & \text{circles} & \subseteq & \text{cycloidal curves} & \subseteq & \dots \\ \text{planes} & \subseteq & \text{spheres} & \subseteq & \text{cycloidal surfaces} & \subseteq & \dots \end{array}$$

### 6.3.3 Ray-tracing on cycloidal curves

The aim of this part is to show that the twist representation of cycloidal curves can also be used to model optical properties on the curves or surfaces. This is interesting for ray-tracing tasks or computer graphics applications.

Cycloidal curves have the useful property that they can be derivated at (nearly) all positions. To build the foundations for ray-tracing on cycloidal curves, the *envelope* of a family of curves is defined as:

**Definition 6.1** *The envelope or discriminant of the family $F$, $F : \mathbb{R}\times\mathbb{R}^r \rightarrow$*

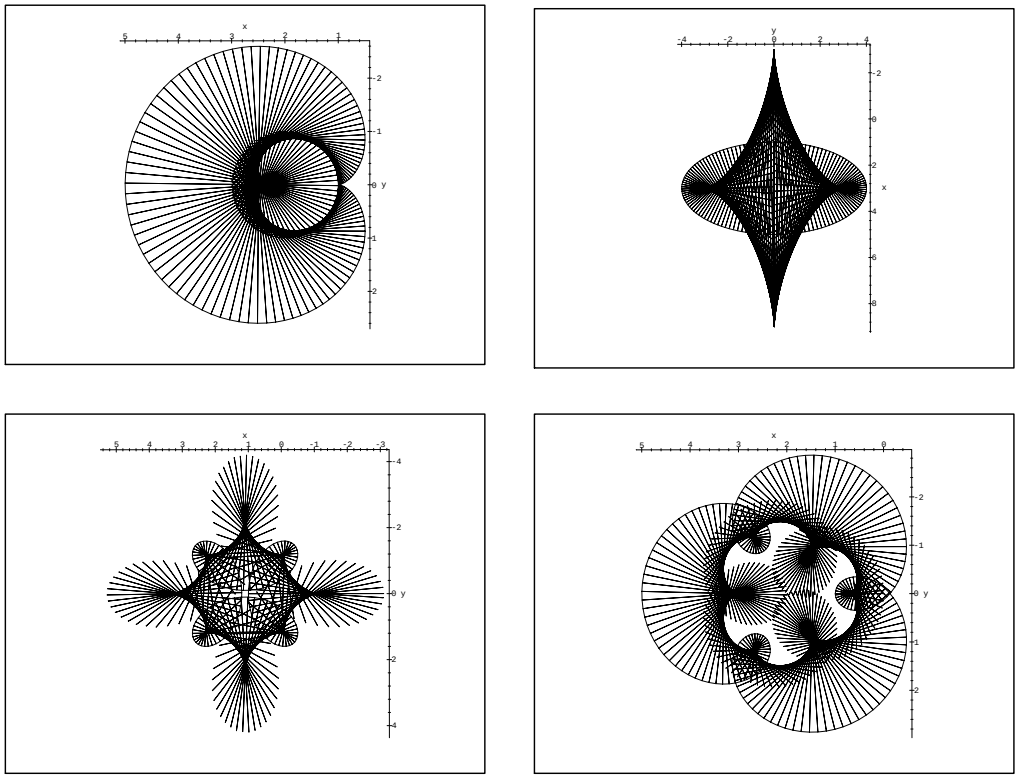


Fig. 6.11: Evolutes of cycloidal curves.

$\mathbb{R}, r \in \mathbb{N}$ *is the set*

$$\mathcal{D} = \mathcal{D}_F \quad = \quad \{x \in \mathbb{R}^r | \exists t \in \mathbb{R} : F(t,x) = \frac{\partial F}{\partial t} F(t,x) = 0\}. \qquad (6.28)$$

Building the envelope means to derive a new curve based on a set of curves. The envelope of a set of curves is a curve $C$ such that $C$ is tangent to every member of the set. Two curves are tangent to each other, if both curves share a common tangent at a common point. Now it is possible to define the *evolute*:

**Definition 6.2** *The evolute is the envelope of the normals to a given curve.*

Figure 6.11 shows examples. The normals of four cycloidal curves (a cardioid, an ellipse, a rose and cycloid) are drawn. These normals define a new curve, which is tangent to each normal of the initial curve. This curve directly *pops* out to the human eye and is not specially drawn. Since cycloidal

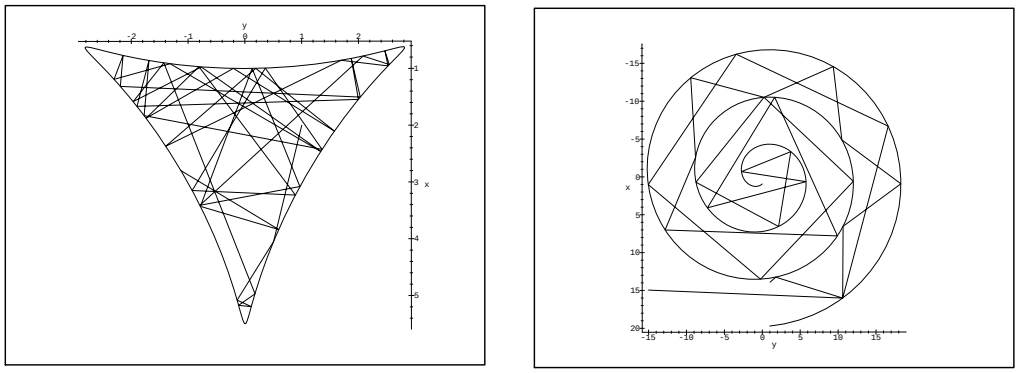


Fig. 6.12: Ray-tracing examples of a ray reflecting inside a deltoid and a ray reflecting in and out of a spiral.

curves can easily be derivated in CGA, it is also easy to estimate reflections of rays with respect to the curve normals. This property is also used by A. Lasenby et al. in [107], but in their work no twist approach is used to model the curves. In section 3.1.2 the derivation of rotors with respect to angles is already introduced. It can directly be applied on the twist representation of motors. Furthermore the product rule can be used to estimate the derivative of a point on a cycloidal curve. Here it will be demonstrated on a 2twist generated curve:

$$\frac{\partial \underline{\boldsymbol{X}}_C^{\phi}}{\partial \phi} = \frac{\partial \left( \boldsymbol{M}^2_{\lambda_2 \phi} \boldsymbol{M}^1_{\lambda_1 \phi} \underline{\boldsymbol{X}}_C \widetilde{\boldsymbol{M}}^1_{\lambda_1 \phi} \widetilde{\boldsymbol{M}}^2_{\lambda_2 \phi} \right)}{\partial \phi} \tag{6.29}$$

$$\begin{aligned} = & \left( \frac{\partial \boldsymbol{M}^2_{\lambda_2 \phi}}{\partial \phi} \boldsymbol{M}^1_{\lambda_1 \phi} \underline{\boldsymbol{X}}_C \widetilde{\boldsymbol{M}}^1_{\lambda_1 \phi} \widetilde{\boldsymbol{M}}^2_{\lambda_2 \phi} \right) + \\ & \left( \boldsymbol{M}^2_{\lambda_2 \phi} \frac{\partial \boldsymbol{M}^1_{\lambda_1 \phi}}{\partial \phi} \underline{\boldsymbol{X}}_C \widetilde{\boldsymbol{M}}^1_{\lambda_1 \phi} \widetilde{\boldsymbol{M}}^2_{\lambda_2 \phi} \right) + \\ & \left( \boldsymbol{M}^2_{\lambda_2 \phi} \boldsymbol{M}^1_{\lambda_1 \phi} \underline{\boldsymbol{X}}_C \frac{\partial \widetilde{\boldsymbol{M}}^1_{\lambda_1 \phi}}{\partial \phi} \widetilde{\boldsymbol{M}}^2_{\lambda_2 \phi} \right) + \\ & \left( \boldsymbol{M}^2_{\lambda_2 \phi} \boldsymbol{M}^1_{\lambda_1 \phi} \underline{\boldsymbol{X}}_C \widetilde{\boldsymbol{M}}^1_{\lambda_1 \phi} \frac{\partial \widetilde{\boldsymbol{M}}^2_{\lambda_2 \phi}}{\partial \phi} \right). \end{aligned} \tag{6.30}$$

Following chapter 3, a tangent $\boldsymbol{t}$ of a point can be rotated by $90^o$ degree to get a normal direction $\boldsymbol{n}$ and the reflection rule can be applied,

$$\underline{\boldsymbol{x}}' = \boldsymbol{n} \underline{\boldsymbol{x}} \boldsymbol{n}, \tag{6.31}$$

to estimate the reflected point. Figure 6.12 shows examples of a light ray reflecting inside a deltoid on the left and a light ray reflecting in and out of a spiral on the right.

### 6.3.4 Constraint equations for pose estimation of cycloidal curves

This section deals with the generation of constraint equations for pose estimation of cycloidal curves. The fusion of cycloidal curves and the pose estimation problem is easy now, since every involved topic is formalized in the conformal geometric algebra. It is just necessary to substitute the formalization of cycloidal curves within the collinearity or coplanarity constraints of chapter 5.4: The constraint equations of pose estimation from image points read

$$\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x})}_{\substack{\text{projection ray,}\\ \text{reconstructed from the image point}}}}_{\substack{\text{collinearity of the transformed object}\\ \text{point with the reconstructed line}}} = 0. \qquad (6.32)$$

The 3D cycloidal curve is assumed as e.g.

$$\underline{\boldsymbol{X}}_Z^{\phi} = \boldsymbol{M}^2_{\lambda_1\phi}\boldsymbol{M}^1_{\lambda_2\phi}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_2\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_1\phi} \quad : \quad \lambda_1,\lambda_2\in\mathbb{R},\ \ \phi\in[0,\ldots,2\pi].$$

The rigid motion of this curve incident to a projection ray can be expressed as

$$\left(\boldsymbol{M}(\boldsymbol{M}^2_{\lambda_1\phi}\boldsymbol{M}^1_{\lambda_2\phi}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_2\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_1\phi})\widetilde{\boldsymbol{M}}\right)\ \underline{\times}\ (\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x})) = 0. \quad (6.33)$$

Since every aspect of the 2D-3D pose estimation problem of cycloidal curves is formalized in CGA, the constraint equation describing the pose problem is compact and easy to interpret: The inner parenthesis contains the parameterized generation of the cycloidal curve with one unknown angle $\phi$. The outer parenthesis contains the unknown motor $\boldsymbol{M}$, describing the rigid body motion of the 3D cycloidal curve. This is the unknown pose. The expression is then combined via the commutator product with the reconstructed projection ray and has to be zero. This describes the cotangentiality of the transformed curve to a projection ray.

The unknowns are the six parameters of the rigid motion $\boldsymbol{M}$ and the angle $\phi$ for each point correspondence.

In a similar way it is possible to formalize constraint equations for incidence of cycloidal surfaces to projection rays,

$$\left(\boldsymbol{M}(\boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2})\widetilde{\boldsymbol{M}}\right) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \;=\; 0. \tag{6.34}$$

But this would not cover all geometric aspects of the surface in the pose problem. It is more efficient to build constraints on the surface contour in the image and to model also tangentiality within the constraints. Therefore the surface tangential plane $\underline{\boldsymbol{P}_{\boldsymbol{x}}}$ of each point $\underline{\boldsymbol{X}}$ can be used to claim incidence of the tangential plane $\underline{\boldsymbol{P}_{\boldsymbol{x}}}$ with each projection ray,

$$\left(\boldsymbol{M}(\boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{P}_{\boldsymbol{x}}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2})\widetilde{\boldsymbol{M}}\right) \overline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \;=\; 0. \tag{6.35}$$

This leads to additional orientation constraints, which can be used in the context of the 2D-3D pose estimation problem. The unknowns of these constraint equations are the rigid motion $\boldsymbol{M}$ and the angles $\phi_1$ and $\phi_2$.

## 6.4 Experiments for pose estimation of extended object concepts

This section presents experimental results of pose estimation with extended object concepts. The experiments will start with estimating the pose of kinematic chains. Then it will be continued with experiments on pose estimation of circles, spheres and cycloidal curves.

### 6.4.1 Pose estimation experiments of kinematic chains

The following experiments visualize the application of the pose estimation algorithm or kinematic chains on different scenarios, see figures 6.13-6.19. In the first image sequence, the object model is a door in a cupboard and both the angle of the door and the position of the robot observing the door are changing. During these movements the correspondences are extracted manually and the transformed and projected object is visualized in the sequence. It is easy to see that both unknowns, the pose of the cupboard and the angle of the door, are estimated and the error is small.

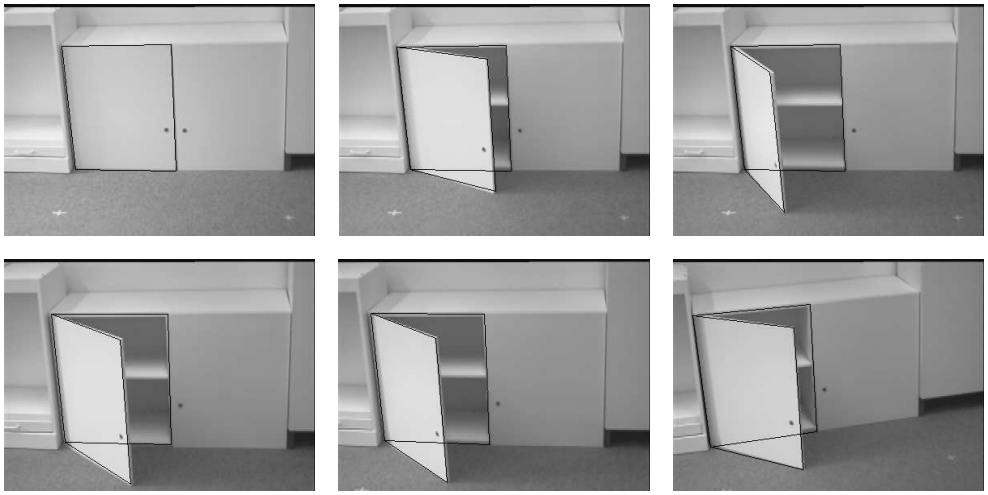


Fig. 6.13: Images of the first real scenario. Both the pose of the cupboard and the opening angle of the door are estimated.

In the second image sequence, the object model is a doll and estimated are the pose, the angle of the upper arms and of the forearms. Figure 6.14 visualizes the transformed and projected object in the sequence. Though only one 3D point for each kinematic chain segment is used and the size of the doll is measured by hand, the pose is also accurate.

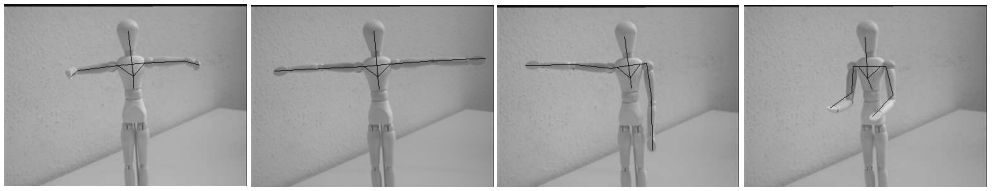


Fig. 6.14: Images of the second scenario. The pose of the doll and the angles of the arms are estimated.

The next images (see e.g. figure 6.15) present results of O. Granert [66], who made experiments with the RX-90 robot arm [169] and implemented a VRML-viewer, parser and other software package for kinematic chains. I supervised his work in the context of a student project.

Figure 6.15 shows some examples of a sequence containing 42 images. In this image sequence the first joint is moving in 5 degree steps from 0 to 25 degree. Then the second joint is moving in 5 degree steps from 0 to 60 degree. This is also shown in figure 6.16. The pose of the robot and the angles of the kinematic chain are estimated. The image features are tracked

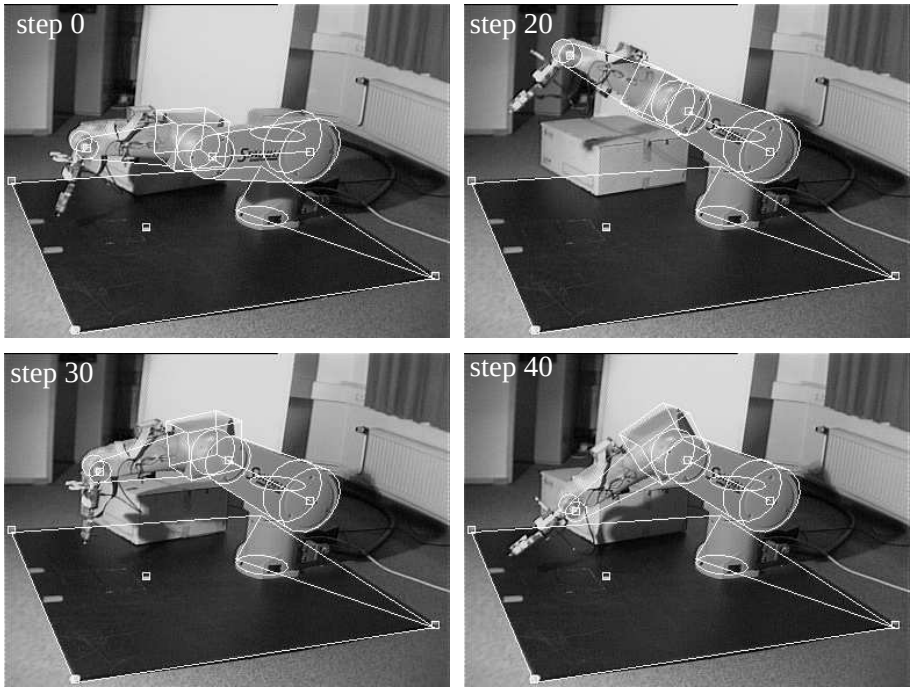



Fig. 6.15: Images of a tracked robot arm taken from a sequence with 40 images.

point markers. Figure 6.16 shows the joint angles estimated and overlaid with the ground truth[1]. Small deviations can be recognized. Dependent on the position of the camera with respect to the object model and the location of the joints, the estimated angles differ around 0.5 to 3 degrees to the ground truth. In simulation environments (and ideal situations) it is clear, that (for non degenerate cases) the parameters during the iterations converge against the ground truth. The errors in these experiments are dependent on the calibration quality, the lens distortions and the accuracy of the color marker detection.

Figure 6.17 shows images of another image sequence. There the stability of the algorithm is visualized in the context of moved color markers: During tracking the robot, a student moves into the scenario, picks up a color marker and moves it around. This leads to outliers in the scenario and therefore to impossible kinematics of the robot. Two things can be seen. Firstly, the geometry of the robot is modeled within the constraints and the model will not be distorted. Instead, the algorithm leads to a spatially best fit of the

[1] Since the positioning accuracy of the robot arm is very good, the positioning values of the robot arm are used as ground truth.

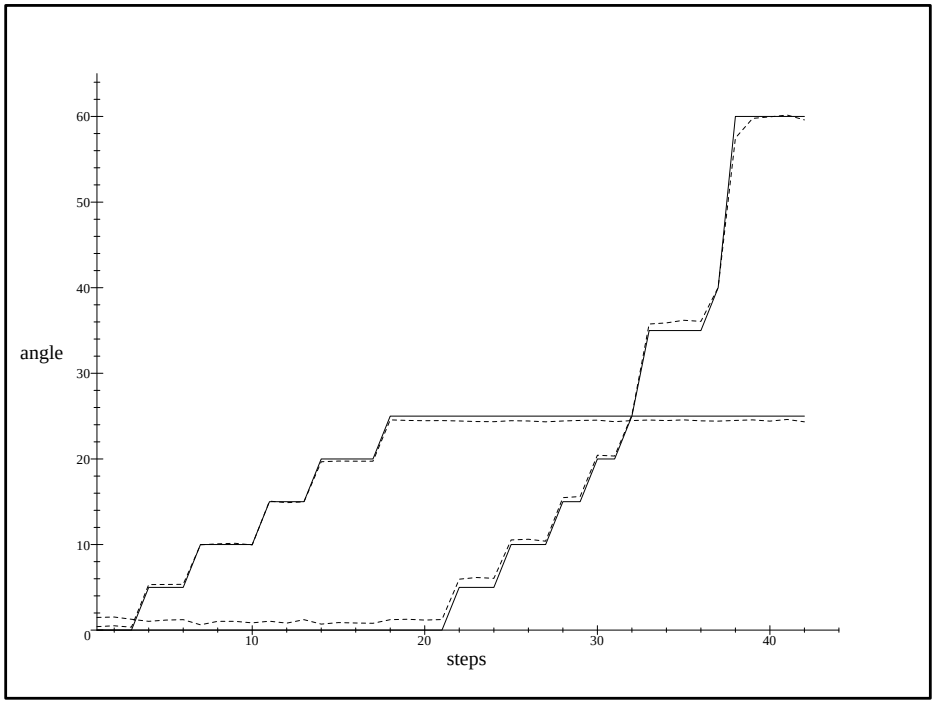



Fig. 6.16: Joint angles estimated and overlaid with the ground truth. The solid lines show the ground truth and the dashed lines show the estimated values.

model to the extracted image data. Secondly, no hierarchical approach for pose estimation of piecewise rigid objects as mentioned in chapter 6.1 is used. Instead a pose estimation based on the model of a kinematic chain. There are two main arguments why I do not recommend a hierarchical approach for pose estimation: Firstly, the geometry of the whole object is not modeled within the constraint equations. That necessitates the second processing step to ensure a non distorted model. This second processing step can be avoided by modeling a kinematic chain within the constraint equations, as is done in this work or by [25]. Secondly, each point of a kinematic chain contributes two linearly independent equations. Also the higher order points of a kinematic chain influence the result of the whole pose. This is strongly wanted in this context because only then all possible geometric information is used simultaneously and is not neglected due to redundancy of the algorithm. The main result of this experiment is to visualize the robustness of the developed algorithm.

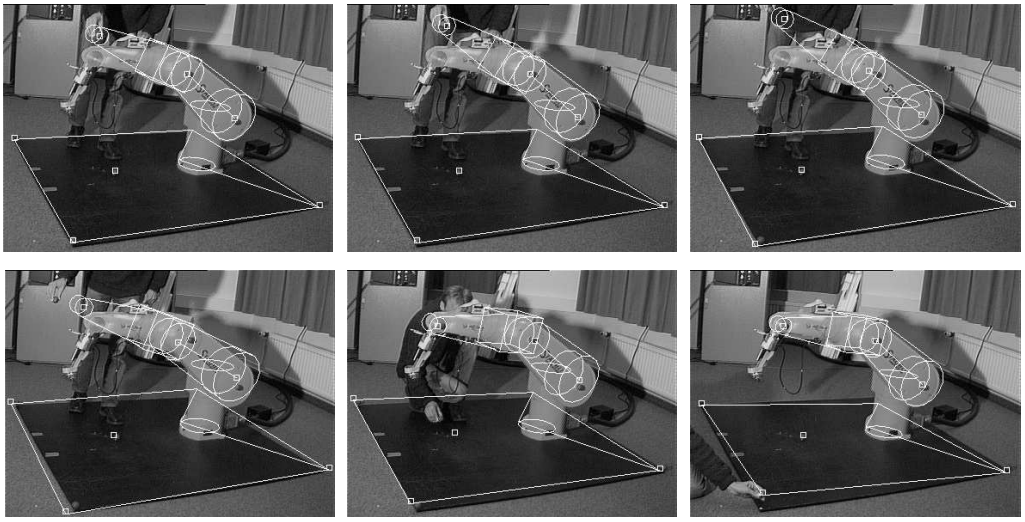


Fig. 6.17: Stability example for distorted color markers and visualization that the geometry of the robot is modeled within the constraints.

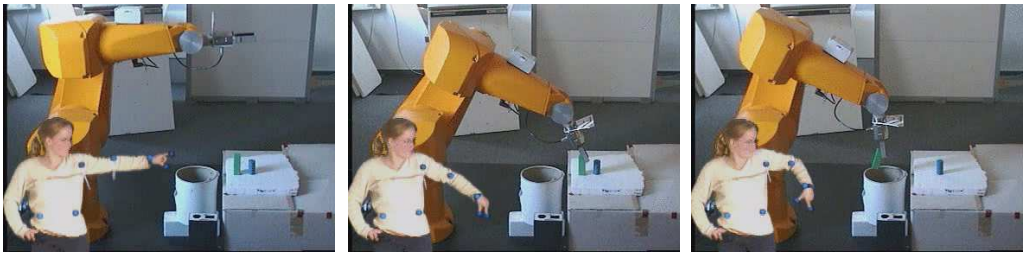


Fig. 6.18: Example images for visual remote controlling of the robot.

Figure 6.18 shows three example images taken from the student project [160] I supervised. The students O. Schmitz and J. Koberstein had the task to implement a *visual remote control* for the robot. This means, a person with color markers attached to its body has to be tracked and the angles of the human arm's movements have to be estimated and translated to the robot kinematics. So the robot arm follows the human arm movements and therefore the human is able to control the robot through its own movements. Color markers on the finger tips indicate the opening and closing of the gripper. This module is used for grasping and manipulation tasks as shown in figure 6.18. Since the students also implemented a client-server package, image processing and tracking of the person is independent from the location of the robot. This allows for a robot controlling though the person is not in the same room in which the robot is located. During the experiments, the students let the image stream run via the university campus, which is 3km

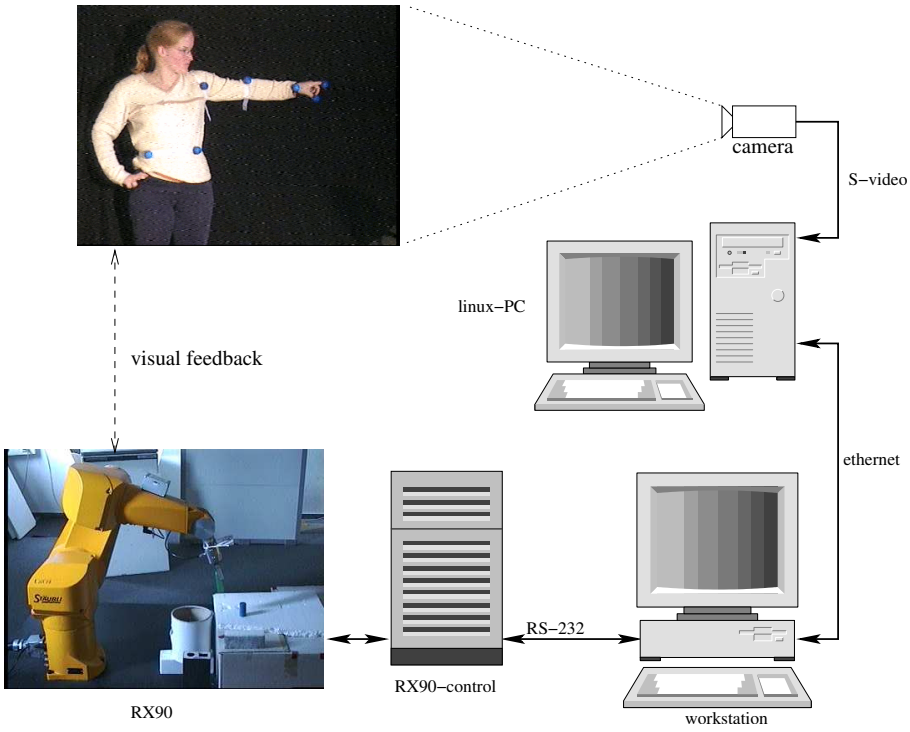



Fig. 6.19: The system design for controlling the robot.

away from the robot lab. The control system is shown in figure 6.19.

### 6.4.2 Pose estimation experiments with circles and spheres

This section presents the use of more extended object concepts for pose estimation. In figure 6.20 pose estimation results of an object containing points, lines, kinematic chains and circles are shown.

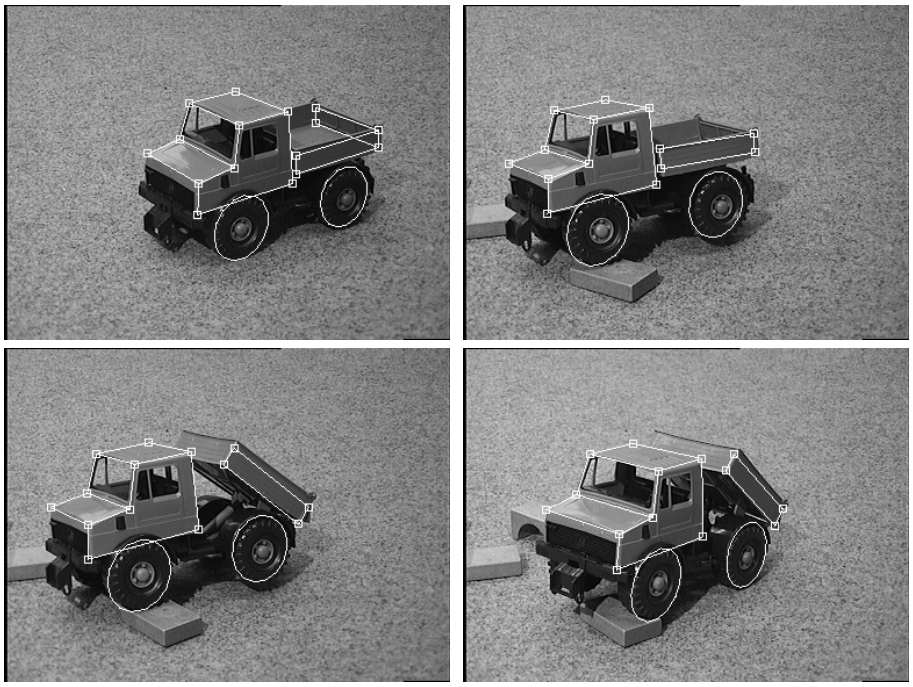


Fig. 6.20: Pose estimation of an object, consisting of 3D points, lines, circles and kinematic chain segments.

In another experiment an object model is used which contains additionally a sphere, a prismatic and a revolute joint. All available object information is used simultaneously for pose estimation. The model and its object features are depicted in figure 6.21. Figure 6.22 shows some pose estimation results of the object model. Though the size of the model is measured by hand, the pose is accurate and also the joint parameters are well approximated. All information is arranged in one linear system of equations, which leads to simultaneous solving of the pose parameters by using all different features.

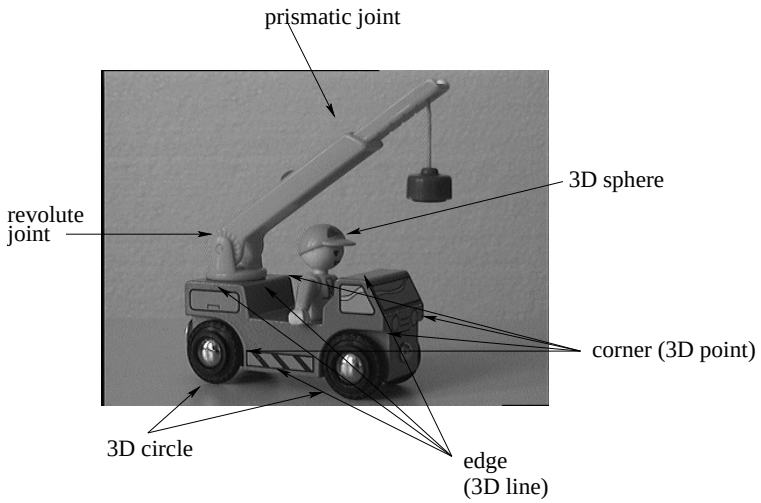



Fig. 6.21: Object model, consisting of different entities.

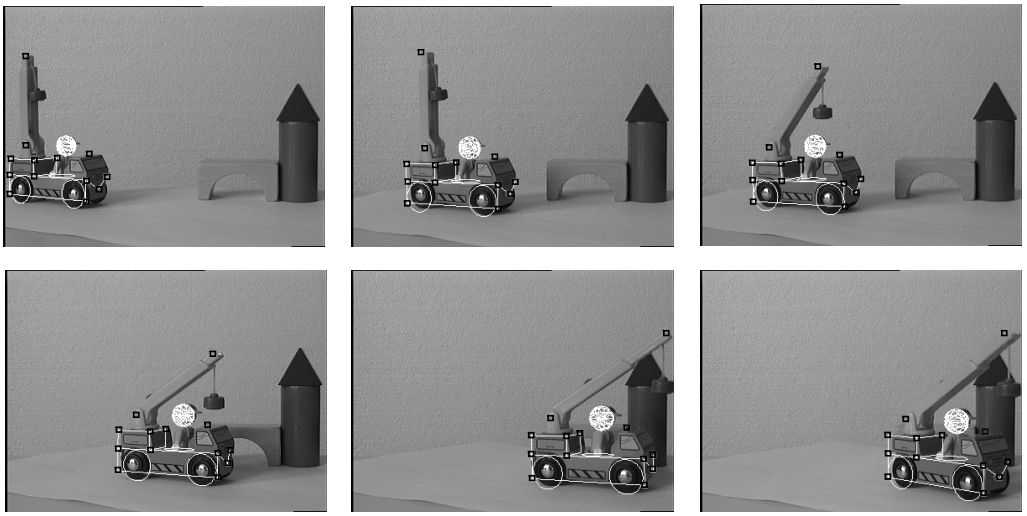


Fig. 6.22: Pose estimation by using all types of model features.

### 6.4.3 Pose estimation experiments with cycloidal curves

The main problem of pose estimation of cycloidal curves is that they are in general not convex. This results in the problem of getting trapped in local minima. Figure 6.23 visualizes the problem and shows the distance function of a line to a parameterized spiral. In this example, the line is perpendicular to the spiral. The resulting error function of the points on the spiral with respect to the 3D line is visualized in the right image of figure 6.23. These kinds of error functions are very hard to solve and for example local search

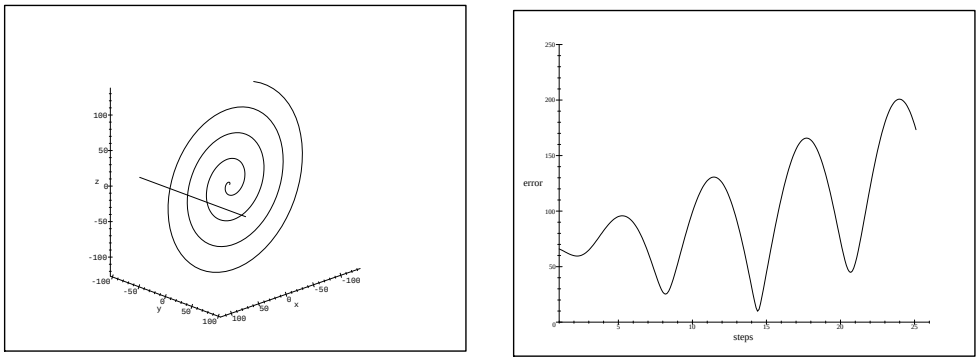



Fig. 6.23: Distance function of a line to a spiral.

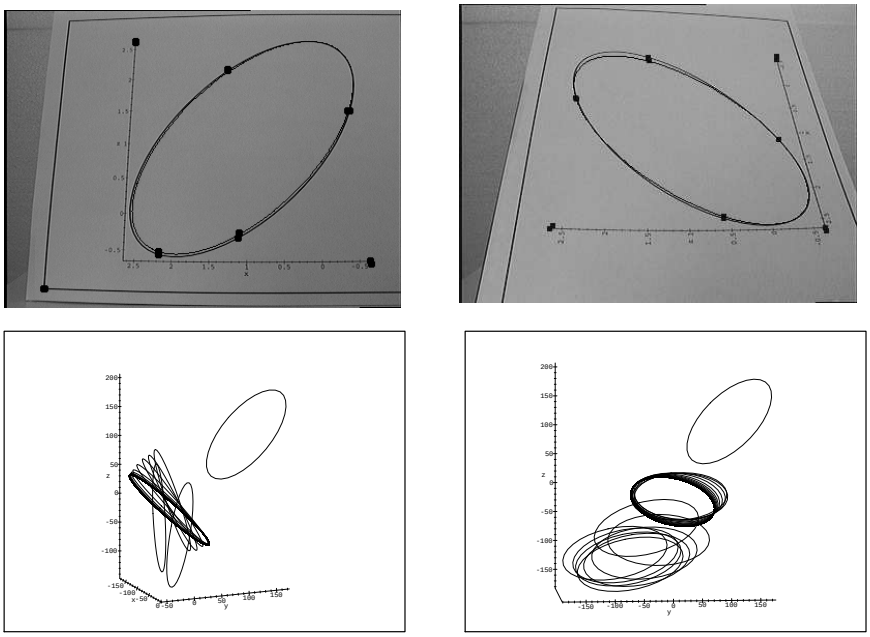


Fig. 6.24: Pose estimation of a 3D ellipse and the convergence behavior.

strategies have to be combined with heuristics to handle these problems and to find a global minimum [18].

The next experiments deal with pose estimation results of convex objects. For these examples the gradient method converges directly.

Figure 6.24 shows the pose estimation result of a 3D ellipse. For this an ellipse is printed on a paper sheet and images are taken of this paper sheet.

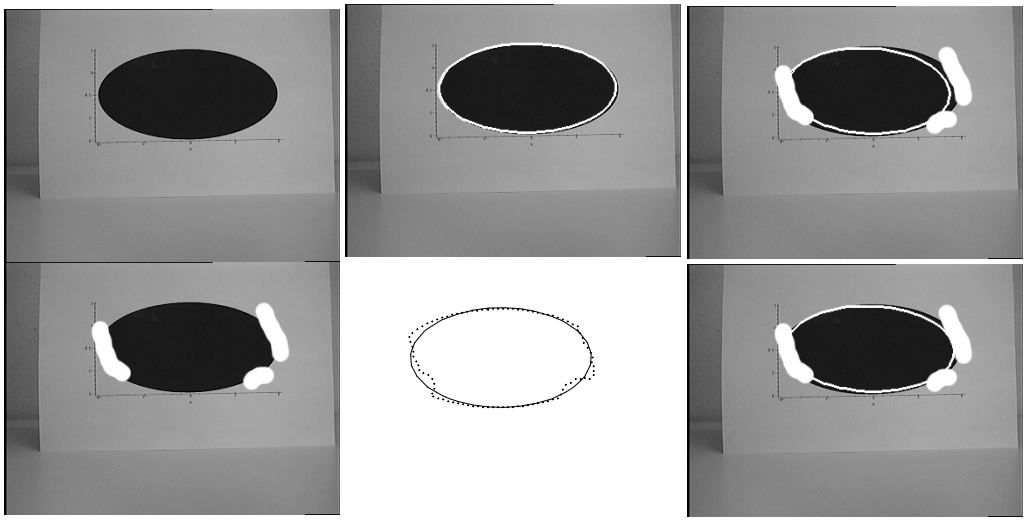


Fig. 6.25: Pose estimation of a 3D ellipse by using undistorted, distorted and interpolated data.

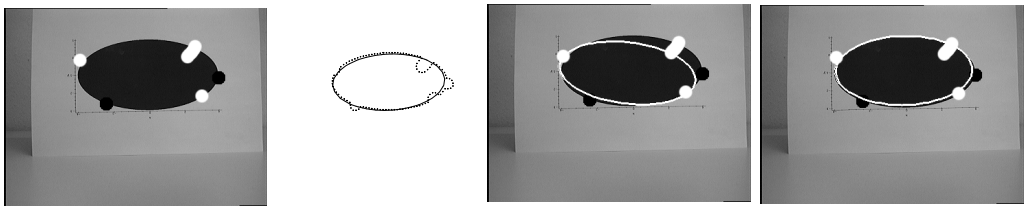


Fig. 6.26: Pose estimation of a 3D ellipse by using distorted and interpolated data.

In the first row, two images are shown with marked extracted point features in the image and the transformed projected reference points. Furthermore, the transformed projected ellipse is shown. While the correspondences between the point features are very well fitted, some parts of the curve are not perfectly fitted to the image curve. This occurs because of the few information used to estimate the location of the ellipse. The second row of figure 6.24 shows the convergence behavior of the ellipse during the iterations.

Figures 6.25 and 6.26 address the self-regularizing properties of image contours: In the first column of figure 6.25 one undistorted and one distorted image ellipse is shown. The second column shows a pose estimation result by taking the undistorted image data and the extracted contour of the distorted image. There are two possibilities to deal with the contour points: On the

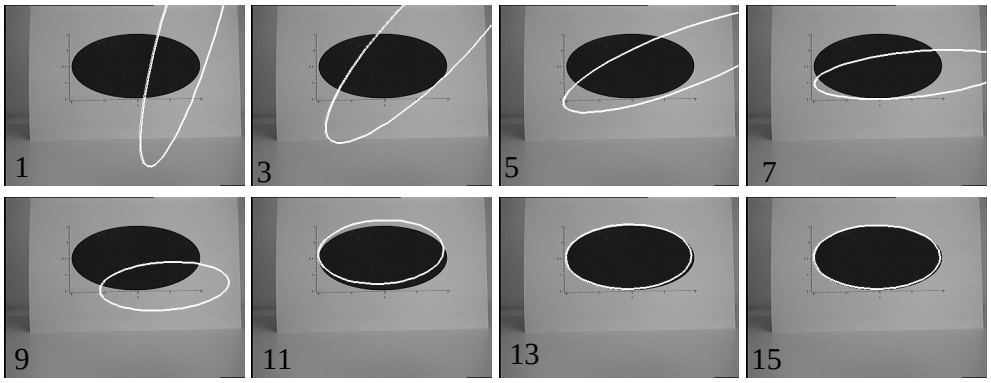



Fig. 6.27: Convergence behavior of the algorithm during the iterations.

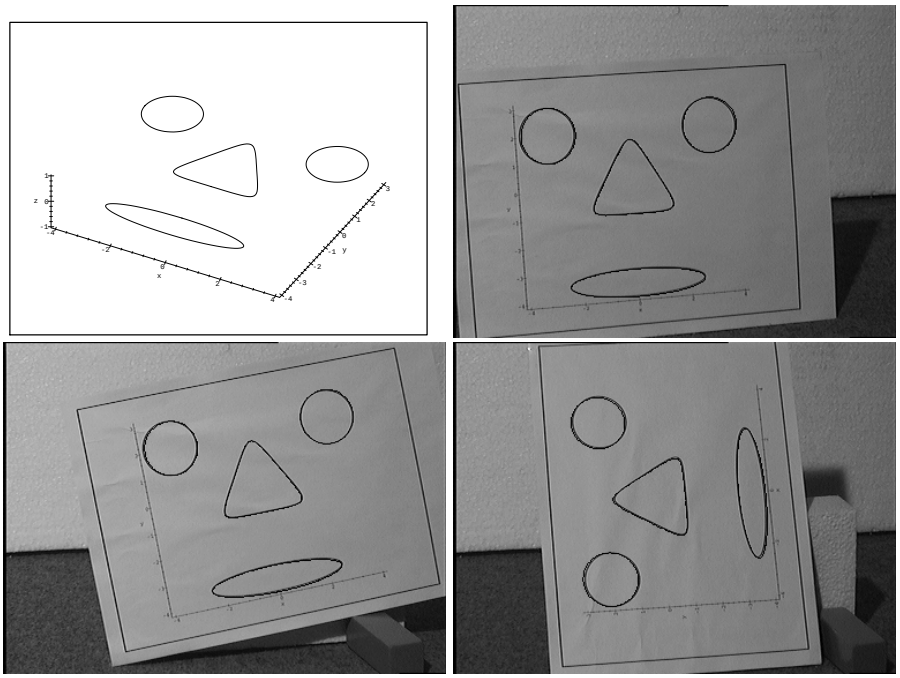


Fig. 6.28: Pose estimation of an object containing one ellipse, two circles and one deltoid.

one hand it is possible to use them directly, or on the other hand it is possible to interpolate them to an image ellipse and use the interpolated data. In [191] several approaches for conic fitting are presented. To interpolate the contour points, the LLS (linear least squares) approach has been re-implemented and used. Though the LLS approach is not the best algorithm discussed in [191], it is easy to implement and fast. The pose results for the raw contour points on the one hand and the interpolated ellipse points on the other hand

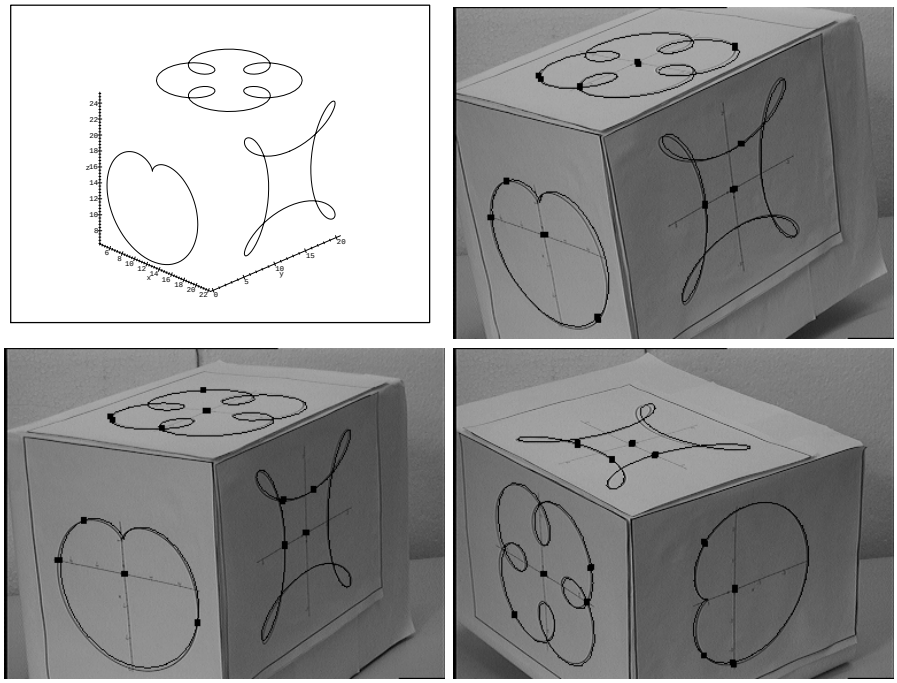


Fig. 6.29: Pose estimation of an object, containing a cardioid and two cycloids.

are shown in the third column of figure 6.25. The upper image shows the pose result achieved by using the pure points, the lower image shows the result achieved by using the interpolated data. Indeed, using the pure points leads to worse results than using the interpolated data. Figure 6.26 shows a more extreme case of distorted image data. It can be seen that the use of interpolated image data leads to more stable results than using the raw contour points.

Figure 6.27 shows the convergence behavior of the algorithm during the iterations. Since the rigid body motion which is to be estimated is very large, the algorithm needs several iteration steps to converge. If only small movements are observed, the number of iterations (and therefore the computing time) can be reduced significantly.

Figure 6.28 shows pose estimation results of a second 3D object model. This object contains two circles, one ellipse and one deltoid. The left image shows the 3D object model. The other images show the transformed projected object model to visualize the quality of the pose.

Now it will be continued with pose estimation of non-convex objects. Fig-

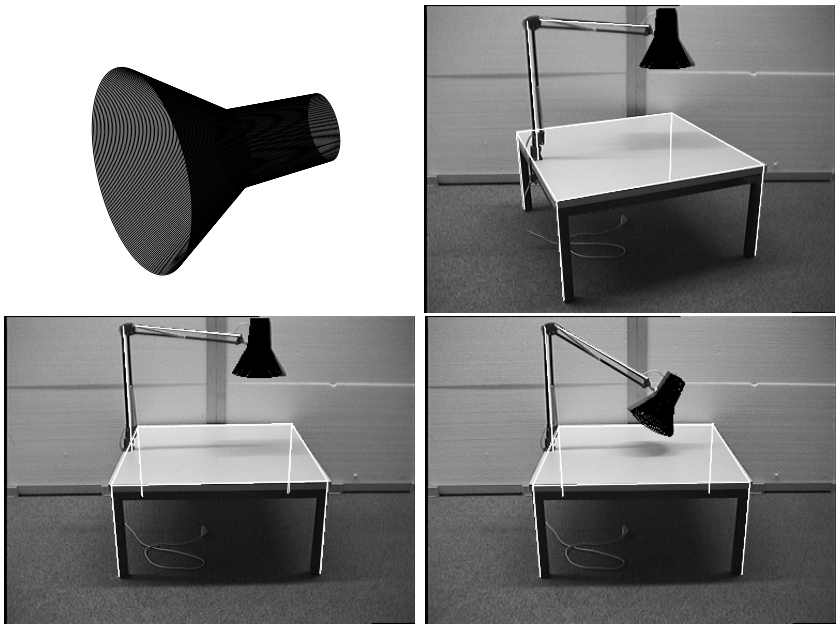


Fig. 6.30: Pose estimation of a 2twist surface, connected via a 2 d.o.f. kinematic chain to a table.

ure 6.29 shows experimental results of an object, containing three cycloidal non-convex curves. All information is used simultaneously to solve the pose parameters. Here the gradient method for pose estimation is encapsulated within a heuristic, since the entities are not convex any more. In the first image, the used object model is shown. The other three images show pose estimation results of the object. Furthermore, the transformed projected cycloidal curves are shown. Since the size of the object model is measured by hand, the pose estimation result is quite accurate.

The combination of the gradient method with the mentioned heuristic leads to slow algorithms. While the pose for convex object models can be estimated in real-time, the estimation of the object, presented in figure 6.29 takes up to 5 minutes to converge to a global minimum. But indeed it is possible to estimate the global minimum. The reason for this long computing time comes along with the fact, that e.g. no tracking assumption is made for the scenario. Such additional assumptions can fasten the algorithm enormously and this is done in chapter 7 to gain real-time performance.

Figure 6.30 shows experimental results of a 2twist surface. The object

model is a lamp-shade connected via a 2 d.o.f.[2] kinematic chain to a table. In this experiment, the pose parameters and the angles of the kinematic chain are estimated. The lamp-shade itself is modeled by two cone parts. Since both surface parts are convex, the gradient method converges directly and there is no need to apply a heuristic to estimate the pose.

The cycloidal curves are the intermediate stage to the next step of generalization, which are general free-form contours. In the experimental part of the next chapter also statistical experiments are presented.

[2] degree of freedom

## Chapter 7

# POSE ESTIMATION OF FREE-FORM OBJECTS

So far it has been discussed how to use a set of coupled twists to generate a 3D curve. Now the reverse problem is addressed: Assume a set of 3D points on a 3D contour. The question is, which $n$twist generated curve interpolates the 3D points in a suitable way. This problem is closely related to Fourier descriptors which are usually defined in the 2D plane and are often used for object recognition [67, 186, 6, 93] and affine 2D pose estimation [7, 138]. The aim of this section is to clarify the relation between twist-generated curves and Fourier descriptors. The 3D Fourier descriptors are then used to model object contours within the perspective 2D-3D pose estimation problem. The key idea is, that a set of coupled twists modeling general rotations is equivalent to a sum over a set of rotors, which act on different phase vectors. This can be interpreted as a Fourier series expansion, leading to a trigonometric interpolation of a set of contour points.

A short introduction into the concepts of Fourier transformation in classical matrix calculus is given in chapter A.4. Instead, in this chapter everything will be derived in CGA to show how elegantly it is possible to combine results of signal theory within the pose problem by using CGA as algebraic language. To introduce the description of free-form contours, Ch. Perwass' idea in [146] will be followed, who firstly formalized Fourier descriptors in the conformal geometric algebra. Therefore the relation of a Fourier series expansion to coupled twists in the 3D Euclidean space is shown at first, since only rotations of different phase vectors around the origin are needed. For pose estimation the expressions are then transformed into the conformal space.

Let

$$\boldsymbol{R}_i^{\phi} \;:=\; \exp(-\frac{\pi u_i \phi}{T}\,\boldsymbol{l}), \tag{7.1}$$

where $T \in \mathbb{R}$ is the length of the closed curve, $u_i \in \mathbb{Z}$ is a frequency number and $\boldsymbol{l}$ is a unit bivector which defines the rotation plane. Furthermore holds

$$\widetilde{\boldsymbol{R}}_i^{\phi} \;=\; \exp(\frac{\pi u_i \phi}{T}\,\boldsymbol{l}). \tag{7.2}$$

Recall that $\boldsymbol{l}^2 = -1$ and, as noted in equations (3.35) and (3.38), therefore it is possible to write the exponential function as

$$\exp(\phi\,\boldsymbol{l}) = \cos(\phi) + \sin(\phi)\,\boldsymbol{l}. \tag{7.3}$$

A 2twist generated curve may then be written in Euclidean space as follows,

$$\begin{aligned}
\underline{\boldsymbol{X}}_C^{\phi} &= \boldsymbol{M}^2_{\lambda_2\phi}\boldsymbol{M}^1_{\lambda_1\phi}\underline{\boldsymbol{X}}_C\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi} \\
\Leftrightarrow \boldsymbol{x}_C^{\phi} &= \boldsymbol{R}_2^{\phi}\left(\left(\boldsymbol{R}_1^{\phi}\left(\boldsymbol{x}_C - \boldsymbol{t}_1\right)\widetilde{\boldsymbol{R}}_1^{\phi} + \boldsymbol{t}_1\right) - \boldsymbol{t}_2\right)\widetilde{\boldsymbol{R}}_2^{\phi} + \boldsymbol{t}_2 \\
&= \boldsymbol{R}_2^{\phi}\boldsymbol{R}_1^{\phi}(\boldsymbol{x}_C - \boldsymbol{t}_1)\widetilde{\boldsymbol{R}}_1^{\phi}\widetilde{\boldsymbol{R}}_2^{\phi} + \boldsymbol{R}_2^{\phi}(\boldsymbol{t}_1 - \boldsymbol{t}_2)\widetilde{\boldsymbol{R}}_2^{\phi} + \boldsymbol{t}_2 \\
&= \boldsymbol{p}_0 + \boldsymbol{V}_1^{\phi}\,\boldsymbol{p}_1\,\widetilde{\boldsymbol{V}}_1^{\phi} + \boldsymbol{V}_2^{\phi}\,\boldsymbol{p}_2\,\widetilde{\boldsymbol{V}}_2^{\phi},
\end{aligned} \tag{7.4}$$

where $\boldsymbol{p}_0 \equiv \boldsymbol{t}_2$, $\boldsymbol{p}_1 \equiv \boldsymbol{t}_1 - \boldsymbol{t}_2$, $\boldsymbol{p}_2 \equiv \boldsymbol{x}_C - \boldsymbol{t}_1$, $\boldsymbol{V}_1^{\phi} \equiv \boldsymbol{R}_2^{\phi}$, $\boldsymbol{V}_2^{\phi} \equiv \boldsymbol{R}_2^{\phi}\boldsymbol{R}_1^{\phi}$ and $\lambda_i = 2\pi u_i/T$. Note that for plane curves the rotors $\boldsymbol{R}_1^{\phi}$ and $\boldsymbol{R}_2^{\phi}$ act in the same plane and the vectors $\boldsymbol{x}_C$, $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ lie in the rotation plane. Hence, the $\{\boldsymbol{p}_i\}$ lie in the rotation plane.

It can be shown that if a vector $\boldsymbol{x}$ lies in the rotation plane of some rotor $\boldsymbol{R}$, then $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{x}\widetilde{\boldsymbol{R}}$. The previous equation can therefore be written as

$$\boldsymbol{x}_C^{\phi} \;=\; \boldsymbol{p}_0 + \boldsymbol{p}_1\,\widetilde{\boldsymbol{V}}_1^{2\phi} + \boldsymbol{p}_2\,\widetilde{\boldsymbol{V}}_2^{2\phi}. \tag{7.5}$$

Note that the square of a rotor is equal to a rotor of twice the angle in the same rotation plane. Therefore, $\widetilde{\boldsymbol{V}}_i^{\phi}\widetilde{\boldsymbol{V}}_i^{\phi} = \widetilde{\boldsymbol{V}}_i^{2\phi}$. Using the exponential form of rotors, equation (7.5) becomes

$$\boldsymbol{x}_C^{\phi} \;=\; \boldsymbol{p}_0 + \boldsymbol{p}_1\,\exp\left(\frac{2\pi u_1\phi}{T}\,\boldsymbol{l}\right) + \boldsymbol{p}_2\,\exp\left(\frac{2\pi u_2\phi}{T}\,\boldsymbol{l}\right). \tag{7.6}$$

This is equivalent to a Fourier series expansion where the imaginary unit $i = \sqrt{-1}$ is replaced with $\boldsymbol{l}$ and the complex Fourier series coefficients with vectors that lie in the plane spanned by $\boldsymbol{l}$. The latter vectors are the phase

vectors. In general it may be shown that any closed, plane curve $C(\phi)$ can be expressed as a series expansion

$$C(\phi) = \lim_{N\to\infty} \sum_{k=-N}^{N} \boldsymbol{p}_k \exp\left(\frac{2\pi k\phi}{T}\boldsymbol{l}\right) = \lim_{N\to\infty} \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi}. \qquad (7.7)$$

For every closed plane curve there is a unique set of phase vectors $\{\boldsymbol{p}_k\}$ that parameterizes the curve. However, such a set corresponds to infinitely many different combinations of coupled twists. That is, given a set of coupled twists, it is possible to obtain the corresponding phase vectors $\{\boldsymbol{p}_k\}$ but not vice versa. The spectral representation of a curve transforms the translational parts of its generating twists into a set of different phase vectors and therefore results in a pure rotor description.

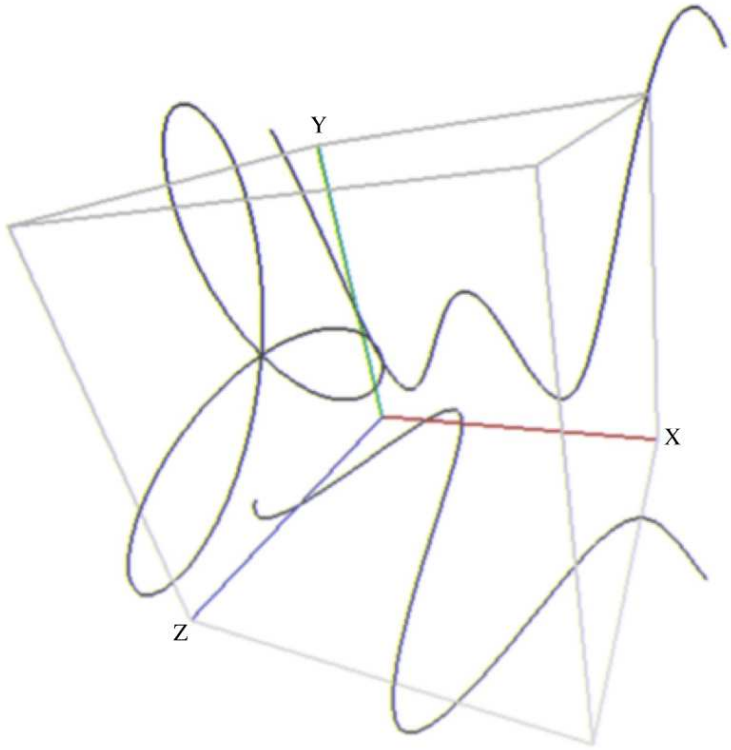



Fig. 7.1: Projections of a curve created by coupled twists.

The expansion in equation (7.7) is again closely related to the standard Fourier series expansion of a real, scalar valued function. In figure 7.1 a closed curve created by two coupled twists is shown in the $yz$-plane. Suppose that

instead of $C(\phi)$, the function $C_S(\phi) := C(\phi) + 2\pi\phi/T\,\boldsymbol{e}_1$ is considered, where $\boldsymbol{e}_1$ is the unit vector along the $x$-axis. If $C_S(\phi)$ is projected onto the $xy$-plane and $xz$-plane, the two other shown curves are obtained. This visualizes the well known fact that any periodic function in a space of dimension $n$ can be regarded as the projection of a closed curve in a space of dimension $n+1$.

The phase vectors $\{\boldsymbol{p}_k\}$ are also called *Fourier descriptors*. It has long been known that one can also construct affine invariant Fourier descriptors [67, 4], that is, entities that describe a closed curve and stay invariant under affine transformations of the curve. This is particularly useful for object recognition and has been used in many applications [6, 57, 173]. The same relations that allow one to construct affine invariant Fourier descriptors also allow for affine pose estimation. This works in the following way: Consider a closed curve that lies on a plane which is tilted with respect to an observer. This curve is projected with an affine camera onto an image plane. The pose of the plane in space can then be estimated, if the Fourier descriptors of the projected curve as well as the Fourier descriptors of the original curve are given. See [5] for more details. A projective pose estimation via Fourier descriptors does not exist so far. The problems occurring there are discussed in the appendix of [146].

To represent a general closed, discretizied 3D curve this can easily be extended to 3D by interpreting the projections along $x$, $y$, and $z$ as three infinite 1D-signals and applying a DFT and an IDFT separately. This leads to the representation

$$C(\phi) = \sum_{m=1}^{3} \sum_{k=-N}^{N} \boldsymbol{p}_k^m \exp(\frac{2\pi k\phi}{2N+1}\boldsymbol{l}_m). \tag{7.8}$$

In the next sections I will formalize the pose constraints for plane curves taken from equation (7.7), but equation (7.8) can directly be inserted for non-plane curves.

## 7.1 Pose estimation constraints for free-form contours

Continuous 3D curves are considered for representing objects. Now a given closed discretizied 3D curve is assumed. That is a 3D contour $C$ with $2N$ sampled points in both the spatial and spectral domain with phase vectors $\boldsymbol{p}_k$ of the contour. The Fourier series development is now replaced by the discrete Fourier transformation. Then the interpolated contour can be expressed in

the Euclidean space as

$$C(\phi) \;=\; \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi}. \tag{7.9}$$

For each $\phi$ does $C(\phi)$ lead to a point in the Euclidean space. Firstly this expression has to be transformed in the conformal space. Then it is possible, similar to the previous section, to substitute this expression in the constraint equations for pose estimation. The transformation of the Fourier descriptors into the conformal space can be expressed as

$$\mathbf{e} \wedge (C(\phi) + \mathbf{e}_-) \;=\; \mathbf{e} \wedge \left( \left( \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \right) + \mathbf{e}_- \right). \tag{7.10}$$

The innermost parenthesis contains the Fourier descriptors in the Euclidean space. The next parenthesis transforms this expression into the homogeneous space and then it is transformed to the conformal space. Substituting this expression into the pose constraint equation leads to

$$\begin{aligned} \left( \boldsymbol{M} (\mathbf{e} \wedge (C(\phi) + \mathbf{e}_-)) \widetilde{\boldsymbol{M}} \right) \underline{\times} \, (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \;&=\; 0 \Leftrightarrow \\ \left( \boldsymbol{M} \left( \mathbf{e} \wedge \left( \left( \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \right) + \mathbf{e}_- \right) \right) \widetilde{\boldsymbol{M}} \right) \underline{\times} \, (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \;&=\; 0. \end{aligned} \tag{7.11}$$

The interpretation of this equation is simple: The innermost part contains the substituted Fourier descriptors in the conformal space taken from equation (7.10). This is then coupled with the unknown rigid body motion (the motor $\boldsymbol{M}$) and compared with a reconstructed projection ray, also given in the conformal space.

Note that twist-generated curves are more general than contours, since contours are assumed as closed curves, whereas twist-generated curves, depending on the boundary angles, are in general not closed. This means that for closed curves Fourier descriptors can be interpreted as generator parameters of special twist-generated curves, but not vice versa. The main point is the coupling of a spectral representation of contours within the pose estimation problem. This is achieved in the previous equation by using a conformal embedding.

### 7.1.1 Multiplicative formalization of 3D Fourier descriptors

So far an additive representation of Fourier descriptors is used in the Euclidean space and then transformed via the homogeneous space to the conformal space. Now a multiplicative description will be derived to gain more compact equations for the pose problem of free-form contours. The inverse Fourier transformation in the Euclidean space can be written as

$$\sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \;=\; \boldsymbol{R}_{-N}^{\phi}\, \boldsymbol{p}_{-N}\, \widetilde{\boldsymbol{R}}_{-N}^{\phi} + \ldots + \boldsymbol{R}_0^{\phi}\, \boldsymbol{p}_0\, \widetilde{\boldsymbol{R}}_0^{\phi} + \ldots + \boldsymbol{R}_N^{\phi}\, \boldsymbol{p}_N\, \widetilde{\boldsymbol{R}}_N^{\phi}. \tag{7.12}$$

Equation (7.12) can be interpreted as a point at the origin, which is then translated $2N+1$ times. Now let

$$\boldsymbol{t}_i^{\phi} \;:=\; \boldsymbol{R}_i^{\phi}\, \boldsymbol{p}_i\, \widetilde{\boldsymbol{R}}_i^{\phi} \quad i \in [-N, \ldots, N] \tag{7.13}$$

$$\boldsymbol{T}_i^{\phi} \;:=\; \frac{1 + \mathbf{e}\boldsymbol{t}_i^{\phi}}{2} \quad i \in [-N, \ldots, N] \tag{7.14}$$

$$\underline{\boldsymbol{O}} \;:=\; \mathbf{e} \wedge \mathbf{e}_0 = \boldsymbol{E}. \tag{7.15}$$

The $\boldsymbol{T}_i^{\phi}$ are translators, containing the rotated phase vectors. Now equation (7.12) can equivalently be written in the conformal space as

$$\sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \;\doteq\; \left(\boldsymbol{T}_N^{\phi} \ldots \boldsymbol{T}_{-N}^{\phi}\right) \underline{\boldsymbol{O}} \left(\widetilde{\boldsymbol{T}}_{-N}^{\phi} \ldots \widetilde{\boldsymbol{T}}_N^{\phi}\right) \tag{7.16}$$

$$= \left(\prod_{k=N}^{-N} \boldsymbol{T}_k^{\phi}\right) \underline{\boldsymbol{O}} \left(\prod_{k=-N}^{N} \widetilde{\boldsymbol{T}}_k^{\phi}\right). \tag{7.17}$$

Note, that the sum on the left hand is an expression in the Euclidean space, whereas the product on the right hand is an equivalent expression in the conformal space. Now the pose estimation problem for free-form contours appears as

$$\left(\boldsymbol{M}\left(\left(\prod_{k=N}^{-N} \boldsymbol{T}_k^{\phi}\right) \underline{\boldsymbol{O}} \left(\prod_{k=-N}^{N} \widetilde{\boldsymbol{T}}_k^{\phi}\right)\right) \widetilde{\boldsymbol{M}}\right) \underline{\times} \left(\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})\right) \;=\; 0. \tag{7.18}$$

It can be seen, that this representation is more compact than the additive one, e.g. in equation (7.11). But indeed it does not change anything for the linearization process in the pose estimation algorithm, but the constraint equation is more concise and the modeling of further extensions (e.g. object deformations) is easier to handle.

### 7.1.2 Coupling kinematic chains with Fourier descriptors

In this section, constraint equations for modeling slight *object deformations* will be derived. Since as first object model a plane object printed on a paper sheet will be used, the motivation is to deal with deformations of the paper sheet during tracking. An object model is assumed and a deformation function $\mathcal{D}$ is added within the pose constraint. A visualization of such deformations is shown in figure 7.2. The deformation function will be modeled as a kinematic chain within the free-form object. Kinematic chains are a compact way to model deformations which keep invariances like the circumference and are therefore well suited in this context.

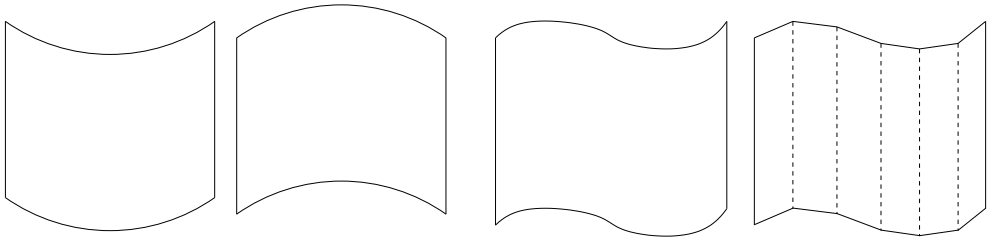


Fig. 7.2: Possible deformations of a sheet of paper along the $y$-axes and their representation as kinematic chain.

The deformation function $\mathcal{D}$ appears in the constraint equation as

$$\left(\boldsymbol{M}\left(\mathcal{D}\left(\left(\prod_{k=N}^{-N}\boldsymbol{T}_k^{\phi}\right)\underline{\boldsymbol{O}}\left(\prod_{k=-N}^{N}\widetilde{\boldsymbol{T}}_k^{\phi}\right)\right)\right)\widetilde{\boldsymbol{M}}\right)\ \underline{\times}\,(\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x}))\ =\ 0. \tag{7.19}$$

A kinematic chain can now be modeled by encapsulating motors $\boldsymbol{M}_i$ of the deformations within the constraint equation,

$$\underbrace{\underbrace{\left(\boldsymbol{M}\underbrace{\left(\prod_{i=1}^{n}\boldsymbol{M}_i^{\theta_i}\underbrace{\left(\left(\prod_{k=N}^{-N}\boldsymbol{T}_k^{\phi}\right)\underline{\boldsymbol{O}}\left(\prod_{k=-N}^{N}\widetilde{\boldsymbol{T}}_k^{\phi}\right)\right)}_{\text{inverse DFT}}\widetilde{\boldsymbol{M}}_i^{\theta_i}\right)}_{\text{kinematic chain on the contour}}\widetilde{\boldsymbol{M}}\right)}_{\text{pose of the deformed contour}}\ \underline{\times}\ \underbrace{(\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x}))}_{\text{projection ray}}}_{\text{collinearity}}\ =\ 0. \tag{7.20}$$

This constraint equation is easy to interpret: The inner parenthesis contains the inverse Fourier transformed phase vectors in the conformal space. The next parenthesis contains the motors $\boldsymbol{M}_i^{\theta_i}$ of the kinematic chain deformation and the last parenthesis contains the motor $\boldsymbol{M}$ with the unknown pose. This is then coupled with the reconstructed projection ray in the conformal space. The unknowns are the pose parameters $\boldsymbol{M}$, the angles $\theta_i$ of the kinematic chain and the angle $\phi$ of the Fourier descriptors. The correspondences (and therefore the angles $\phi$) are later established by using a so-called *ICP*-algorithm.

## 7.2 Experiments on pose estimation of free-form contours

This section presents experimental results of free-form contour based pose estimation. At first the main algorithm for pose estimation of free-form contours is introduced. Though the numerical estimation of the pose parameters is already clarified in chapter 5.5, the main problem is to determine suited correspondences between 2D image features and points on the 3D model curve. Therefore a version of an ICP algorithm is presented and called the *increasing degree* method. Afterwards experiments on the convergence behavior of the algorithm and time performance versus accuracy will be presented. Stability examples for distorted image data are also shown. The algorithm proves as stable and fast (real-time capable) for the scenarios. To deal with 3D objects, partially occluded aspects of objects and object deformations during tracking, modified versions of the increasing degree method are introduced and discussed. It is possible to deal with self-occlusion problems by using sets of Fourier descriptors to model aspects of the object within the scenario and deformations are modeled by coupling kinematic chains within the free-form contours.

### 7.2.1 The algorithm for pose estimation of free-form contours

The aim is to formulate a 2D-3D pose estimation algorithm for any kind of free-form contour. The assumptions are the following:

1. The object model is given as a set of $2N$ 3D points $f_j^3$, spanning the 3D contour. Further their phase coefficients $\boldsymbol{p}_j$ are assumed to be known.

2. In an image of a calibrated camera the object is observed in the image plane and a set of $n$ 2D points $x_j^2$ spanning the 2D contour is extracted.

Since the number of contour points in the image is often too high (e.g. 800 points in the experimental scenario), just every $k$th point (e.g. $k \in 5, \ldots, 20$) is used to get an equal sub-sampled set of contour image points.

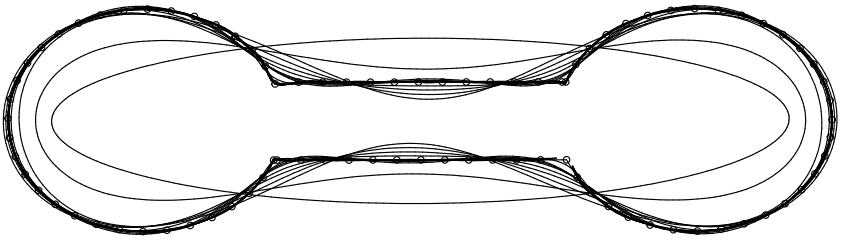


Fig. 7.3: The different approximation levels of a 3D object contour (the *bone* model).

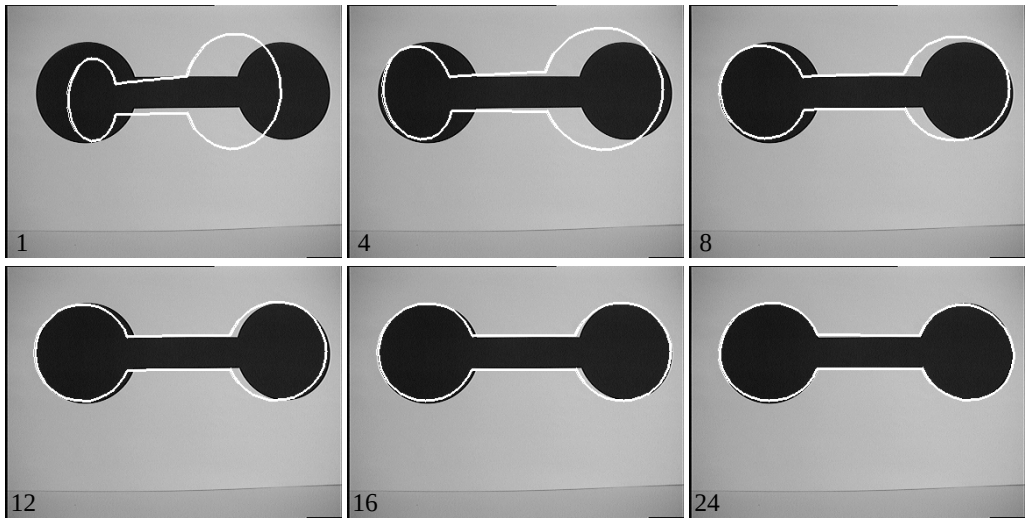



Fig. 7.4: Pose results during the iteration.

Note that there is no knowledge which 2D image point corresponds to which 3D point of the interpolated model contour. Furthermore, a direct

correspondence does not generally exist since the contours are mostly sampled from different starting points and the number of image and object points may also vary.

Using the approach for pose estimation of point-line correspondences, the algorithm for free-form contours consists of iterating the following steps:

```
(a) Reconstruct projection rays from the image points.
(b) Estimate the nearest point of each projection ray to a
    point on the 3D contour.
(c) Estimate the pose of the contour with the use of this
    correspondence set.
(d) goto (b).
```

The idea is, that all image contour points simultaneously pull on the 3D contour. The algorithm itself corresponds to the well-known ICP algorithm, e.g. discussed in [158, 187]. But whereas it is mostly applied on sets of 2D or 3D points, here it is applied on a trigonometric interpolated function and on 3D projection rays, reconstructed from image points.

Note that this algorithm only works if a scenario is assumed where the observations in the image plane are not too different. Thus, it is useful for tracking tasks. A projection of the used object model for the first experiments is shown in figure 7.3. The discrete points and the different approximation levels are shown. The model itself consists of 90 contour points, is plane and has the width and height of $24 \times 8$ cm. I call this object model the *bone* model. Pose estimation results at different iterations are shown in figure 7.4. The white 2D contour is the transformed and projected 3D object model overlaid with the image.

Using the Fourier coefficients for contour interpolation works well, but the algorithm can be made faster by using a low-pass approximation for pose estimation and by adding successively higher frequencies during the iteration. This is basically a multi-resolution method. I call this technique the *increasing degree* method. Therefore the pose estimation procedure starts with just a few Fourier coefficients of the 3D contour and estimates the pose to a certain degree of accuracy. Then the order of used Fourier coefficients is increased and the algorithm proceeds to estimate the pose with the refined object description. This is shown in figure 7.5. In this experiment, the indicated iteration number corresponds directly to the number of used Fourier coefficients *minus one*. This means that two Fourier coefficients are used in the first iteration, four Fourier coefficients in the third iteration, etc. Iteration

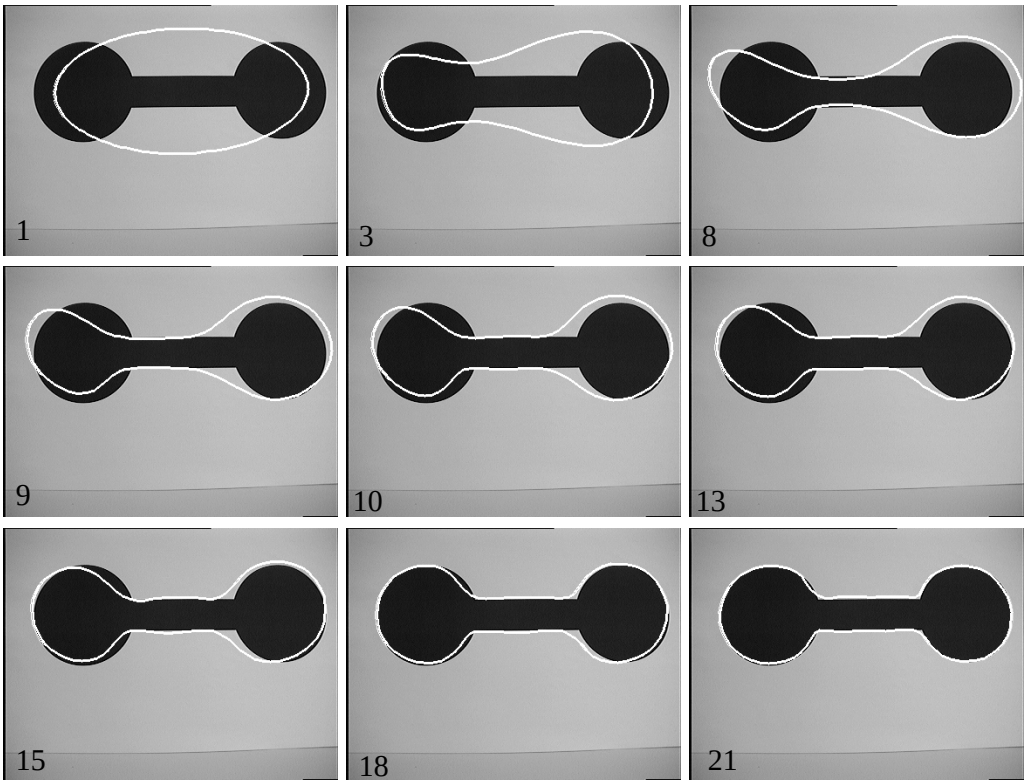



Fig. 7.5: Pose results of the low-pass filtered contour during the iteration.

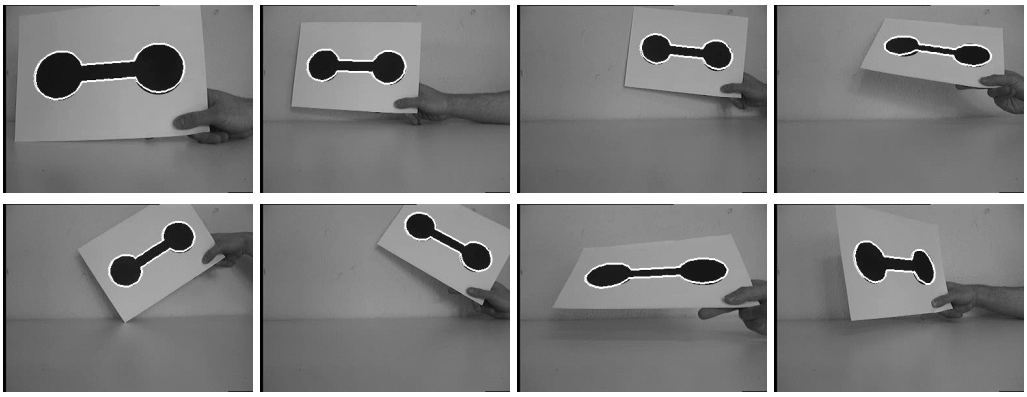


Fig. 7.6: Different pose results of the free-form contour.

21 uses 22 Fourier coefficients and figure 7.5 shows that the result is nearly perfect. Figure 7.6 shows pose results during an image sequence containing 530 images. As can be seen also perspective views of the free-form contour can be estimated.

### 7.2.2 The performance of the pose estimation algorithm

The accuracy and time performance of the algorithm is dependent on the number of object and image points spanning the contours in 2D and 3D respectively. Furthermore, low-pass approximation of the 3D contour can be used. Now results of experiments with changing approximation levels and changing numbers of image points are presented. Note, that the results are estimated on a Sun Ultra 10 work station. There the absolute computing time is not real-time capable. The time performance is also tested on different machines and real-time performance can be achieved with Linux machines, which are much faster.

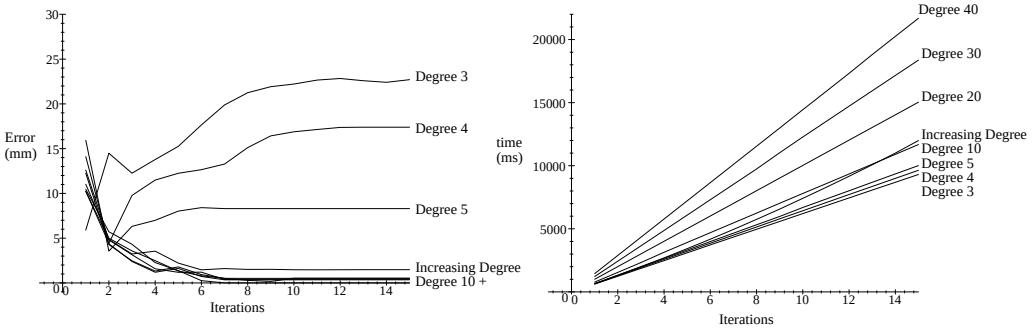



Fig. 7.7: Accuracy and computing time of the algorithm for a constant number of image points (80) and different approximation levels of the contour.

In the first experiment the results of the algorithm for different degrees of contour approximation are compared. Here, the degrees (3, 4, 5, 10, 20, 30, 40) for the contour approximation over the iteration are used and compared with the results of the increasing degree method, similar to figure 7.5. The accuracy of the algorithm is estimated by comparing the translational error vector with the ground truth. The result is shown in figure 7.7. It can be seen that the algorithm converges after less than 10 iterations. Then the error vectors do not change any more. It is clear, that the use of fewer numbers of Fourier coefficients leads to fast but more inaccurate results. The *increasing degree* method finds a good optimum between computing time and accuracy of the result. The algorithm converges after 7 iterations.

In a second experiment, the *increasing degree* algorithm is used. Now the number of extracted contour points is changed. In this experiment 10, 12,

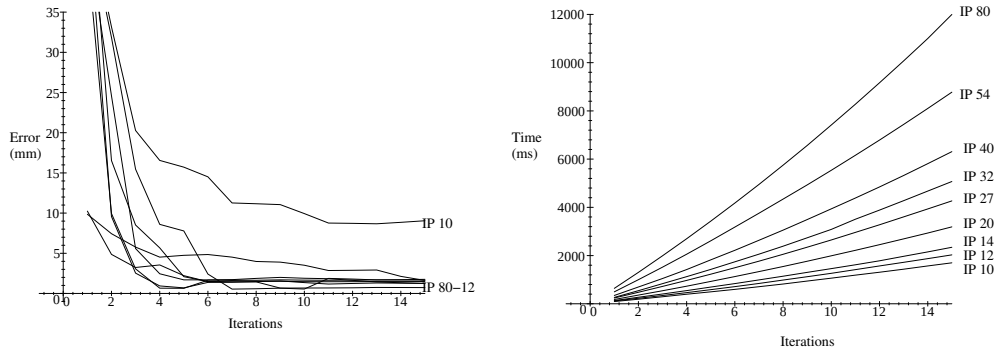



Fig. 7.8: Accuracy and computing time of the algorithm for a changing number of used image points and the increasing degree algorithm.

| Computer | pose | min. search | total (25 It.) |
|---|---|---|---|
| Pentium 4, 2 GHz | 3ms | up to 20 ms | 405 ms |
| Pentium 3, 850 MHz | 15 ms | up to 25 ms | 783 ms |
| Sparc Ultra 10 | 325 ms | up to 80 ms | 10295 ms ($\sim$ 10 sec) |
| Sparc Ultra 1 | 8921 ms | up to 667 ms | 232265 ms ($\sim$ 3.5 min) |
| Sparc 4 | 13322 ms | up to 1505 ms | 356811 ms ($\sim$ 6min) |
| Sparc 10 | 23509 ms | up to 1743 ms | 622975 ms ($\sim$ 10min) |

Tab. 7.1: Computing time of the increasing degree method for the same scenario on different machines for 90 model points, 80 image points and 25 iterations.

14, 20, 27, 32, 40, 54 and 80 regularly sampled image points are used and their accuracy and computing time is compared. The result is presented in figure 7.8. It can be seen, that the number of image points used affects both the computing time and the accuracy. But in comparison with the previous experiment there exists a critical break point with regard to the accuracy of the algorithm. While the use of $14 - 80$ image points does not affect the quality of the pose too much, the use of 10 points or less leads to wrong poses.

These results hold for just this scenario and will change for other scenarios. The main result is that it is indeed possible to use the whole image and object information available to estimate the pose of the free-form contour. But this will be paid with high computing time. Instead, the use of low-pass filtered or of sub-sampled contours fastens computing and leads to good results. In this scenario the computing time can be reduced from 35 seconds

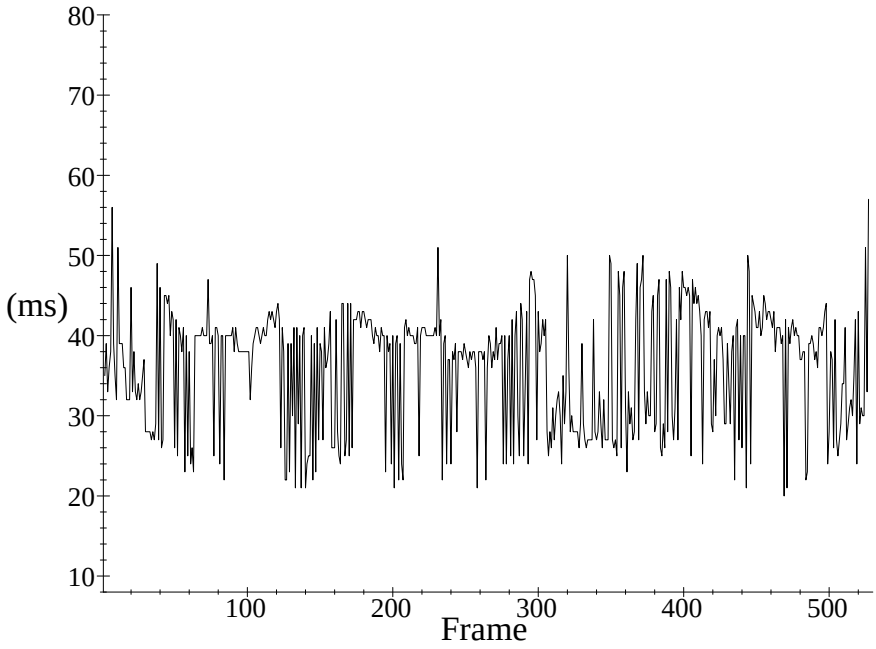



Fig. 7.9: Computing times for an image sequence containing 500 images.

to less than 1 second without introducing non-tolerable errors.

Indeed, the computing time is very dependent on the machine itself. Therefore, the same algorithm is compiled on different machines. Then the computing time for exactly the same scenario is compared. The result is shown in table 7.1. As can be seen, e.g. the computing time for the increasing degree method with 80 image points is 783 ms on a standard Linux 850 MHz machine. The column *pose* shows the computing time for each pose. The column *min. search* shows the computing time for estimating the minimum distance between projection rays and the model curve. Since the increasing degree method is used, the computing time for estimating the distances varies and increases with increasing number of Fourier coefficients. Therefore just the maximum computing time is shown. The column *total* shows the total computing time for 25 iterations. But since the algorithm is applied on image sequences, 25 iterations are seldom needed.

Figure 7.9 shows the computing times for an image sequence containing

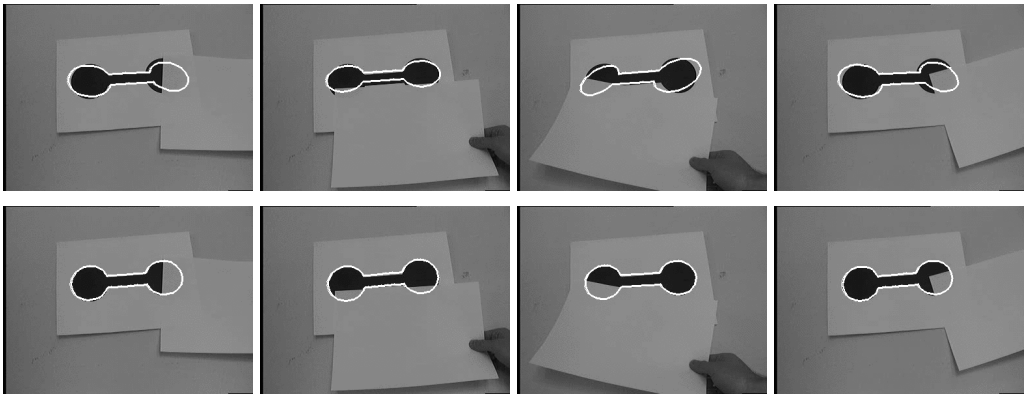


Fig. 7.10: Different pose results for distorted image data. The first row shows results obtained with the non-modified ICP algorithm. The second row shows pose results obtained with the outlier-elimination during the ICP algorithm.

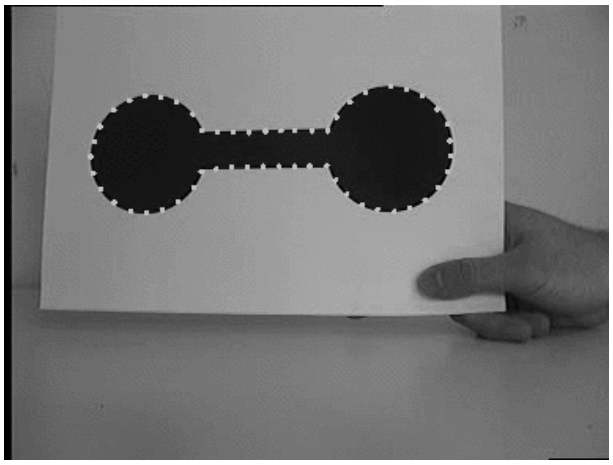


Fig. 7.11: Point correspondences for the noise sensitivity experiment.

520 images. The computing time for each image varies between 20ms and 55ms. The average computing time is 34ms, which is equivalent to 29 fps. These results were achieved with a 2GHz Pentium 4 computer. Many ideas to fasten the algorithm can also be found in [158]. This is not done yet and will be part of future work. The main result is that the algorithm can also be used for real-time applications on standard Linux machines.

The robustness of the algorithm with respect to distorted image data is shown in figure 7.10. In this image sequence (containing 450 images) the image contour is distorted by covering parts of the contour with a white paper.

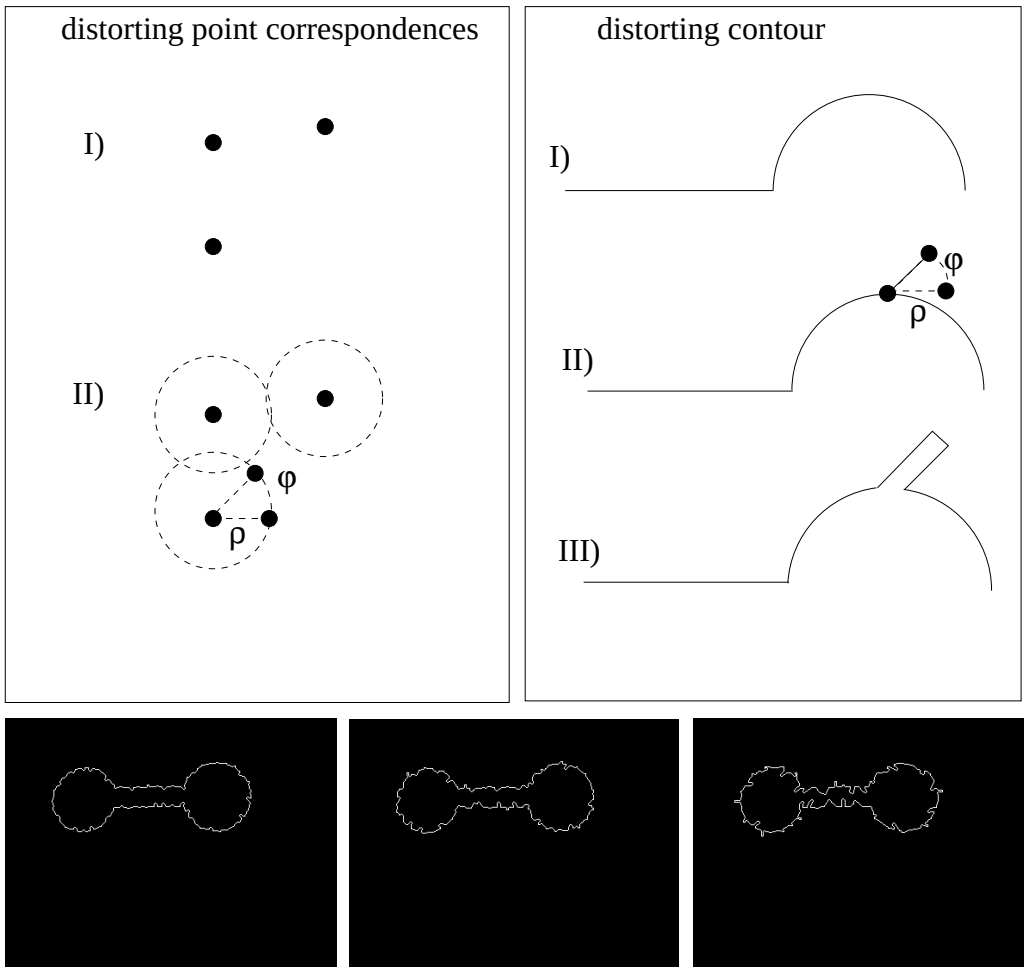



Fig. 7.12: Top: Principles for distorting the image and contour data. Bottom: Example images for distorted contours.

This leads to minor or more extreme errors during the contour extraction in the image. The first row of figure 7.10 shows the results obtained with a non-modified ICP-algorithm. In section 5 it is already clarified, that the constraint equations express a geometric distance measure in the 3D space. Therefore it is easy to detect outliers and implement an algorithm which automatically detects outliers and eliminates their equations. Some results of the modified algorithm are shown in the second row of figure 7.10. I call this procedure the *outlier-elimination* method. As can be seen, the obtained results are much better. But indeed, these examples give just a guess about the stability of the proposed method. It is not possible to compensate totally wrong extracted contours or too much missing information.

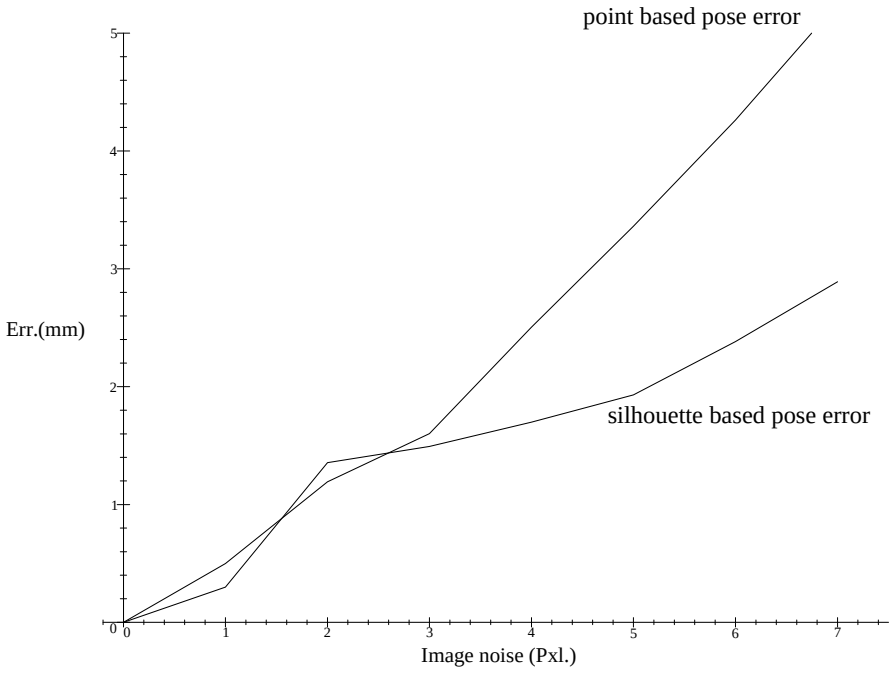



Fig. 7.13: Noise stability of the different algorithms.

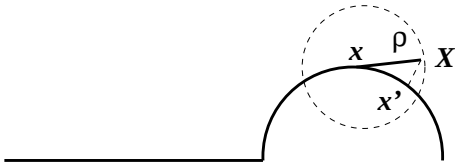



Fig. 7.14: Example for self-adaption during the ICP-algorithm: Instead of the correspondence $(X,x)$, the correspondence $(X,x')$ is chosen.

In the next experiment the noise sensitivity of point based versus silhouette based pose estimation is compared with respect to image noise. Therefore the *bone*-model is chosen as object model and 50 image and model correspondences are estimated. The correspondences are depicted in figure 7.11. The main question is now, how to distort the image data on the contour, so that they can be compared with distorted point sets. The idea is visualized in figure 7.12: For the original point set (left, I), each point is translated with the image noise $\rho \in [0,\ldots,7]$ and rotated with a random angle $\phi$. This leads to a distorted point set (left, II). For the original image contour (right, I) each sampled point is translated with the image noise $\rho \in [0,\ldots,7]$ and rotated with a random angle $\phi$, similar to the point set. Then the line segment from

the contour point to the distorted point is drawn. This is shown in figure 7.12 (right, II). From this distorted silhouette the new contour is extracted (right, III). Note, that the circumference is increasing, therefore the sampling rate is adapted to the new contour. Figure 7.12 shows examples of different distorted contours in the last row.

Using these (comparable) kinds of distorting the images, the pose results for different noise levels are compared and shown in figure 7.13. As can be seen, little deviation (up to 3 pixel) does not lead to real differences, but higher deviation leads to a more stable behavior of the silhouette based pose estimation algorithm. The reason is the combination of pose estimation within the ICP algorithm: For distorted image data, the corresponding model data are adapted to the specific situation and therefore better suited correspondences between the image and model data are estimated. This means, if a point $x$ is distorted and the correspondences are adaptively chosen, the correspondence between the distorted point and a neighbor model point $x'$ might be better suited as correspondence set. This is visualized in figure 7.14. That is the reason for the more stable behavior of the silhouette based pose estimation algorithm. Note, that this experiment is done with the original ICP-algorithm, without the outlier elimination.

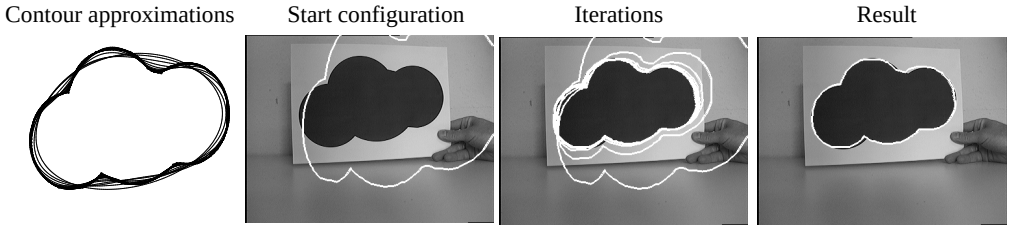



Fig. 7.15: Contour approximations of the *cloud* object model and its convergence behavior.

Figures 7.15-7.19 present results of other object models: I call the first object model the *cloud* and the second object model the *edge*. Figure 7.15 shows the 3D contour approximations of the cloud model in the left image and a convergence example in the other images. The correspondences during the iteration of the ICP algorithm are visualized in figure 7.16. It can be seen that each sampled point on the image contour is assigned to a point on the 3D object contour (and not vice versa). Though many correspondences are wrong in the beginning, during the iteration they are more and more corrected.

Figure 7.17 presents results of a sequence containing 700 images. As can

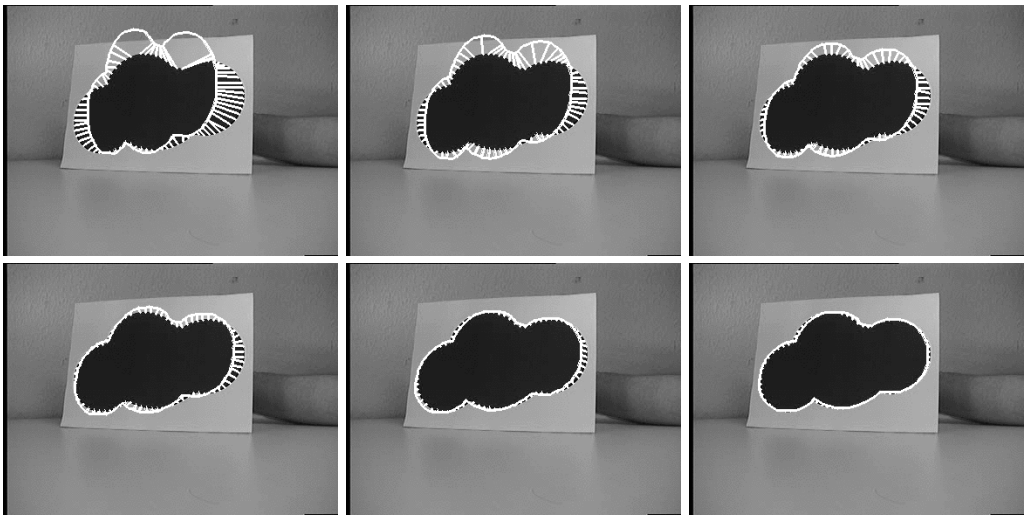


Fig. 7.16: Pose results during an iteration with visualization of the chosen correspondences.

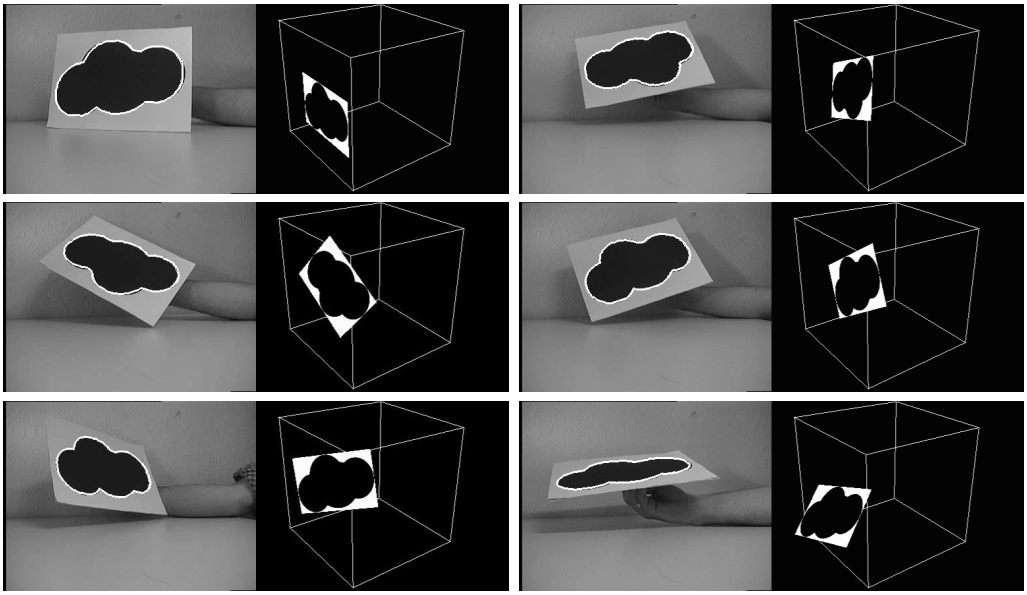


Fig. 7.17: Example images and 3D poses taken from an image sequence containing 700 images.

be seen, also strong perspective views, as in the lower right image, can be estimated. To compare the visual observable error (as a drawn contour in the image) with its real 3D pose the relative 3D pose in a virtual environment is visualized in figure 7.17. The 3D pose matches with the observations in the

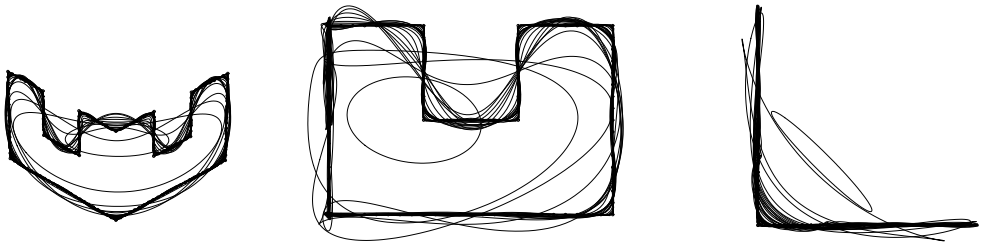


Fig. 7.18: Three perspective views of the non-plane *edge* model and its approximations.

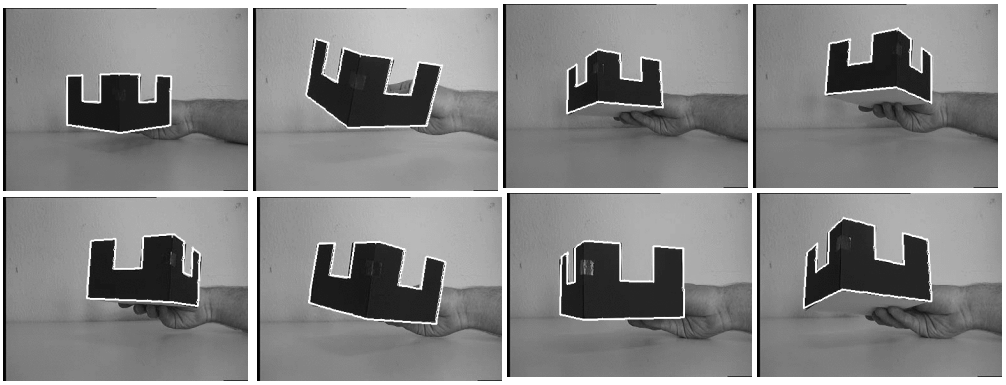


Fig. 7.19: Example images from an image sequence containing 500 images.

image.

Figure 7.18 shows approximation levels of a non-plane object model in three different perspective views. This is an example in which the object model contains edges. Interpolation of a contour with Fourier descriptors leads to a trigonometric interpolated function. So the edges are always smoothed and several descriptors (around 40 are used for the experiments) are required to achieve an acceptable result. Figure 7.19 presents different results from an image sequence containing 500 images.

Object contours which contain concavities are in danger to get trapped in local minima during using the ICP algorithm with the gradient descent method for pose estimation. Though it is not always possible to find the global minimum (and therefore the best pose), using contour approximations helps to avoid local minima. This effect is achieved by using firstly a low-

pass contour for pose estimation and then over the iterations a more refined contour.

The last two object models (the *cloud* and the *edge*) contain more local minima than the first one. Therefore more Fourier descriptors are needed to gain acceptable results. This increases the computing time. While for the first object model the average computing time is 34ms, the average computing time of the cloud and edge model are 60ms and 120ms, respectively.

### 7.2.3 Simultaneous pose estimation of multiple contours

In the experiments of the last sections, the object model is assumed as one (closed) contour. But many 3D objects can more easily be represented as a set of 3D contours expressing the different aspects of the object. In this section the object model will be extended to a set of 3D contours. The main problem here is, how to deal with occluded or partially occluded contour parts of the object. For the first experiment the *edge* object model will be used, which is presented in figures 7.18 and 7.19. Now the model will be interpreted as an object containing two sides and one ground plate. This means a set of three plane contours is gained to model the object. The three contours are merged to one object and perspective views are shown in figure 7.20. The three contours are assumed as rigidly coupled to each other. This means that the pose of one contour automatically defines the pose of the other contours.

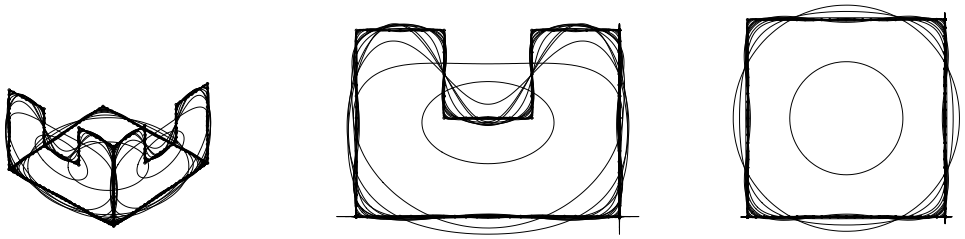


Fig. 7.20: Three perspective views of an object which is interpreted as a set of contours. The different approximations of the contours are also drawn.

The algorithm to deal with partially occluded object parts is simple and

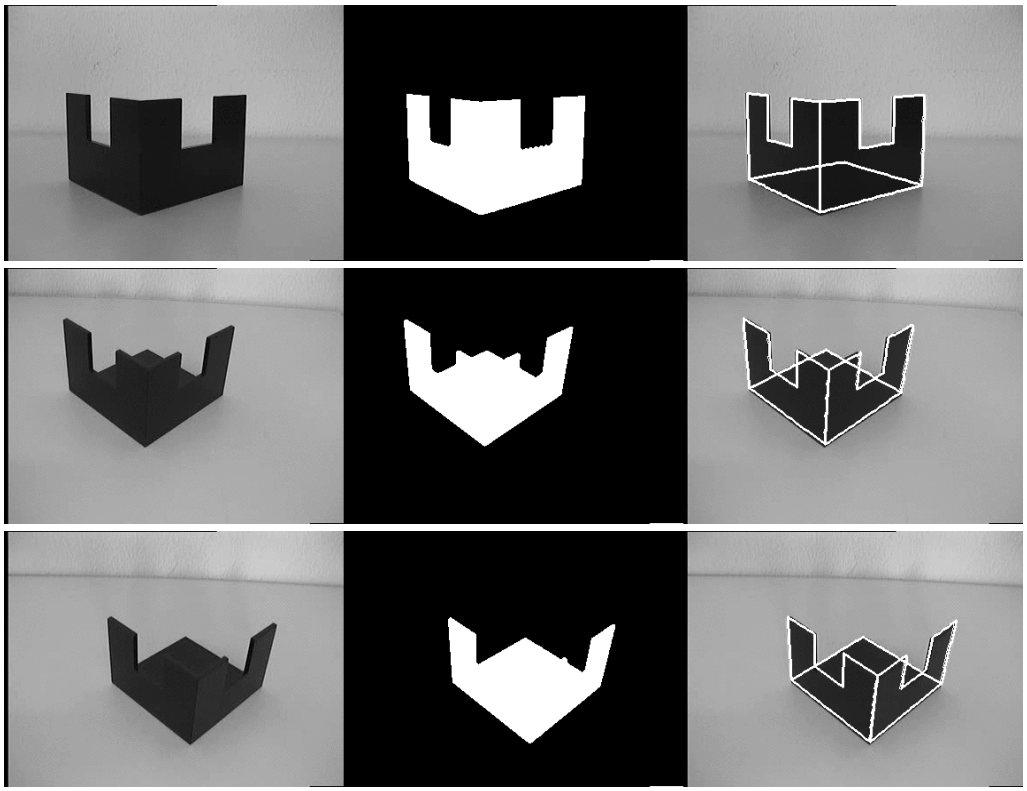


Fig. 7.21: Pose results of an object with partially occluded contours. The left image shows the original image. The middle image shows the extracted silhouette (from which the boundary contour is extracted) and the right image visualizes the pose result. Note, that also the occluded parts of the model are drawn and uniquely determined by the visible parts.

effective:

```
Assumptions: n 3D contours and one boundary contour in the image
             dist(P,R) a distance function between a 3D point P
             and a 3D ray R.
Result: Correspondences and pose.

(a) Reconstruct projection rays from the image points.
(b) For each projection ray R:
(c)   For each 3D contour:
     (c1) Estimate the nearest point P1 of ray R to a point on
     the contour.
```

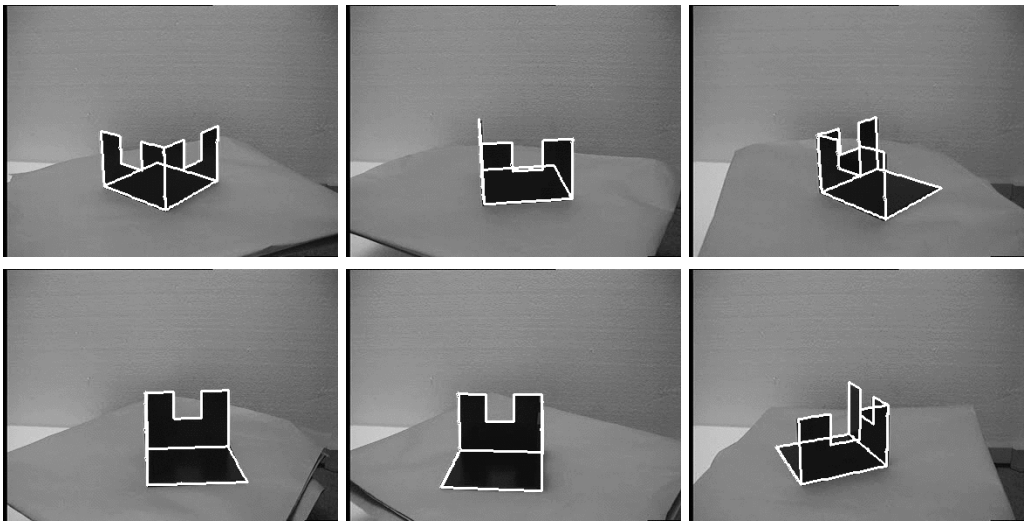


Fig. 7.22: Pose results of an image sequence containing different aspect changes and degenerate situations.

```
     (c2) if (n==1) choose P1 as actual P for the
               point-line correspondence
     (c3) else compare P1 with P:
             if dist(P1,R) is smaller than dist(P,R) then
                choose P1 as new P.
(d) Use (P,R) as correspondence set.
(e) Estimate pose with this correspondence set.
(f) Transform contours, goto (b).
```

The idea is to apply the ICP algorithm not only to one image contour and one 3D contour, but now to one image contour and a set of 3D contours.

This implies: For each extracted image point must exist one model contour and one point on this contour which corresponds to this image point. Note, that the reverse is in general not possible.

Figure 7.21 visualizes the problem of partially occluded contour points. The only image information used is the observed boundary contour of the object. By using a priori knowledge (e.g. assuming a tracking assumption), the pose can be recovered uniquely. This means that the algorithm can infer the position of hidden components from the visible components.

The computing time is proportional to the number of used contours. While the algorithm needs 120ms for each image in the experiments of figure 7.19, it now needs 400ms for each image. But a more general concept is

achieved, since there is no restriction to one special view of the object any more. Instead the algorithm can deal with aspect changes of the contour in an efficient manner. This is demonstrated in figure 7.22 in case of quiet different aspects of a 3D object. The images are taken from an image sequence containing 325 images. In this image sequence the object is put on a turn table which makes a $360^o$ degree turn of the whole object. The aspects of the objects are changing and half-side models can not be used any more, but just the whole object. The tracking algorithm does not fail and is even able to cope with degenerate situations.

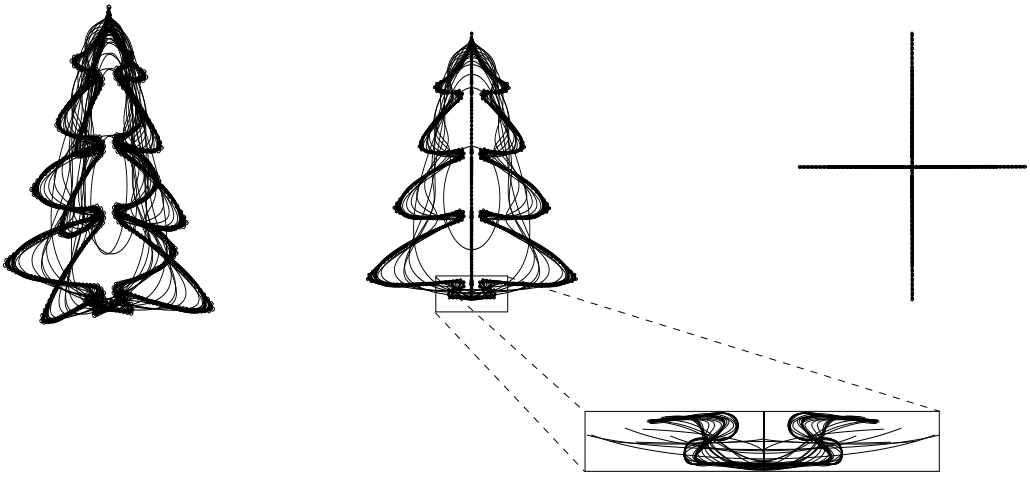


Fig. 7.23: One perspective, frontal and top view with approximations of the tree model. The close up visualizes the complexity of the object model.

In the next experiment, the shape of a 3D tree is used as object model. The contour approximations are shown in figure 7.23. As can be seen in the close-up, here also many descriptors (around 50) are needed to get a sufficient approximation of the model. Pose results of an image sequence containing 735 images are shown in figure 7.24. The interesting part of this model, in contrast to the previous ones, is not only its complexity: This model contains two *nested* contours and is therefore much more complicated than the previous ones. Because of its complexity (the number of Fourier coefficients and the nested contours) the computing time is two seconds for each image on a 2 GHz Linux machine.

The next experiment, see figure 7.25, presents results of a tracked cup during an image sequence. The cup is modeled by three contours of the cup

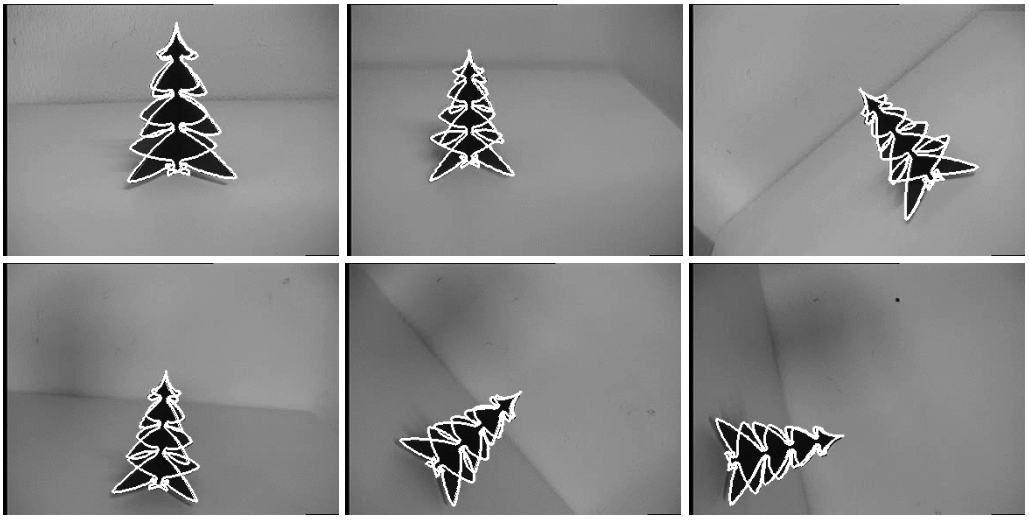


Fig. 7.24: Pose results of the tree model during an image sequence.

and since the cup is moved by a human hand, the extracted silhouette is noisy and the outlier elimination method explained in section 7.2.2 has to be applied additionally. Figure 7.25 shows four processed images with their processing steps: The first image in each row shows the original image. The second image in each row shows the extracted silhouette, from which the boundary contour is extracted. The third image shows the pose result of the previous image, projected in the actual processed image and the initial correspondences from the extracted silhouette to the object model. As can be seen, the hand leads to remarkable outliers in the correspondence set, which will be detected and eliminated. The last image shows the pose result after the outlier elimination and the ICP-algorithm. As can be seen, the pose result is accurate and it is possible to deal with multiple contours combined with an outlier elimination during image sequences.

From the last experiments results the possibility to model objects with contours representing different aspects of the object and to fuse these within the pose scenario.

### 7.2.4 Pose estimation of deformable objects

To model object deformations (as kinematic chains) within the ICP-algorithm, they must be modified to arrange the kinematic chain segments along the contour in a suited manner. The ICP-algorithm is the same as in the previous section, but it is now modified to the kinematic chain: Since the object is

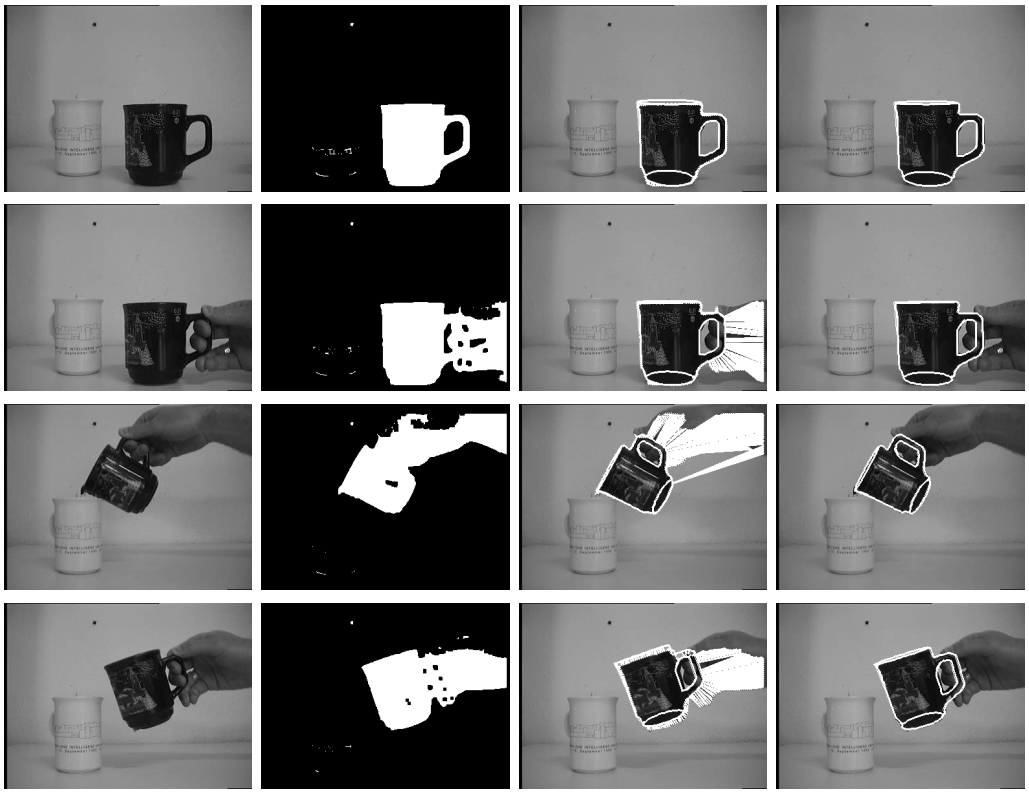
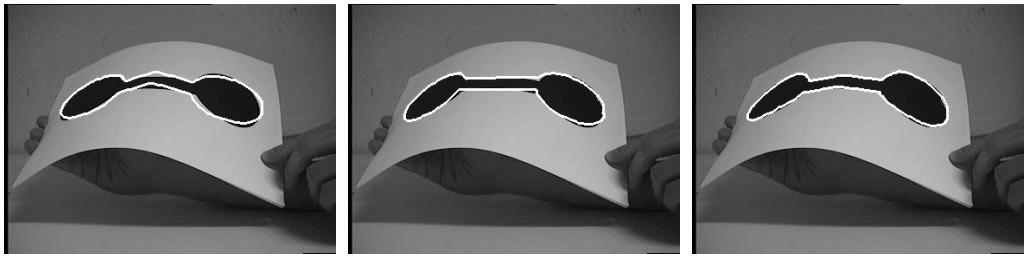


Fig. 7.25: Image processing steps and pose results of the cup model during an image sequence.

not rigid any more, an index function is defined to establish the degree of the kinematic chain for each point on the contour.



1–Twist kinematic chain 2–Twist kinematic chain 3–Twist kinematic chain

Fig. 7.26: Pose result of a free-form object containing one, two or three kinematic chain segments.

The pose estimation algorithm for kinematic chains (section 6.1) can now be used within the constraint equations defined in section 7.1.2 and used to

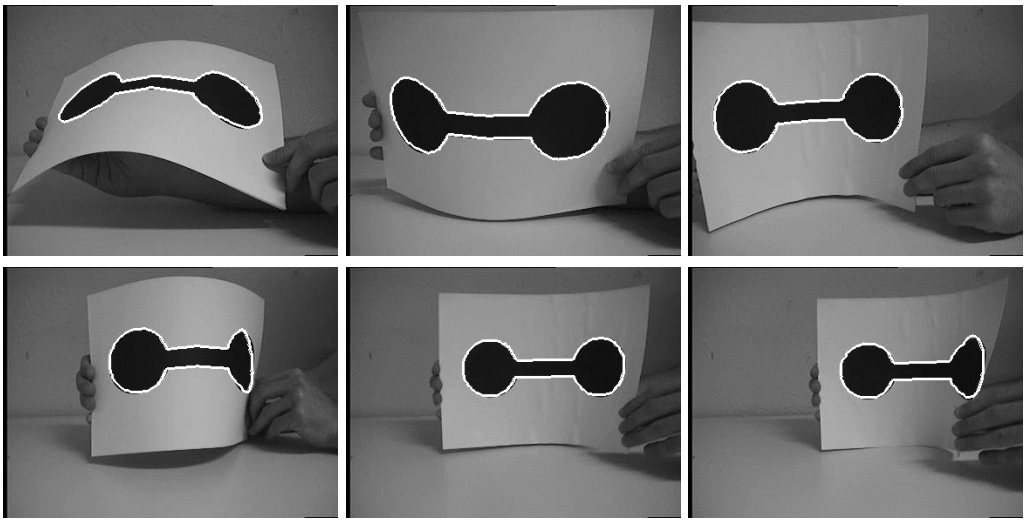


Fig. 7.27: Pose results of a free-form object taken from an image sequence with 520 images. The deformation function is modeled as a kinematic chain.

estimate the pose and the kinematic chain parameters of the object. Iterating this leads to a new pose and a new object configuration, which converges during the ICP-algorithm to the local minima. The algorithm can be summarized as follows:

```
(a) Reconstruct projection rays from the image points.
(b) Estimate the nearest point of each projection ray
    to a point on the 3D contour,
(c) Estimate the number of revolute joints to the point
    on the 3D contour.
(d) Estimate the pose and kinematic chain parameters
    with the use of this correspondence set.
(e) Transform contour by using the estimated pose
    and kinematic chain parameters.
(f) Estimate new Fourier descriptors.
(g) goto (b).
```

In the implemented algorithm, the number of kinematic chain segments is free to choose and the effect of just one, two or three twists is visualized in figure 7.26. As can be seen, not many twists are needed to get a good approximation of the deformation. In our experiments 1 to 5 twists are used. Figure 7.27 shows pose results of an image sequence containing 520 images.

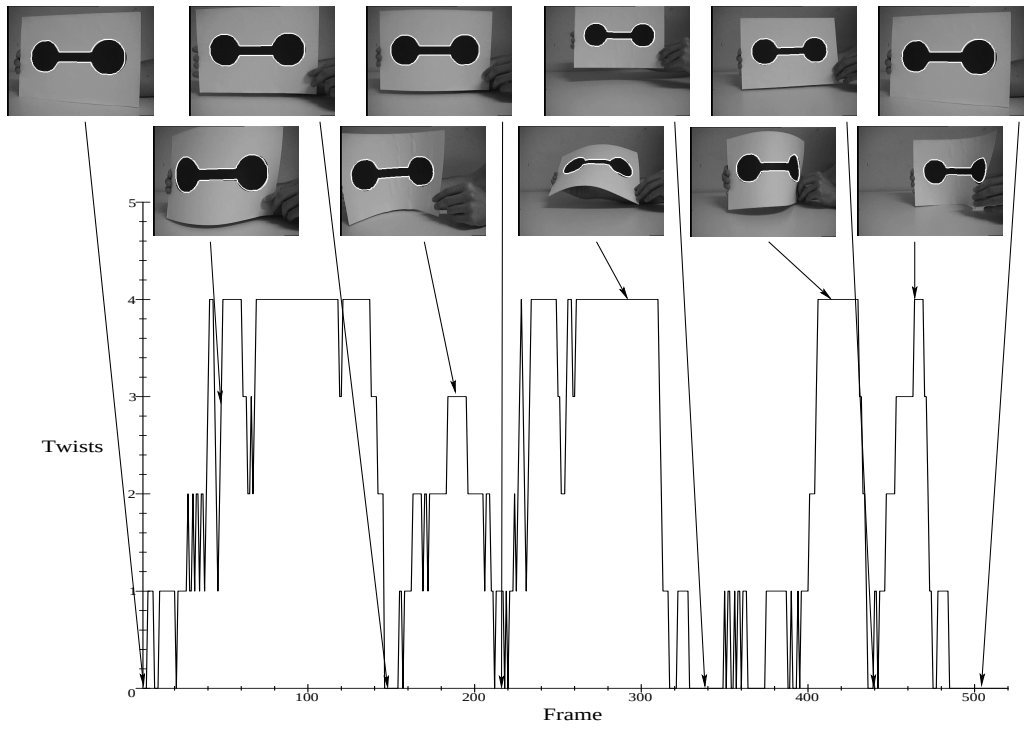



Fig. 7.28: Adaptive choice of twists for modeling object deformations during an image sequence. Note, that for slight deformations fewer twists are used than for larger deformations.

There are two major problems in dealing with a fixed set of twists modeling the object deformation: Firstly can the use of too many twists lead to local minima and *wrong* poses. This occurs especially, when not many twists are needed (e.g. there is only a slight object deformation), but many twists are modeled. Secondly does the use of *many* twists decrease the computing time of the pose estimation algorithm, since additional unknowns are modeled which are not always needed. Therefore a modification of the algorithm is done, which chooses the number of twists adaptively, depending on the level of deformation. Examples of an image sequence are presented in figure 7.28. The diagram shows the frame number during the image sequence on the $x$-axis and the used number of twists on the $y$-axis. The example images visualize that the used number of twists is consistent with the degree of deformation. The increased time performance is shown in figure 7.29. The $y$-axis gives on the one hand the used number of twists (consistent with figure 7.28) and on the other hand the computing time for estimating one pose during the ICP-algorithm. As can be seen, the use of more twists increases the comput-

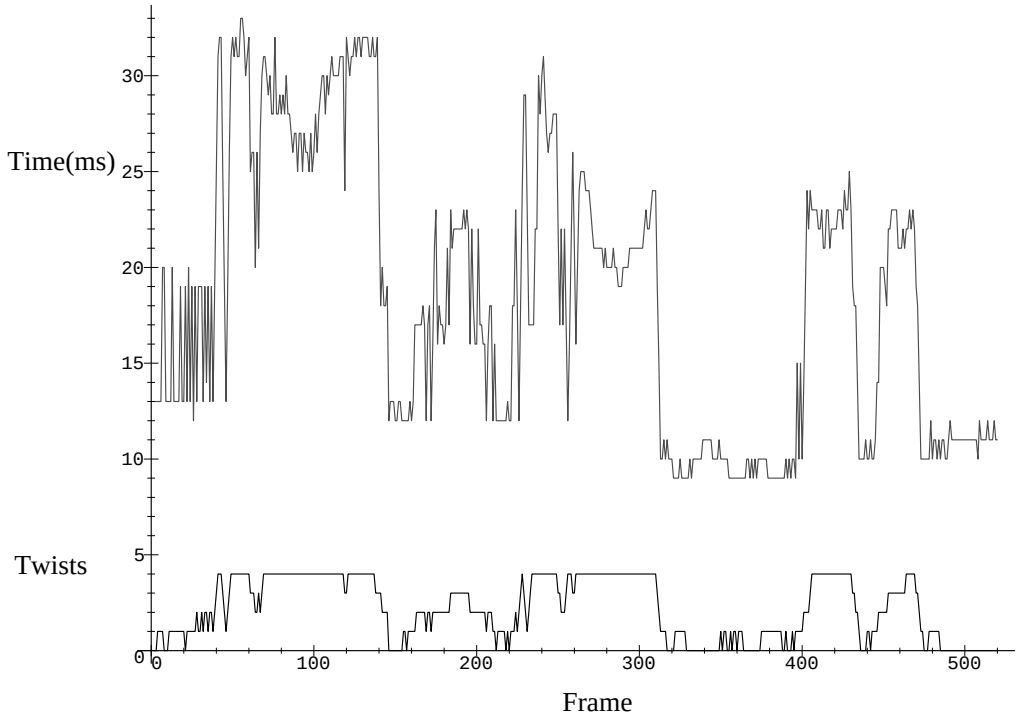



Fig. 7.29: The time performance for using different numbers of twists during an image sequence.

ing time, and the adaptive choice of the number of twists during the image sequence leads to a situation dependent optimized time performance. Note, that only the time for one pose is shown. Combined with the ICP-algorithm (which takes between two and eight iteration steps), the overall computing time for one frame varies between 20ms and 250ms.

## Chapter 8

# CONCLUSION

The main topic of this thesis can be summarized as:

**The problem of 2D-3D pose estimation**.

The problem of 2D-3D pose estimation means to estimate a rigid body motion, which leads to a best fit between object data and image data. This is one of the oldest computer vision tasks and as mentioned in chapter 1.1, several solution approaches for several variations of this problem already exist. The topic is important for many computer vision and robot vision applications, like self-localization of mobile robots, object localization, recognition, grasping or manipulation. The main focus concentrates on the geometric modeling and application of the pose problem. 3D object models are treated two-fold, feature based and free-form based: While the feature based pose scenarios provide constraint equations to link different image and object entities, the free-form approach for pose estimation is achieved by applying extracted image silhouettes from objects on 3D free-form contours modeled by 3D Fourier descriptors. The reason is, that starting from simple features (e.g. point features) an extended scenario is derived, which deals with higher order features such as lines, planes, circles, spheres, kinematic chains or cycloidal curves. This scenario is extended to general free-form contours by interpreting contours generated with 3D Fourier descriptors as $n$-times nested cycloidal curves. The results of this thesis are summarized in the following points:

Firstly, the geometry of the 2D-3D pose estimation scenario is analyzed and the interaction of entities given in different mathematical spaces (Euclidean, affine and projective) is considered in a geometric algebra: The conformal geometric algebra provides a homogeneous model for stereographic projection and expresses rigid motions in 3D space as rotations on a hyper-

sphere (chapter 3, 4). This coupling of projective geometry with kinematics is used within the multivector concepts to formalize the pose estimation problem for point, line and plane correspondences. It leads to a compact and linear description of the pose problem which contains a distance measure (chapter 5). These equations can further be scaled by a scalar which allows for an adaptive weighting of the constraints (chapter 5.6.1). The constraint equations are solved by linearizing and iterating the equations. This corresponds to a gradient descent method as in [116], but applied to the 3D space (chapter 5.5). The estimation of pose parameters is very fast and takes 3 to 5ms on a standard Linux machine.

Secondly, the basic entities for pose estimation are extended to more complex ones: kinematic chains, circles, spheres or cycloidal curves. The approach for pose estimation of kinematic chains extends the approaches presented in [25, 74, 181], since no scaled orthographic camera model is assumed but a full perspective one. Furthermore, the full geometry of kinematic chains is modeled and no hierarchy is built up which would be accompanied by the loss of geometric information as discussed in chapter 6.4.1. The experiments in chapter 6.4.2 do further show the possibility of using different entities simultaneously.

Thirdly, the twist approach for modeling cycloidal curves as virtual kinematic chains is related to model 3D contours by using Fourier descriptors. This leads to constraint equations for 3D free-form contours in a full projective camera model and therefore extends e.g. affine pose algorithms [6]. In this context ICP-algorithms are used to estimate the correspondences and poses for image silhouettes and object contours. The use of low-pass information enables one further to avoid local minima and to speed up the algorithm. Furthermore, an automatic outlier detection is possible, which stabilizes the pose results. Extensions to multiple contours, partially covered contours and adaptive deformable contours are also presented (chapter 7.2.3 and 7.2.4). This part links signal theory, geometry and kinematics and is applied advantageously for 2D-3D silhouette based free-form pose estimation.

The pose estimation algorithm is applied in several projects [68, 66, 139, 101, 160, 102] and proves as robust and fast.

Parts of this thesis are published in [168, 167, 154, 189, 153, 144, 145, 143, 152, 146, 151, 147, 148, 150, 149].

## 8.1 Further extensions, open problems and future work

The next (and non-trivial) extension of contour based free-form pose estimation is pose estimation of free-form surfaces. This has a much higher degree of complexity, similar to the extension of the 1D analytic signal and 1D quadrature filters to 2D in an isotropic way, as presented in [54]. New mathematical foundations are necessary, maybe by using the $n$-dimensional hyperbolic model, containing the half-space model, the Klein ball model or the hyperboloid model, presented by H. Li in [111] as further extensions of the conformal geometric algebra. Nonetheless it is also necessary not only to model incidence of projection rays with points on the contours, but also tangentiality from silhouette reconstructed projection rays to the surface, similar to equation (6.34), (6.35) and the experiments with the lamp-shade presented in figure 6.30. There exist several algebraic forms of surface descriptions, like superquadrics, polygonal meshes, generalized cylinders or ruled surfaces [33]. The task is to identify and apply a well suited representation within the pose formalism.

For free-form pose estimation a so-called ICP-algorithm is used to estimate correspondences during iteration with the pose estimation algorithm. Though the ICP-algorithm works fine and stable in tracking situations, its computational overhead (the nested loops) leads to hardly realizable real-time systems for complex object models. Here also some work is possible and promising. E.g. no fast Fourier transformation is applied so far and the minima-search in the gradient descent method is highly parallelisable. But maybe new search strategies are better suited than the used ICP-algorithm.

Another extendable topic is the image processing for pose estimation. So far easy scenarios are assumed, e.g. with little background noise. The image processing is kept simple, since the geometric aspects of the pose scenario are dealt with in this thesis. Noise is considered in the context of a distance measure within the pose constraints (see e.g. figures 7.10 and 7.25). Since the image processing is kept simple, dealing with more complex scenarios requires a suited image processing which leads to a new field of work regarding stability versus complexity. I dealt with these problems a little bit in the experiments for the navigation system in chapter B.4. The navigation system shows that dealing with further uncertainties and noise ratios leads to hardly realizable stable real-time systems, and parallel system architectures are necessary to cope with complex tasks.

Indeed, also the navigation system presented in chapter B.3 is just dealing with basic behaviors for navigation (follow path, self localization, etc.). It

is kept simple and several extensions can be imagined: It can be extended to more complex behaviors, e.g. visuable obstacle detection (using e.g. the inverse perspective [119]), map building, door opening, dealing with multiple landmarks simultaneously, etc. The interaction of the navigation system with these tasks is interesting for future research and important for stable running systems.

In this work only pinhole-camera models are considered. New camera geometries like catadioptric mirrors for omni-directional views require a new embedding to extend the results of this thesis to more general camera systems. Since catadioptric mirrors are often half-spheres or parabolas, it should be possible to apply the homogeneous model to these camera models. That it is possible to model reflections on entities in the conformal geometric algebra is discussed in chapter 6.3.3 and in more detail in e.g. [107]. Indeed, the pose scenario is formalized in the 3D kinematic framework. This means for extended camera models, that mainly the reconstruction of image entities (e.g. estimating projection rays on reflected mirrors) is needed to apply the results of this thesis on catadioptric or other camera models. The reason is, that reconstructed rays are independent of the camera model they are reconstructed from and only the reconstructed rays are used in the pose scenario (see e.g. figure 2.3). This leads to further extensions for computer graphics or navigation and is an interesting topic for future research.

## Appendix A

# BASIC MATHEMATICS

This chapter introduces the basic mathematical definitions used in this thesis. In chapter A.1 the definition of a field, a vector space and the Euclidean space will be introduced. Then it will be continued with the definition of the affine space, the kinematic space as a special affine space and the projective space. In chapter A.2 groups, Lie groups and Lie algebras are defined and applied to the modeling of rigid body motions. Afterwards, the projective space is used to formalize the pinhole camera model introduced in chapter A.3, which proofs useful to model CCD-cameras. Section A.4 is devoted to the basics of signal theory. This part is related to chapter 7 in the thesis and introduces the discrete Fourier transformation by using matrix calculus and complex numbers. For more detailed information about these topics, the reader should consult [62, 114, 124, 123, 32, 48, 25, 47, 180, 96, 91, 29, 172, 20]. This chapter will only give a brief introduction into the different concepts. The whole theory is well studied and would be too extensive in this context.

## A.1 Mathematic spaces

A common way to introduce vector spaces is to start with the definition of a *field* $K$:

**Definition A.1** *A field is a set $K$, with two functions $+ : K \times K \to K$ and $\cdot : K \times K \to K$ fulfilling the following rules:*

*(A1)* $\forall a, b, c \in K : a + (b + c) = (a + b) + c$
*(A2)* $\forall a, b \in K : a + b = b + a$
*(A3)* $\exists! 0 \in K : a + 0 = 0 + a = a$
*(A4)* $\exists! (-a) \in K : a + (-a) = 0$
*(M1)* $\forall a, b, c \in K : a \cdot (b \cdot c) = (a \cdot b) \cdot c$
*(M2)* $\forall a, b \in K : a \cdot b = b \cdot a$
*(M3)* $\exists! 1 \in K : a \cdot 1 = 1 \cdot a = a$
*(M4)* $\forall a \in K \neq 0 \exists! a^{-1} \in K : a \cdot a^{-1} = 1$
*(D)* $\forall a, b, c \in K : (a + b) \cdot c = a \cdot c + b \cdot c.$

For further notations it is convenient to neglect the $\cdot$, this means $a \cdot b \equiv ab$. Furthermore it is convenient to write $a - b$ instead of $a + (-b)$. Examples are rational $\mathbb{Q}$, real $\mathbb{R}$ and complex $\mathbb{C}$ numbers with the *standard* multiplication $(\cdot)$ and addition $(+)$. Now it is possible to define a vector space:

**Definition A.2** *Let $K$ be a field. A vector space $V$ (over $K$) is a set with an addition $+ : V \times V \to V$ and a scalar multiplication $\cdot : K \times V \to V$ fulfilling the following rules:*

*(A1)* $\forall a, b, c \in V : a + (b + c) = (a + b) + c$
*(A2)* $\forall a, b \in V : a + b = b + a$
*(A3)* $\exists! 0 \in V : a + 0 = 0 + a = a$
*(A4)* $\exists! (-a) \in V : a + (-a) = 0$
*(SM1)* $\forall a \in V, x, y \in K : x \cdot (y \cdot a) = (x \cdot y) \cdot a$
*(SM2)* $\exists! 1 \in K \; \forall a \in V : a \cdot 1 = 1 \cdot a = a$
*(SM3)* $\forall a \in V, \forall x, y \in K : a \cdot (x + y) = a \cdot x + a \cdot y$
*(SM4)* $\forall a, b \in V, \forall x \in K : (a + b) \cdot x = a \cdot x + b \cdot x.$

An example is the set $K^n$ of all $n$-tuples with component wise addition and the scalar multiplication.

Let $u_1 \dots u_m$ be vectors of a $K$-vector space $V$. The vector $r \in V$ is a *linear combination* of $u_1 \dots u_m$ iff $\exists a_1 \dots a_m \in K : r = a_1 u_1 + \dots + a_m u_m$. Let

$$\langle \{u_1 \dots u_m\} \rangle := \{\sum_{i=1}^{m} k_i u_i | k_1 \dots k_m \in K\}, \tag{1.1}$$

the set of all possible linear combinations of $\{u_1 \dots u_m\}$. Then $\langle \{u_1 \dots u_m\} \rangle$ is also called the *generating system* of $\{u_1 \dots u_m\}$. A minimal set, which generates a vector space leads to the *basis*:

**Definition A.3** *Let $V$ be a $K$-vector space, and $u_1 \dots u_m \in V$. If $\langle \{u_1 \dots u_m\} \rangle =$*

*$V$ and $u_1 \dots u_m$ a linear independent set. Then $\{u_1 \dots u_m\}$ is called a basis of $V$.*

A vector space $V$ equipped with a special bilinear form leads to the *Euclidean space*:

**Definition A.4** *An Euclidean space is a real vector space $V$ equipped with a symmetric bilinear form $\phi : V \times V \to \mathbb{R}$ that is positive definite. More explicitly, $\phi$ satisfies the following axioms:*

*(A1)* $\phi(u_1 + u_2, v_1) = \phi(u_1, v_1) + \phi(u_2, v_1)$
*(A2)* $\phi(u_1, v_1 + v_2) = \phi(u_1, v_1) + \phi(u_1, v_2)$
*(A3)* $\phi(\lambda u, v) = \lambda\phi(u, v) = \phi(u, \lambda v)$
*(A4)* $\phi(u, v) = \phi(v, u)$
*(A5)* $u \neq 0 \Rightarrow \phi(u, u) > 0.$

The bilinear form $\phi(u, v)$ is also called *inner product* or *scalar product* of $u$ and $v$. The inner product is used to define a *norm* $\|.\|$ on vectors,

$$\|u\| := \sqrt{\phi(u, u)}. \tag{1.2}$$

In most cases matrix and vector calculus is used to handle computer vision and robotic tasks. But since in many applications special transformations like the Euclidean and projective ones are involved, the classical Euclidean vector calculus becomes difficult and extensions are necessary.

In [62] examples can be found which explain the *need of an origin*. Since I am also interested in relations of points, points and directions must be distinguished. This leads directly to the definition of an *affine space*:

**Definition A.5** *An affine space is either a degenerated space, reduced to the empty set, or a triple $\langle V, \vec{V}, + \rangle$ consisting of a nonempty set $V$ (of points), a vector space $\vec{V}$ (of translations, or directions), and an action $+ : V \times \vec{V} \to V$, satisfying the following conditions:*

*(A1)* $\forall a \in V : a + 0 = 0 + a = a$
*(A2)* $\forall a \in V, u, v \in \vec{V} : a + (u + v) = (a + u) + v$
*(A3)* $\forall a, b \in V, \exists! u \in \vec{V} : a + u = b.$

The unique vector $u \in \vec{V}$, such that $a + u = b$ is sometimes denoted with $u = ab$, or $u = b - a$. Thus $b$ can be written as $b = a + ab$.

A mapping from one affine space to another one can be described by an *affine transformation*. It can be shown [62] that affine transformations can be described by a linear transformation in the following way:

**Lemma A.1** *Let* $\langle V, \vec{V}, + \rangle$ *and* $\langle V', \vec{V}', +' \rangle$ *be two affine spaces.* $\forall a \in V, b \in V', h : \vec{V} \to \vec{V}'$ *is* $f : V \to V'$ *with* $f(a + v) = b + h(v)$ *an affine transformation, or affine map.*

Proof: See [62].

**Definition A.6** *Let* $V$ *be a vector space over* $K$*. The projective space* $P(V)$ *is the set* $(V \backslash \{0\})/\sim$ *of equivalence classes of nonzero vectors in* $V$ *under the equivalence relation* $\sim$ *defined such that*

$$\forall u, v \in V \backslash \{0\} : u \sim v \quad \Leftrightarrow \quad \exists \lambda \in \mathbb{R} : v = \lambda u. \tag{1.3}$$

Mathematically a projective space $\mathcal{P}(V)$ is a set of equivalence classes of vectors in $V$. The idea behind projective geometry is to view an equivalence class $(u)_\sim$ as an *atomic* object, forgetting the internal structure of the equivalence class. For this reason it is customary to call an equivalence class $a = (u)_\sim$ a *point* (the entire equivalence class $(u)_\sim$ is collapsed into a single object, viewed as a point). The theory about projective geometry is well studied [123, 32, 62, 48] and has proved useful for many computer vision tasks.

## A.2 Lie groups and Lie algebras

**Definition A.7** *A set $G$ together with a binary operation $\cdot$ defined on elements of $G$ is called a group if it satisfies the following axioms:*

*(A1)* $\forall g_1, g_2 \in G : g_1 \cdot g_2 \in G$
*(A2)* $\exists! e \, \forall g \in G : g \cdot e = e \cdot g = g$
*(A3)* $\forall g \in G \, \exists! \overline{g} : g \cdot \overline{g} = \overline{g} \cdot g = e$
*(A4)* $\forall g_1, g_2, g_3 \in G : (g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.

For example the groups $O(n)$ and $SO(n)$ are defined as

$$O(n) := \{R \in \mathbb{R}^{n \times n} : RR^T = I, det(R) \in \{+1, -1\}\}, \quad (1.4)$$
$$SO(n) := \{R \in \mathbb{R}^{n \times n} : RR^T = I, det(R) = +1\}. \quad (1.5)$$

The notation $O$ abbreviates *orthonormal* and the notation $SO$ abbreviates *special orthogonal* and denotes the space of rotation matrices. For $n = 3$, $SO(3)$ is a group under the operation of matrix multiplication. The group action rotates one 3D point with respect to the origin to another 3D point.

Another common way to introduce 3D rotation matrices is by introducing Euler angles. Rotations with amount $\psi$, $\phi$ and $\theta$ around the three coordinate axes $x$, $y$ and $z$ can be expressed by the matrices

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{pmatrix}, \quad (1.6)$$

$$R_y(\phi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix}, \quad (1.7)$$

$$R_z(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (1.8)$$

Concatenation of these three matrices leads to general rotations and therefore elements of $SO(3)$.

An extension is the group of *rigid body motions*, which is nothing more than a special affine map:

**Definition A.8** *The set of affine maps $\rho$ of $K^n$, defined such that $\forall X \in V$ : $\rho(X) = RX + t$, with $R \in SO(n)$ a rotation matrix, and $t \in K^n$ a translation vector , is a group under composition called the group of direct affine isometries or rigid motions denoted by $SE(n)$.*

A rigid motion of an object is a continuous movement of the particles in the object such that the distance between any two particles remains fixed at all times [124].

One way to distinguish between directions and points in a vector space is to introduce a fourth coordinate (a *homogeneous coordinate*) for the origin. A 3D point can then be described by

$$P = (x_1, x_2, x_3, 1) \tag{1.9}$$

and a direction by

$$d = (x_1, x_2, x_3, 0). \tag{1.10}$$

Then the standard vector addition between a point $P$ and a direction $d$ leads to a new point. Every 3D rigid motion can then be represented by the $4 \times 4$ matrix

$$\boldsymbol{M} = \begin{pmatrix} R_{3\times3} & t_{1\times3} \\ 0_{3\times1} & 1 \end{pmatrix}. \tag{1.11}$$

A rigid motion of a point can be estimated by multiplying the matrix $\boldsymbol{M}$ with the point $P$, $P' = \boldsymbol{M}P$.

**Definition A.9** *A group is an $n$-dimensional Lie group if the set of its elements can be represented as a continuously differentiable manifold of dimension $n$, on which the group product and inverse are continuously differentiable functions as well.*

**Definition A.10** *A linear space is a Lie algebra if a product, the Lie bracket, is associated with each pair of Elements $F$, $G$ and $H$ such that*

| | | | | |
|---|---|---|---|---|
| *(A1)* | $[G, H]$ | $=$ | $-[G, H]$ | |
| *(A2)* | $[\alpha G, \beta H]$ | $=$ | $\alpha\beta[H, G]$ | $\forall \alpha, \beta \in \mathbb{R}$ |
| *(A3)* | $[F, [G, H]] + [G, [H, F]] + [H, [F, G]]$ | $=$ | $0$. | |

For a one parameter Lie group given in matrix representation, the tangent space in $\mathbb{R}^{n\times n}$ of the group manifold at the identity defines a Lie algebra.

The groups $SO(3)$ and $SE(3)$ are *Lie groups* and their *Lie algebras* turn out to be $so(3)$ or $se(3)$, with

$$\begin{aligned} so(3) &= \{A \in \mathbb{R}^{3\times 3} | A = -A^T\} && (1.12)\\ se(3) &= \{(v,\omega) | v \in \mathbb{R}^3, \omega \in so(3)\}. && (1.13) \end{aligned}$$

These Lie groups and Lie algebras are connected via the exponential function. To motivate this, the velocity of a point $p$ attached to the rotating body is considered. If the body rotates at constant unit velocity about the axis $\omega$, the velocity of the point $\dot{q}$ may be written as

$$\dot{q}(t) = \omega \times q(t) = \hat{\omega} q(t), \quad (1.14)$$

with the definition

$$\hat{\omega} := \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \quad (1.15)$$

This matrix can be interpreted as the tangents of the Euler matrices at the identity, since

$$\begin{aligned} \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_1 \\ 0 & \omega_1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & \omega_2 \\ 0 & 0 & 0 \\ -\omega_2 & 0 & 0 \end{pmatrix} + \\ &\quad \begin{pmatrix} 0 & -\omega_3 & 0 \\ \omega_3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} && (1.16)\\ &= \frac{\partial R_x}{\partial \psi}(0) + \frac{\partial R_y}{\partial \phi}(0) + \frac{\partial R_z}{\partial \theta}(0). && (1.17) \end{aligned}$$

Equation (1.14) is a time-invariant linear differential equation which may be integrated to give

$$q(t) = \exp(\hat{\omega} t) q(0), \quad (1.18)$$

where $q(0)$ is the initial ($t = 0$) position of the point and $\exp(\hat{\omega} t)$ is the matrix exponential

$$\exp(\hat{\omega} t) = \sum_{k=0}^{\infty} \frac{(\hat{\omega} t)^k}{k!}. \quad (1.19)$$

It follows that if a point is rotated around the axis $\omega$ at unit velocity $\theta$ units of time, then the rotation matrix is given by

$$R(\omega, \theta) = \exp(\hat{\omega}\theta). \tag{1.20}$$

An extension of 3D rotations around the origin are 3D rigid motions. The group of 3D rigid motions is denoted as $SE(3)$. Its Lie algebra is

$$se(3) = \{(v, \omega) | v \in \mathbb{R}^3, \omega \in so(3)\}. \tag{1.21}$$

An element $\xi = (v, \omega) \in se(3)$ is called a twist and its matrix representation takes the form

$$\hat{\xi} = \begin{pmatrix} \hat{\omega} & v \\ 0_{3\times 1} & 0 \end{pmatrix}. \tag{1.22}$$

The exponent $\exp(\theta\hat{\xi})$ leads to a rigid motion and corresponds to a screw motion. The twist concept is also explained in chapter 3.3.6 by using geometric algebra.

To calculate the group action from the twist, the Rodrigues' formula can be applied by computing

$$\exp(\hat{\xi}\theta) = \begin{pmatrix} \exp(\theta\hat{\omega}) & (I - \exp(\hat{\omega}\theta))(\omega \times v) + \omega\omega^T v\theta \\ 0_{3\times 1} & 1 \end{pmatrix}, \omega \neq 0, \tag{1.23}$$

with $\exp(\theta\hat{\omega})$ given as

$$\exp(\theta\hat{\omega}) = I + \hat{\omega}\sin(\theta) + \hat{\omega}^2(1 - \cos(\theta)). \tag{1.24}$$

The reader should consult [62, 124] for further information about Lie groups and their relations to Lie algebras.

## A.3 The pinhole camera model

The common way to model perspective projections on an image plane is to use the *pinhole camera* model. In this section, the ideas and concepts will be scratched. The reader can find a more detailed introduction in [47]. The pinhole camera model is visualized in figure A.1. A geometric model of the

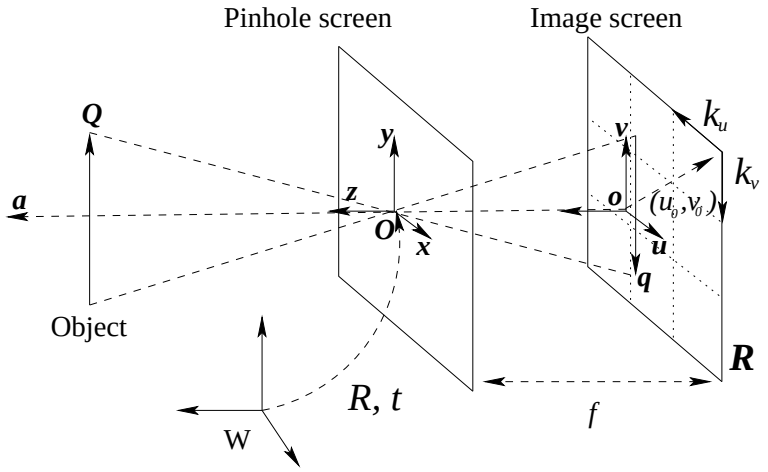



Fig. A.1: The pinhole camera model

pinhole camera consists of the plane $\boldsymbol{R}$, called the *retinal* plane and the point $\boldsymbol{O}$, the *optical center*, located at distance $f$, the *focal length* of the optical system. The *optical axis* $\boldsymbol{a}$ is the line perpendicular to $\boldsymbol{R}$ and intersecting $\boldsymbol{O}$.

It is now possible to attach on $\boldsymbol{O}$ a three-dimensional space $(\boldsymbol{O}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, with $\boldsymbol{x}$ and $\boldsymbol{y}$ orthonormal and parallel to $\boldsymbol{R}$, $\boldsymbol{z}$ perpendicular to $\boldsymbol{R}$ and one two-dimensional space $(\boldsymbol{o}, \boldsymbol{u}, \boldsymbol{v})$ for the retinal plane. This is the so called *standard coordinate system* of the camera. The relationship between image coordinates $(u, v)$ and 3D coordinates $(x, y, z)$ can then be written as

$$-\frac{f}{z} = \frac{u}{x} = \frac{v}{y}, \tag{1.25}$$

which can be written in projective coordinates as

$$\begin{pmatrix} U \\ V \\ S \end{pmatrix} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{1.26}$$

where

$$u = \frac{U}{S} \quad , \quad v = \frac{V}{S} \qquad : S \neq 0. \tag{1.27}$$

A camera can be considered as a system that performs a linear projective transformation from the projective space $\mathcal{P}^3$ to the projective plane $\mathcal{P}^2$.

This idealized model must now be adapted to a scenario in which the world and retinal coordinate systems vary. So far, the image coordinate system is centered at the intersection of the optical axis $\boldsymbol{a}$ with the retinal plane $\boldsymbol{R}$. In an image, the coordinate system is often attached to one corner and will sometimes have different units on both axes due to the electronics of acquisition. The transformation from a point given in the retinal plane to a pixel point can be modeled by a transformation matrix $K$, which contains scaling parameters along the axis and a translation,

$$K = \begin{pmatrix} -fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.28}$$

The projection matrix $P$ can now be written as

$$P = \begin{pmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.29}$$

It can be proceeded similarly with the world coordinate system, which is so far attached to the optical center $\boldsymbol{O}$. The transformation of another world coordinate system (e.g. $\boldsymbol{W}$ in figure A.1) to the one attached at the optical center can be described by a rigid body motion $M$, containing a rotation matrix $R_{3\times3}$ and a translation vector $t_{1\times3}$. The projection matrix $P$ can now be expressed as

$$P = \begin{pmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R_{3\times3} & t_{1\times3} \\ 0_{3\times1} & 1 \end{pmatrix} \tag{1.30}$$

$$= K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} M. \tag{1.31}$$

The components of the matrices $K$ and $M$ are called the *intrinsic* and *extrinsic* camera parameters, respectively. Camera *calibration* is the process of estimating the intrinsic and extrinsic parameters of a camera. Estimation of

both matrices $K$ and $M$ explicitly is called *explicit* calibration, whereas estimating of the projection matrix $P$ (without its decomposition in $K$ and $M$) is called *implicit* calibration [180]. In this thesis only the projection matrix $P$ is used and is not further separated in its intrinsic and extrinsic components. Therefore in the scenario of pose estimation only implicitly calibrated cameras are assumed.

## A.4 Signal theory

This section deals with signal theoretic foundations. The aim is to define the discrete Fourier transformation and its extension to the 3D space in classical matrix calculus and by using complex numbers. More detailed information can be found in [96, 91]. Chapter 7 introduces the discrete Fourier transformation in terms of geometric algebra. It allows for an extension of cycloidal curves to $n$-times nested cycloidal curves which are generated by 3D Fourier descriptors. This links signal theory and geometry and is applied advantageously for 2D-3D silhouette based free-form pose estimation, see chapter 7.2.

### A.4.1 Foundations

The aim of this part is to clarify the basic definitions used in signal theory. This part is no complete introduction and many concepts (like the Dirac impulse, etc.) and other important theorems are neglected, since they are not used in this thesis. Periodic functions build the basis of Fourier transformations:

**Definition A.11** *A function $f : \mathbb{R} \to \mathbb{R}$, is called a periodic function, iff*

$$\exists p \in \mathbb{R}^+ : \forall x \in \mathbb{R} \quad : \quad f(x+p) = f(x). \tag{1.32}$$

The value $p \in \mathbb{R}^+$ is also called the *period* or *period length* of $f$. As an example, the sine-function has the period $2\pi$, since $\sin(x + 2\pi) = \sin(x)$. This means, $\sin(x)$ is a $2\pi$-periodic function. If $p$ is the smallest number with property (1.32), it is called the *primitive period* of $f(x)$. For example $\sin(x)$ has the periods $2\pi m$, $m \in \mathbb{N}$, and the primitive period $2\pi$. If $f(x)$ is $p$-periodic, it is clear, that $\forall a \in \mathbb{R}$, $f(x - a)$ is also $p$-periodic. For the sine and cosine function, the additive value $a$ is also called the *phase* of $\sin(x)$ or $\cos(x)$. Multiplying $x$ with a scalar $\frac{1}{b}$ leads to a change of the period length and

$$g(x) \quad := \quad f\left(\frac{x}{b}\right), \quad b \in \mathbb{R}^+ \tag{1.33}$$

transforms the $p$-periodic function $f$ in a $bp$-periodic function, since

$$g(x+bp) \quad = \quad f\left(\frac{x+bp}{b}\right)$$

$$
\begin{aligned}
&= f\left(\frac{x}{b}+p\right) \\
&= f\left(\frac{x}{b}\right) = g(x). \qquad (1.34)
\end{aligned}
$$

The unit-interval is often defined as

$$I_1 := [-\frac{1}{2}, \frac{1}{2}] \qquad (1.35)$$

and in the notation $\sin(2\pi rx)$ is $r \in \mathbb{R}$ the *frequency* number of periods in $I_1$.

## A.4.2 Trigonometric interpolation

Polynomial interpolation of data sets $(x_i, f_i)$ is well known from numerical mathematics. For example the Lagrange formula or the Hermite interpolation [172] are often used. Trigonometric interpolation is used for the analysis of periodic functions. For a periodic function in the interval $[0, \ldots, M-1]$ and regular sample values $(j, f_j), j \in 0, \ldots, M-1$, the trigonometric interpolation is given in the form

$$\Psi(x) := \sum_{m=0}^{M-1} \left( A_m \cos\left(\frac{2\pi mx}{M}\right) + B_m \sin\left(\frac{2\pi mx}{M}\right)\right). \qquad (1.36)$$

The coefficients $A_m$ and $B_m$ are given as the following series expressions [29],

$$A_m = \frac{1}{M} \sum_{u=0}^{M-1} f_m \cos\left(\frac{2\pi um}{M}\right) \qquad (1.37)$$

$$B_m = \frac{1}{M} \sum_{u=0}^{M-1} f_m \sin\left(\frac{2\pi um}{M}\right). \qquad (1.38)$$

Note, that since the interval $[0, \ldots, M-1]$ is used, the values $A_m$ and $B_m$ must be normalized with $\frac{2\pi}{M}$ within the sine and cosine functions. $A_m$ is also called the cosine-series and $B_m$ the sine-series. In the complex domain the sine and cosine expressions can be summarized in a complex exponential function. This leads directly to the 1D discrete Fourier transformation.

## A.4.3 The 1D discrete Fourier transformation

Assume a set $f_j \in \mathbb{R}$ of values, with $j = 0, \ldots, M-1, M \in \mathbb{N}$. The set of values can be interpreted as elements of a $M$ dimensional vector space $V^M$,

which are orthonormal with respect to the basis vectors $\mathbf{e}_0, \ldots, \mathbf{e}_{M-1}$, and fulfill the property

$$\mathbf{e}_i \cdot \mathbf{e}_j \quad = \quad \delta_{ij}.$$

The $\delta_{ij}$ denotes the *Kronecker* function. Then the values $f_j$ can be re-written as

$$f_j \mathbf{e}_j \quad \in \quad V^M.$$

This is the standard interpretation of the signal space. The discrete Fourier transformation (DFT) performs a base change to another orthonormal base of same length with complex values:

$$F_m \quad = \quad \frac{1}{M} \sum_{u=0}^{M-1} f_u \exp\left(\frac{-2\pi imu}{M}\right).$$

The value $F_m$ is complex valued and can be decomposed in its real and imaginary part as

$$\begin{aligned} Re(F_m) \quad &= \quad \frac{1}{M} \sum_{u=0}^{M-1} f_u \cos\left(\frac{2\pi um}{M}\right), \\ Im(F_m) \quad &= \quad -\frac{1}{M} \sum_{u=0}^{M-1} f_u \sin\left(\frac{2\pi um}{M}\right). \end{aligned}$$

The inverse transformation (IDFT) can be written as

$$f_m \quad = \quad \sum_{u=0}^{M-1} F_u \exp\left(\frac{2\pi ium}{M}\right).$$

Indeed, it can be shown [96], that

$$f(x) \quad = \quad \sum_{u=0}^{M-1} F_u \exp\left(\frac{2\pi iux}{M}\right), \qquad x \in [0, \ldots, M-1]$$

can be interpreted as a trigonometric interpolation and is therefore well suited for smooth function interpolation. It has the advantage that the interpolation function can be derivated in each point in contrast to piecewise linear interpolators. Furthermore it is possible to use low-pass information for approximation. This helps to avoid local minima in iteration processes. The DFT is a coordinate transformation, whose basis vectors are sampled sine

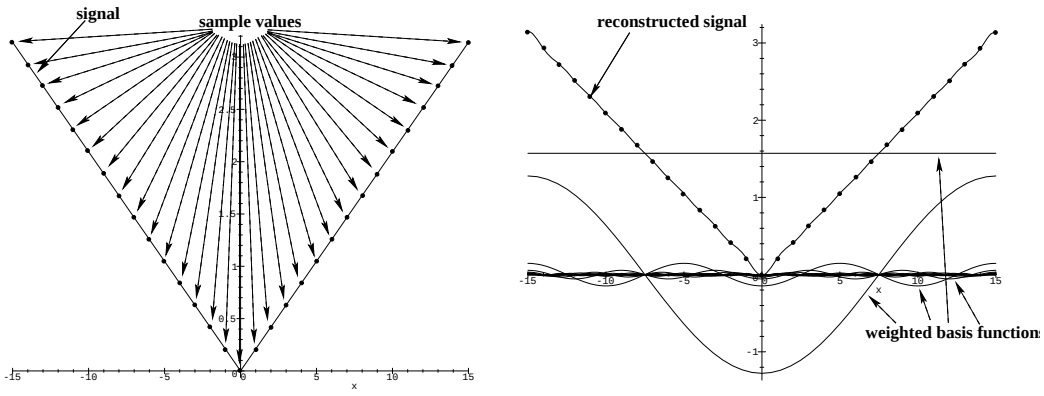



Fig. A.2: Example for DFT and IDFT of a discrete signal: The left image shows the fabs$(x/15) * \pi$-function as a one-dimensional signal. Discrete sample points are also shown. The right image shows the real-part of the weighted basis functions from which the original signal is reconstructed. The reconstructed signal leads to a trigonometric interpolated function of the sample points.

and cosine functions. For short notations, it is also convenient [91] to write for $\boldsymbol{f} = (f_0, \ldots, f_{M-1})$,

$$F_m = \boldsymbol{f} \cdot \boldsymbol{b}_m \quad \text{with} \quad \boldsymbol{b}_m = \frac{1}{M} \begin{pmatrix} 1 \\ \exp(2\pi i m/M) \\ \exp(2\pi i 2m/M) \\ \vdots \\ \exp(2\pi i (M-1)m/M) \end{pmatrix}, \quad 0 \leq m \leq M.$$

Note, that the inner product of two complex vectors is defined as

$$F_m = \boldsymbol{f} \cdot \boldsymbol{b}_m \;\; = \;\; \sum_{n=0}^{M-1} \boldsymbol{f}_n (\widetilde{\boldsymbol{b}_m})_n. \tag{1.39}$$

Geometrically, the basis functions can be interpreted as projections of (different fast) spinning orthogonal phase vectors over the time.

Figure A.2 shows an example for the DFT and IDFT of a signal. The left image shows the fabs$(x/15) * \pi$-function as a one-dimensional signal. Discrete sample points are also shown. The right image shows the real-part of the weighted basis functions from which the original signal is reconstructed by computing its linear combination. The reconstructed signal leads to a trigonometric interpolated function of the sample points.

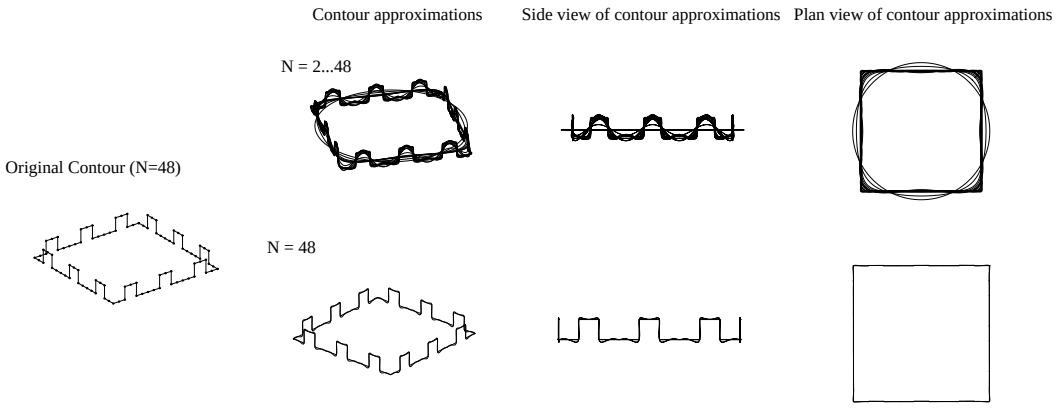



Fig. A.3: 3D contour interpolation and its approximations: A 3D contour (containing 48 points) is successively approximated. For $N = 48$ all contour points intersect with the interpolation function, $N < 48$ leads to a low-pass contour.

For 3D contour interpolation a set $f_j^3 \in \mathbb{R}^3$ of 3D values is assumed with $j = 0, \ldots, M-1$, $M \in \mathbb{N}$. These values can be contour points of a closed contour, as for example shown in figure A.3. To achieve a 3D contour interpolation, the 3D signal can be interpreted as 3 separate 1D signals:

$$F_m^3 \;=\; \frac{1}{M} \sum_{u=0}^{M-1} \begin{pmatrix} f_u^3(1) \\ f_u^3(2) \\ f_u^3(3) \end{pmatrix} \exp\left(\frac{-2\pi i u m}{M}\right),$$

and its inverse transformation can be written as

$$f_u^3 \;=\; \sum_{m=0}^{M-1} \begin{pmatrix} F_m^3(1) \\ F_m^3(2) \\ F_m^3(3) \end{pmatrix} \exp\left(\frac{2\pi i m u}{M}\right).$$

Taking only a subset of the phase vectors leads to a low-pass approximation of the contour. This is applied in chapter 7 to speed up the algorithm for pose estimation of free-form contours and to avoid local minima during iterations. Indeed, for classical matrix calculus, the fusion of complex numbers within twists and Plücker lines is not impossible but it is hard not to loose the oversight. This is the main argument why I use the conformal geometric algebra to formalize the pose scenario in chapter 7. Also extensions to deformable free-form contours become clearer in this language, see e.g. chapter 7.1.2.

## Appendix B

# IMPLEMENTED MODULES OF THE NAVIGATION SYSTEM

Since the initial motivation of the thesis is self localization for robot navigation, this chapter presents modules of a running system I supervised in student projects during the main research. The implementation was mainly done by T.Rabsch [139] and D. Grest [68]. They had to solve several small subproblems to get a running system. As this part contains only little research but mainly engineering and implementation details, the main message of this chapter is to give an idea about outstanding problems for real applications which are mainly neglected or simplified (e.g. image processing, the correspondence problem, process communications, etc.). Furthermore, the aim is to close the loop to the motivation and to sketch other research projects and open problems for further work, which are also discussed in the chapter 8.

Section B.1 starts with the definition of the tracking problem for feature based pose estimation within the navigation scenario. Then the image processing for the navigation scenario will be explained, and the chapter will end up in the behavior based system architecture and experiments with the navigation system.

## B.1 The tracking problem

Tracking means to trace an object in an image sequence. This tracing can happen in the pure image (2D-2D tracking), e.g. by using optical flow techniques [12], or the tracing can happen in the 3D space in combination with known 3D objects.

In this thesis *matching* means to minimize a matching error by solving

two problems:

1. The correspondence problem: To determine the mapping between model elements (here 3D model lines) and image features.

2. The spatial fitting problem (pose estimation): To determine the best parameters (here rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$) for each correspondence, so that the spatial fit error of the model lines to the image lines is minimized.

To estimate the motion of an object, it is necessary to establish the correspondences between the image features and the object features. If a correspondence can be assumed, it can be used to estimate the pose and vice versa, the error of the pose can be used to evaluate the correspondences. An interaction between the correspondence problem and the pose estimation problem can be identified and is visualized in figure B.1.

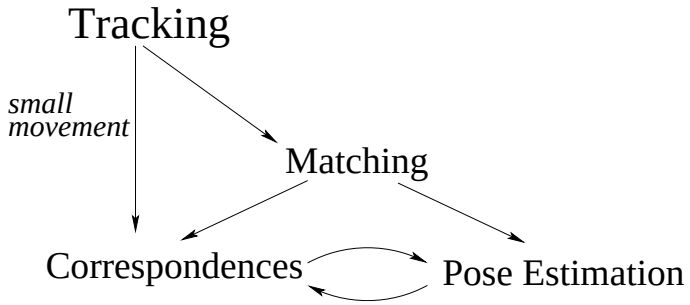
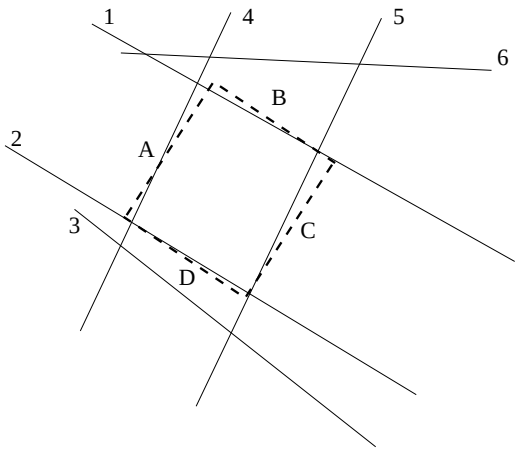



Fig. B.1: The matching problem as interaction of the correspondence problem and the pose estimation problem.

First of all the use of the tracking assumption within the navigation scenario is motivated. Then it will be continued with the matching problem which contains the correspondence problem and its interaction with the pose estimation problem.

### B.1.1 The tracking assumption

Often the property is used, that the motion of an object between two images is *small*, if the images are taken in a short time and the object is not too far away and moving sufficiently slow. If such a situation is assumed, the observation of the object movement between two images is not large. This assumption




| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| A | ✕ | ✕ | ✕ | ■ | | ✕ |
| B | ■ | | | ✕ | ✕ | |
| C | ✕ | ✕ | ✕ | | ■ | ✕ |
| D | | ■ | | ✕ | ✕ | ✕ |

Fig. B.2: Matching example for a rectangle. The model lines are labeled with letters and the extracted image lines are labeled with numbers. The table indicates the correspondence space with the allowed possibilities (white/black), the impossible matches (cross) and the current match (black).

is often made for optical flow techniques and the search window for image features is mostly set to a few pixels in the image. Indeed it is possible to take images while driving the robot but the main problem is that the landmark is not always in the field of view during the drive. Therefor, it is futile to use images which are taken in short time intervals. Instead it is possible to make use of the movement commands of the mobile robot: The odometric data is translated in terms of the object coordinate system and used to make a prediction of the object in the image. This coordinate transformation is explained in detail in [68]. Accumulating the odometric data leads to increasing errors which are corrected by using the pose estimation algorithm. The advantage of the prediction is that it is possible to reduce the search space for the correspondence problem by assuming boundary conditions.

### B.1.2 The matching algorithm

It is well known, that for $l = m \times n$ potential pairs, there are $S = 2^{|l|}$ correspondences. This means that the search space is in general very large and not practicable for applications.

The tracking assumption allows to use local criteria like distances and angles to reduce the search space significantly, depending on the error bound-

aries. In this context the correspondence space for $m$ model lines and $n$ image lines is represented by a $m \times n$ fit-matrix. In this matrix flags represent the needed information for a match, mismatch or potential match, figure B.2 shows an example. In this example the model lines are labeled with letters and the extracted image lines are labeled with numbers. The table indicates the correspondence space with the allowed matches (white/black), the impossible matches (cross) and the current match (black). See also [18] for further information.

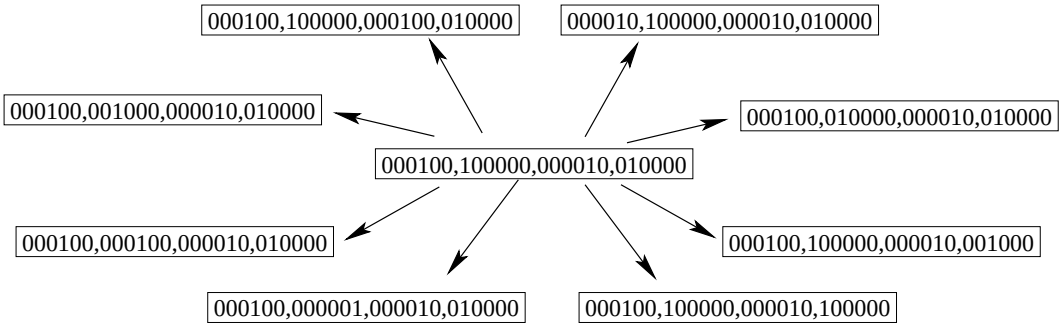



Fig. B.3: Visualization of Hamming distance 2 (HD2) neighbors of a bit string.

Random start local search [18] is an effective and easy strategy for solving optimization problems. The basic algorithm contains the following steps:

```
1. (a) Image Processing:
          Hough transformation to extract 2D image lines.
          Reconstruct 2D image lines to 3D planes.
   (b) Generate fit-matrix F and reduce search space.
   (c) Choose a random start configuration.
2. Estimate pose for the hypothetic correspondences and
   estimate the match error EMatch(F).
3. Choose neighbor F' and estimate its match error EMatch (F').
   If EMatch(F') < EMatch(F) choose F' as actual match.
   Check Termination condition, if required
   terminate the algorithm.
   Goto (2)
```

In short notation, the aim is to estimate

$$F : \forall F' : E_{match}(F) \leq E_{match}(F').$$

In this algorithm, the Hamming distance $n$–neighborhood [18] is used to define neighbors between fit matrices. Therefor the current match is represented as bit string by encoding the $n \times m$ fit-matrix as $n^m$-binary vector, with the value 1 for a current match and 0 for no match. The Hamming distance (HD) is now the number of different bits, or equivalently the weight of the XOR-combination of two binary strings of the same length. Figure B.3 visualizes some HD2 neighbors of the bit string generated from the example in figure B.2.

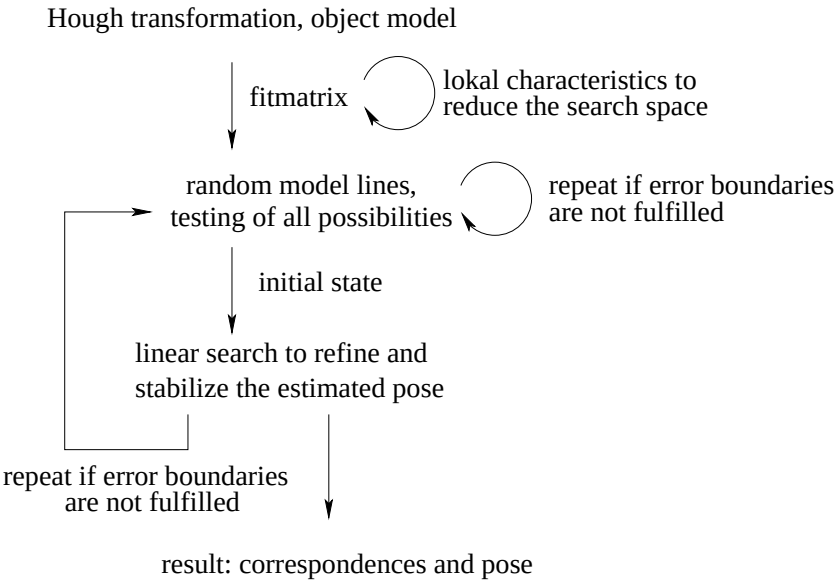



Fig. B.4: A scheme of the tracking algorithm.

For the pose estimation problem a modified version of the random start local search [18] algorithm is chosen. The algorithm is summarized in figure B.4. The principle of the heuristic relies upon a combination of iterative improvement and random sampling. Iterative improvement refers to a repeated generate-and-test principle by which the algorithm moves from an initial state to its local optimum. The algorithm itself consists of two main steps: Firstly find an initial state for a minimum of correspondences and then refine the result by the other correspondences. For the first step a few (mostly five till eight) random model features are chosen and then all combinations of the object features to the allowed image features are used. Taking the pose error as error function leads to an optimal pose. This is possible since the error measure corresponds directly to the Hesse distance and leads to a suitable 3D error measure. Once the initial pose is estimated, the algorithm continues with a linear search for the other object features. Therefor

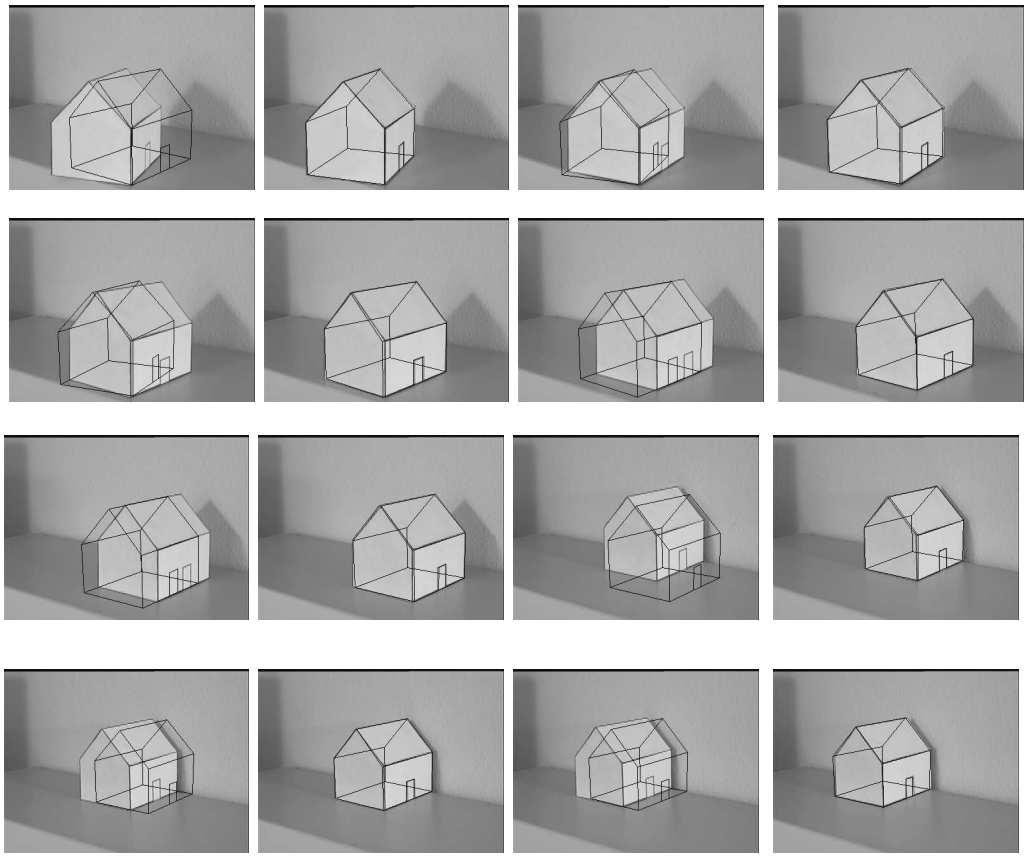


Fig. B.5: Tracking of a model house with partially covered and invisible object lines.

one object feature is added and the whole match error is estimated for all allowed image lines. Dependent on the error boundaries, the object feature is neglected or added for the whole pose. Then the algorithm continues with another object feature. Note: The second part of the algorithm is linear and converges very fast. This is possible because the algorithm deals with infinite extended Hough lines and not with finite line segments as in [18]. Once a correct initial guess is found, all mismatches distort the pose and lead to remarkable error peaks.

This algorithm can be repeated (with other starting configurations) and the more time the algorithm gets, the higher is the probability to reach a global minimum. In the navigation system it is possible to define how often the robot has to localize its position (mostly after 30 seconds). This *update*-time is used as computing time for the mobile robot and the correspondences are estimated during the movement of the robot.

This means that the matching algorithm has indeed no real-time capability. Yet it is possible to overcome this problem in the parallel system architecture which is explained in detail later. Though the algorithm is slow, the results are sufficient for the navigation scenario and also robust, since it is possible to neglect object features if they are not visible in the image. Figure B.5 shows an example. Indeed the matching algorithm is not perfect and the use of infinite extended image lines does sometimes lead to only a small match error but a wrong pose. Examples are given in figure B.6. Fortunately these problems can be avoided in the navigation system since it is possible to use the stereo camera system and compare the pose results of both images with each other. This enables the algorithm to find falsely estimated poses and neglect them, if necessary.

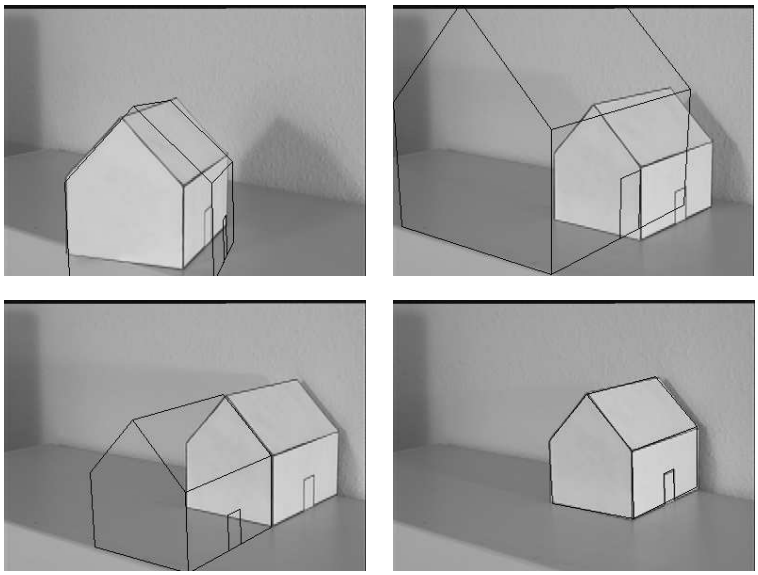


Fig. B.6: Falsely estimated poses in comparison to a rightly estimated pose. The errors occur because of the use of infinite extended Hough lines. The error of the tracking algorithm is very small, but the pose is *wrong*. To understand the error pose, extend the image lines to the infinite extended lines and compare them with the object model.

### B.1.3 Image feature extraction

This section deals with the image processing which leads to the used image features. Image processing means to reduce the image information to task-dependent image primitives. Image primitives can be

- Corners or wedges.
- Lines or line segments.
- Frequency information like Fourier components or Gabor wavelet responses.

For the autonomous navigation system image lines will be extracted and used. To extract lines in an image the well known Hough transformation [88] is applied. The idea is that specific arrangements of gray value edges are voting for certain analytic shapes, e.g. straight lines or ellipses. To search for lines the polar form of image lines is used,

$$l(r,\theta) = x\cos(\theta) + y\sin(\theta) - r \quad = \quad 0. \tag{2.1}$$

The polar form of image lines interprets the line as the normal angle $\theta$ of the line to the origin and the perpendicular distance $r$ of the origin to the image line.

The image is at first smoothed with a binomial filter. Then the gradient image is estimated and a threshold segmentation leads to a binary gradient image.

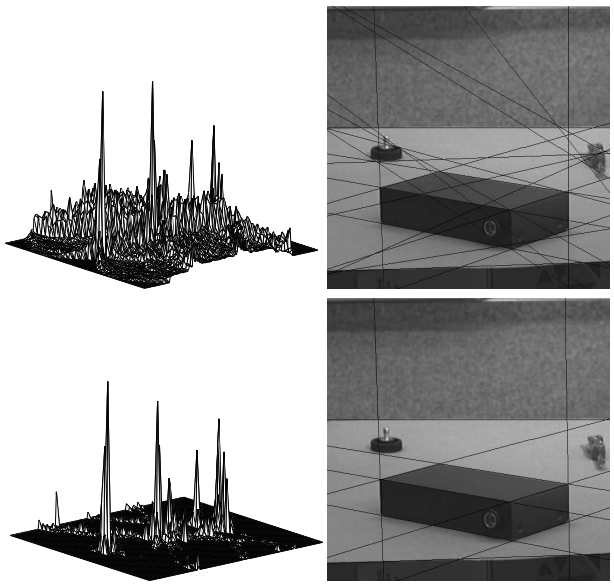


Fig. B.7: Standard-Hough-transformation and orientation selective Hough-transformation.

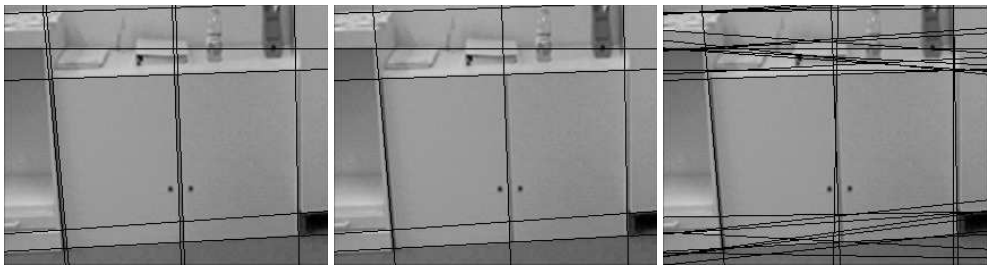


Fig. B.8: Representative Hough lines extracted by different methods.

Every such gradient point $p = (x, y)$ votes for all parameters $(\theta, r)$. These are then accumulated in an accumulator array. The robustness of the Hough transformation can be increased by using not only information about the presence of edges but by also checking the agreement of lines and local orientation, i.e. by applying the orientation selective Hough transformation [137]. The Hough transformation results in an accumulator array (see figure B.7) from which the representative lines show up as peaks. These are easily detectable for *simple* images such as the one in figure B.7 but difficult to extract in more complex situations.

To avoid the extraction of additional lines caused by locally neighbored peaks in the accumulator array (often occurring in the presence of noise in the image data) usually some kind of metric on the accumulator array is defined to allow only lines corresponding to peaks with certain distance. A problem of these methods is that important lines may have small distance in the Hough space (see e.g. narrow parallel lines in figure B.8). To extract the significant lines also information about the areas which do support lines are used, i.e. by evaluating image information. This allows to extract lines with small distance in the accumulator array which are usually not extractable by other methods. The algorithm itself was developed by M. Ackermann (for details see [1]).

Figure B.8 shows extracted Hough lines using different kinds of metrics. In the left image, the proposed method in [100] has been used. In the middle image a neighborhood in the accumulator array is set to zero for each selected peak (as, e.g. in [127]) while in the right image connected areas which occur after thresholding the accumulator array are treated as one line (as e.g. in [104]). Note that the narrow parallel lines could only be extracted by the method described in [100]. The procedure used in the middle image extracts the most significant lines but not the narrow parallel lines because the corresponding peaks are too close in the accumulator array. The procedure used for the right image has great difficulties with locally neighbored peaks which

are above threshold. In [68] D. Grest made several experiments, which kind of Hough transformation and maximum estimation in the Hough array is the easiest and fastest one for the robot navigation scenario.

## B.2 The hardware setup

As hardware setup a B21 mobile robot from Real World Interfaces (RWI) [159] is used. The robot is able to rotate with a maximum of 160 de-

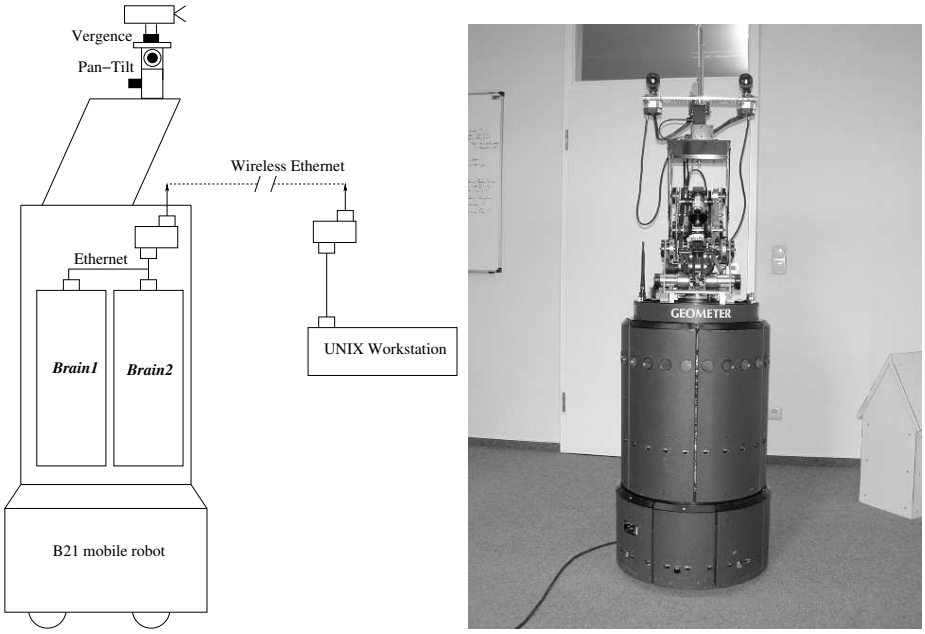



Fig. B.9: The components and a frontal view of the B21 mobile robot.

gree/second and to translate with a maximum of 1 meter/second. There are two Intel Pentium 3, 800 MHz computers (*Brain1* and *Brain2*) on board, equipped with Matrox frame grabber cards, connected with two color CCD cameras. The onboard computers are connected via a local network and can be accessed by a wireless Ethernet from a UNIX workstation. The local network is running with 10MBit/s and the wireless Ethernet with 2 MBit/s. The wireless Ethernet is only used for programming the robot. All image processing and robot controlling is done with Brain1 and Brain2 online. See figure B.9 for an overview. The cameras can be manipulated by a pan-tilt unit for both cameras and a vergence unit for each camera. The images are taken by a synchronized Genlock-interface and are digitized by the frame

grabber cards. The resolution of the cameras is $768 \times 576$ pixel in PAL resolution and the cameras do not have zoom. In the experiments the resolution $384 \times 288$ pixel is used and the focal length is set on $2m$. The field of view is approximately $30^0$ degree. The software package to control the robot is called *beesoft* [159] and contains different client-server programs to access the B21 hardware. These servers are running as independent processes and call back functions can be defined which enable an asynchronic function poll for different modules. This software package is used for the behavior based system architecture.

### B.2.1 Behavior based robotics

In a *knowledge* based robotic system the sensor data are mapped on an abstract model of its environment. A planning device determines the following action and transfers these information to an execution device. The planning device is the *intelligence* of the whole system. The design of knowledge based systems in general is done in a *top-down* strategy. Different modules, which solve partial problems of the system are implemented and are running sequentially. See figure B.10 for the difference of a sequential and parallel system architecture. The main problem of sequentially arranged modules is, that the path from the sensors to the effectors is quite long since it passes through every module. All modules must be complete and working before any behavior is produced. Furthermore the sensor processing and world modeling modules must extract all the information the robot may ever need, even when the necessary information is much easier to obtain. Another problem is reliability: A processing chain is only as strong as its weakest link. If only one module fails, the whole behavior can be wrong.

Behavior based robotics were developed as fusion of computer science, neuro science and neurophysiology. The aim is a robust and adaptable system for an environment which can not be modeled and is changing rapidly. In [166] four requirements are identified:

1. Competence: Instead of implementing a central *intelligence*, the goal is to gain competences. Competences are organized as levels containing different classes of behaviors.

2. Situatedness: The robotic system uses the sensor information directly and not an abstraction of the sensor data.

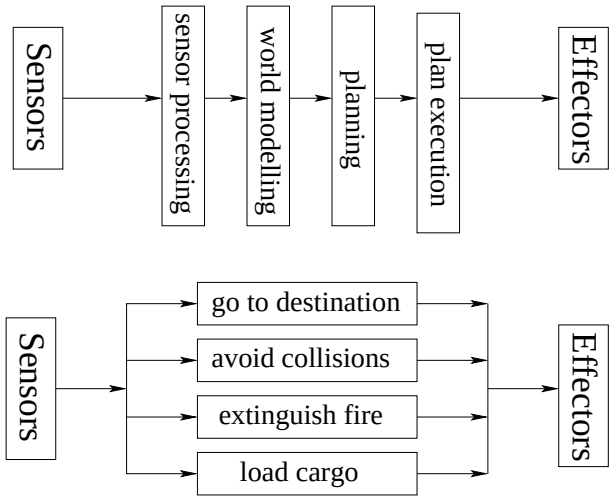



Fig. B.10: Top: Decomposition of a control system into functional modules. Bottom: Decomposition of a control system as task-achieving behaviors.

3. Corporeality: The robotic system can only show behavior, which corresponds to its physical properties. The robot is able to interact with its environment and the sensors are able to notice the robot's action.

4. Emergence: The interaction of different competences leads to the observable *intelligence* of the whole system.

Brooks presented in [27] the *subsumption architecture* as an example for behavior based robot systems. In contrast to the sequential architecture, the aim is a decomposition of the problem into different behaviors. These behaviors all get the original sensor data and can all send movement commands to the motors. This parallel architecture is also shown in figure B.10.

## B.3 The navigation system

Figure B.11 summarizes the different independent processes and their communication possibilities in the implemented navigation system. First, there is the *Navigator*, which is organizing the position update of the robot and is communicating with the *Pilot* and the *TrackPose* module. Then there is the *Pilot*, organizing the path and sending movement commands to the hardware interface. The collision server is allowed to cover the movement commands if there is an obstacle in the way of the path. All these modules are running in

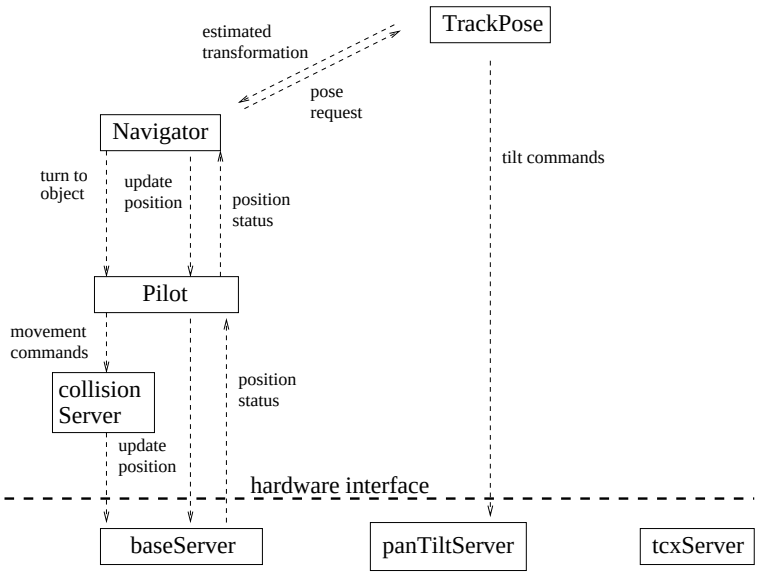



Fig. B.11: The implemented system architecture.

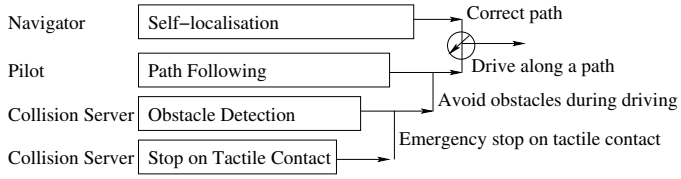



Fig. B.12: The layered control system.

parallel processes and are independent of the hardware. They communicate by different autonomous running servers with the hardware itself.

Figure B.12 shows the control system of the robot. The lowest layer is the test of tactile contact through bump sensors. If the robot hits an obstacle or a wall, an emergency stop is performed to protect the hardware and environment against damages. The second control layer is the obstacle detection and avoidance. It can cover the movement commands and uses an ultrasonic based potential field approach [90] to avoid obstacles on the one hand and to reach a goal on the other hand. The third layer contains path planning and driving to reach a goal. The highest layer contains the self-localization module to correct the navigation errors. The self-localization module is the most complex behavior in the system design. The two last layers are in conflict with each other. This means, the system can not turn to the object to perform a self-localization and at the same time follow a path. Therefore a time dependent finite state machine decides which behavior will

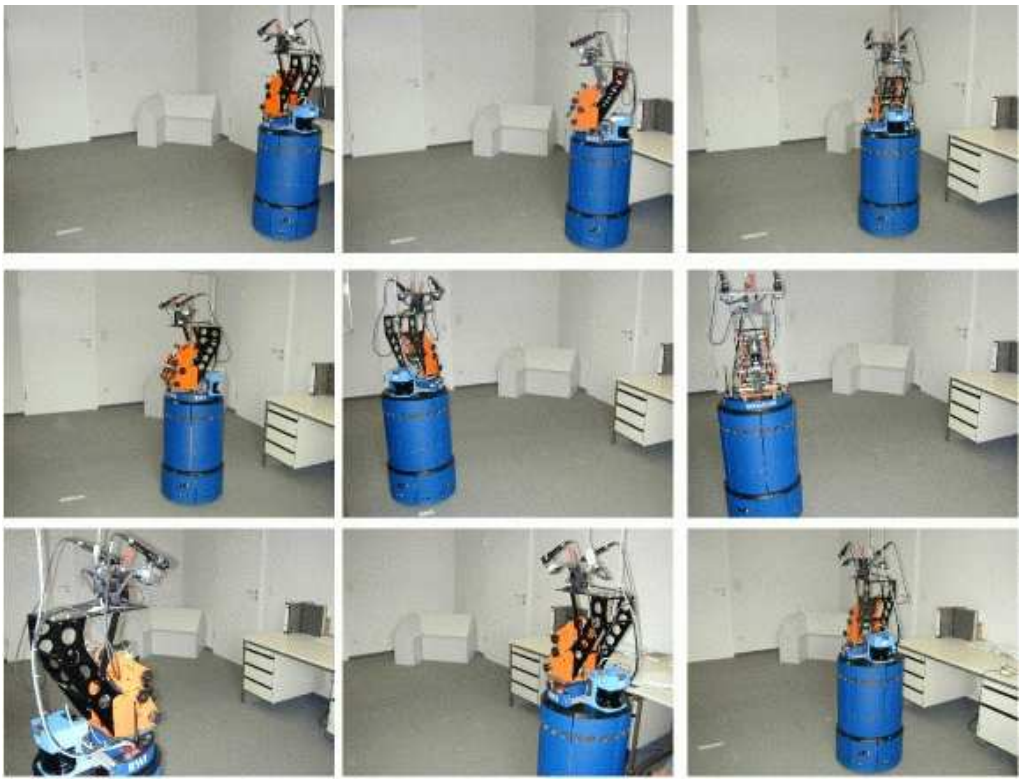


Fig. B.13: The moving robot

be suppressed and which one is active.

## B.4 Experiments with the navigation system

Figure B.13 shows images of the moving robot. The robot has to reach a goal on a given path in the object coordinate system. This path is first translated into robot movement commands. Since the movements of the robot are not exact, the *Navigator* gives the pilot (after some movements) the command to interrupt the current path, turn to the landmark, take images, return to the path and continue the motion. Then the *Navigator* gives a pose request to the *TrackPose* module (to process the actually taken images of the landmark) and uses its results to correct the driven path. Figure B.14 shows screen shots of the vision module, the result of the image feature extraction and the pose estimation before and after the tracking process. These results are used to update and correct the real position of the robot as shown in figure B.15. In the left there is a comparison of the planned path with the really

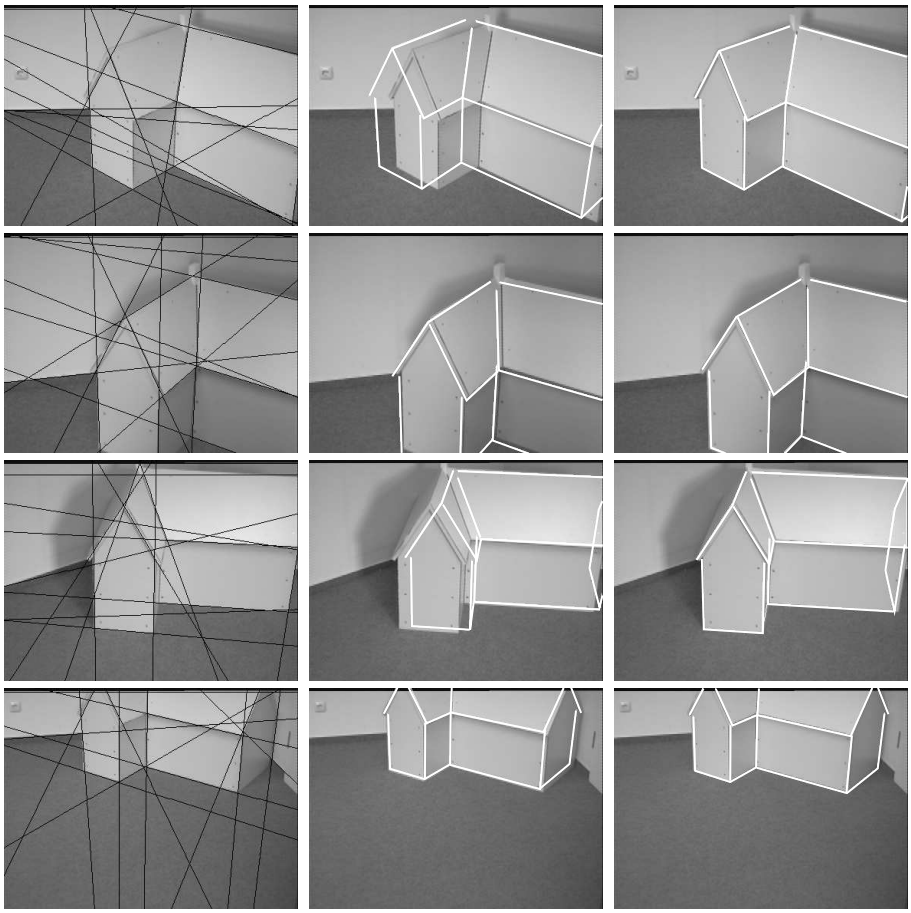


Fig. B.14: The image processing steps during the navigation: The left column shows images with extracted Hough lines. The middle column shows the projected object model before matching and the right column shows the projected transformed object model after matching.

driven path and its increasing odometric errors. The path itself consists of a square, which is driven several times. The right image of figure B.15 shows the accumulated angle differences between the planned path to the really driven path, and between the really driven path and the visually corrected path. The accumulated odometric data result in increasing error and, with increasing time, will lead to non-tolerable results. Instead, the visual error correction leads to a nearby constant error function and, thus, to much better and more stable results. The level of the corrected error function is dependent on both the calibration quality and the match error.

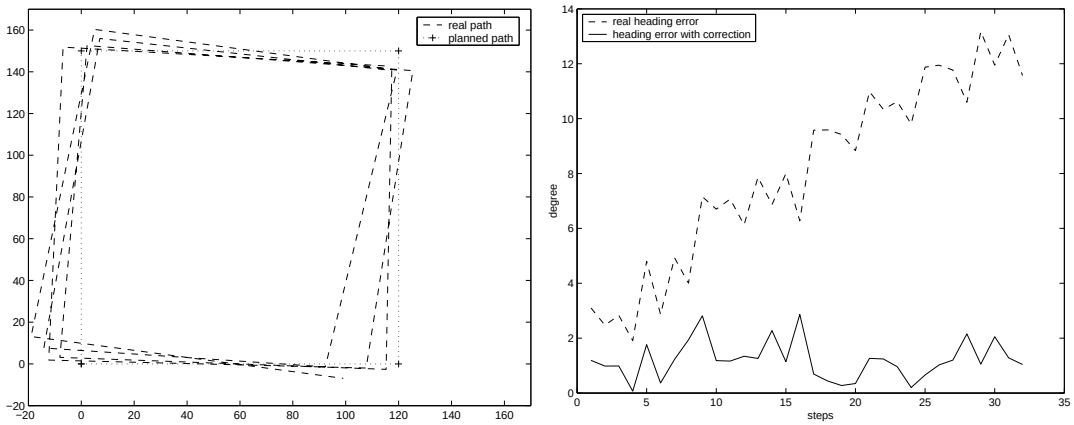



Fig. B.15: Left: Planned path overlaid with the really driven path. Right: Odometric (angle) error vs. visually corrected (angle) error

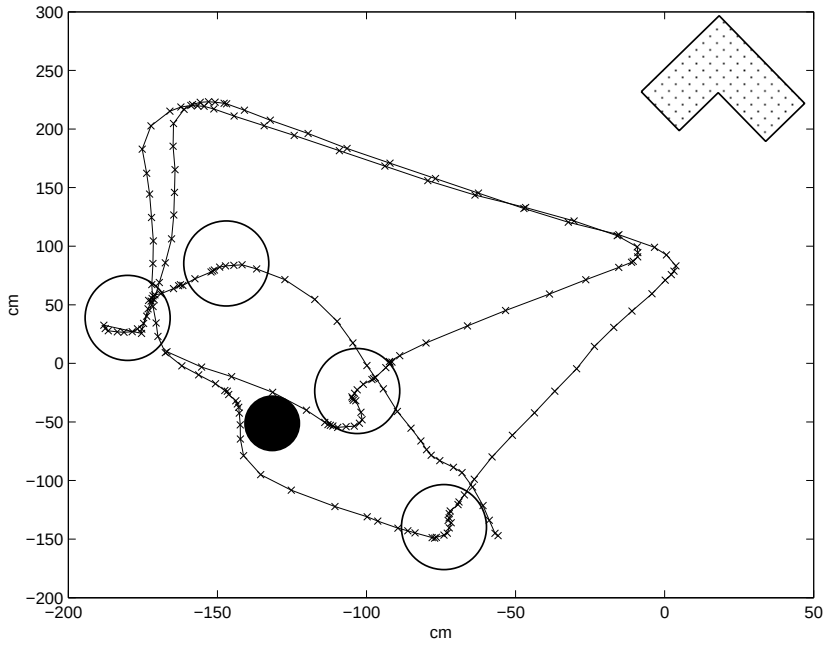



Fig. B.16: Navigation path: The white circles show the localization during the navigation. The black circle indicates an obstacle, which is avoided.

Figure B.16 shows the path during another navigation sequence. The white circles in figure B.16 indicate the positions of self-localization: The robot interrupts the path and turns toward the landmark. Then the robot continues with the movements. The black circle in figure B.16 indicates an

Fig. B.17: Hallway navigation example: The left column shows images with extracted Hough lines. The middle column shows the projected object model before matching and the right column shows the projected transformed object model after matching.

obstacle. The collision server overrides the movement commands and drives around the obstacle.

It is not only possible to use compact landmarks for self localization of the robot. In figure B.17 a hallway is used as object model. This means, it is also possible to apply the algorithm on CAD maps of office environments.

These experiments show the performance of the navigation system, mainly implemented by D. Grest. More details about the system can be found in his diploma thesis [68].

# BIBLIOGRAPHY


[1] Ackermann M. Akkumulieren von Objektrepräsentationen im Wahrnehmungs–Handlungs Zyklus. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2000.

[2] Andersson B. D. O. and Moore J.B. Optimal Filtering. *Prentice Hall, Englewood Cliffs, N. J.*, 1979.

[3] Araujo H., Carceroni R.L. and Brown C.M. A Fully Projective Formulation for Lowe's Tracking Algorithm. *Technical Report 641*, University of Rochester, 1996.

[4] Arbter K. Affine-invariant fourier descriptors. In *From Pixels to Features. Simon J.C. (Editor),* Elsevier Science Publishers, pp. 153-164, 1989.

[5] Arbter K. Affininvariante Fourierdeskriptoren ebener Kurven. *Technische Universität Hamburg-Harburg*, Ph.D. Thesis, 1990.

[6] Arbter K., Snyder W.E., Burkhardt H. and Hirzinger G. Application of affine-invariant fourier descriptors to recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 12, No. 7, pp. 640-647, 1990.

[7] Arbter K. and Burkhardt H. Ein Fourier-Verfahren zur Bestimmung von Merkmalen und Schätzung der Lageparameter ebener Raumkurven. *Informationstechnik*, Vol. 33, No. 1, pp.19-26, 1991.

[8] Arkin R.C. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, Vol. 8, No. 4, pp. 92–112, 1989.

[9] Arkin R.C. *Behaviour-Based Robotics*. MIT Press, Cambridge, Massachusetts, 1998.

[10] Ayache N. and Faugeras O. Building, registrating and fusing noisy visual maps. *International Journal of Robotics Research*, Vol. 7, No. 6, pp. 45-65, 1988.

[11] Bar-Itzhack I.Y. and Oshman Y. Attitude determination from vector observations: quaternion estimation. *IEEE Trans. Aerospace and Electronic Systems*, AES-21, pp. 128-136, 1985.

[12] Barron J.L. and Fleet D.J. and Beauchemin S.S. and Burkitt, T.A. Performance of optical flow techniques. *International Journal of Computer Vision (IJCV)*, Vol. 12, No. 1, pp. 43–77, 1994.

[13] Bayro-Corrochano E. The geometry and algebra of kinematics. In [164], pp. 457-472, 2001.

[14] Bayro-Corrochano E., Daniilidis K. and Sommer G. Motor algebra for 3D kinematics: The case of the hand-eye calibration. *Journal of Mathematical Imaging and Vision (JMIV)*, Vol. 13, pp. 79-100, 2000.

[15] Bayro-Corrochano E. and Kähler D. Kinematics of robot manipulators in the motor algebra In [164], pp. 473-490, 2001.

[16] Bayro-Corrochano E. and Rosenhahn B. A geometric approach for the analysis and computation of the intrinsic camera parameters. *Pattern Recognition*, No. 35, pp. 169-186, 2002.

[17] Bayro-Corrochanno E., Daniilidis K. and Sommer G. Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis*, Vol. 1, pp. 397-404 Lappeenranta, 1997.

[18] Beveridge J.R. Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose. *Ph.D. Thesis, Computer Science Department, University of Massachusetts, Amherst*, 1993. Available as Technical Report CS 93–5.

[19] Besl P.J. The free-form surface matching problem. *Machine Vision for Three-Dimensional Scenes, Freemann H. (Editor)*, pp. 25-71, Academic Press, San Diego, 1990.

[20] Beutelspacher A. Lineare Algebra. Eine Einführung in die Wissenschaft der Vektoren, Abbildungen und Matrizen. *Vieweg Lehrbuch Mathematik*, 1994.

[21] Blaschke W. Kinematik und Quaternionen, Mathematische Monographien 4. *Deutscher Verlag der Wissenschaften*, 1960.

[22] Bouget J.-Y. and Perona P. Visual navigation using a single camera *Proc. 5th Int. Conference on Computer Vision (ICCV)*, Boston MA, IEEE Computer Society Press, pp. 645-652, 1995.

[23] Bouma T.A. From unoriented subspaces to blade operators In *Applied Geometric Algebras for Computer Science and Engineering (AGACSE), Dorst L., Doran C. and Lasenby J. (Editors)*, Birkhäuser Verlag, pp. 59-68, 2001.

[24] Brand L. Vector and Tensor Analysis. *John Wiley and Sons*, 1947.

[25] Bregler C. and Malik J. Tracking people with twists and exponential maps. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Santa Barbara, California, pp. 8-15, 1998.

[26] Brooks R.A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23, March 1986.

[27] Brooks R.A. Intelligence without reason. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, pp. 569–595, 1991.

[28] Brooks R.A. From earwigs to humans. *Robotics and Autonomous Systems*, Vol. 20, No. 2-4, 1997.

[29] Bronstein I.N., Semendjajew K.A., Musiol G. and Mühlig H. Taschenbuch der Mathematik. *Verlag Harri Deutsch*, Frankfurt a.M., 1995.

[30] Bunten A. Ein autonomes Robotersystem zum Folgen einer Person. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

[31] Burgard W., Cremers A.B., Fox D., Hähnel D., Lakemayer G., Schulz D., Steiner W. and Thrun S. Experiences with an Interactive Tour-guide Robot. *Technical Report CMU-CS-98-139, Carnegie Mellon University, Computer Science Department*, Pittsburgh, PA, 1998.

[32] Busemann H. and Kelly P.J. Projective geometry and projektive matrices. *Academic Press inc. Publishers New York*, 1953.

[33] Campbell R.J. and Flynn P.J. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding (CVIU)*, No. 81, pp.166-210, 2001.

[34] Carceroni R. L. and C. M. Brown. Numerical Methods for Model-Based Pose Recovery. *Technical Report 659, Computer Science Department, The University of Rochester, Rochester, N. Y.*, August 1998.

[35] Chiuso A. and G. Picci. Visual tracking of points as estimation on the unit sphere. In *The Confluence of Vision and Control*, pp. 90-105, Springer-Verlag, 1998.

[36] CLU Library, A C++ Library for Clifford Algebra. Available at http://www.perwass.de/CLU *Lehrstuhl für Kognitive Systeme, University Kiel,* 2001.

[37] Christensen H. and Kirkeby N. Model-driven vision for in-door navigation. *Robotics and Autonomous Systems*, Vol. 12, pp. 199–207, 1994.

[38] O'Connor J.J. and Robertson E.F. Famous Curves Index. Available at *http://www-history.mcs.st-andrews.ac.uk/history/Curves/Curves.html*.

[39] Curry D. *UNIX Systems Programming*. O 'Reilly Associates, 1996.

[40] Czopf A., Brack C., Roth M. and Schweikard A. 3D-2D registration of curved objects. *Periodica Polytechnica*, Vol. 43, No. 1, pp. 19-41, 1999.

[41] Daniilidis K. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, Vol. 18, pp. 286–298, 1999.

[42] Denavit J. and Hartenberg R.S. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, Vol. 22, pp. 215-221, 1955.

[43] Devernay F. and Faugeras O. From projective to Euclidean reconstruction. In *Conference on Computer Vision and Pattern Recognition 96 (CVPR)*, pp. 264-269, 1996.

[44] Dorst L. The inner products of geometric algebra. In *Applied Geometric Algebras for Computer Science and Engineering (AGACSE), Dorst L., Doran C. and Lasenby J. (Editors)*, Birkhäuser Verlag, pp. 35-46, 2001.

[45] Dorst L. Honing geometric algebra for its use in the computer sciences. In [164], pp. 127-152, 2001.

[46] Drummond T. and Cipolla R. Real-time tracking of multiple articulated structures in multiple views. In *6th European Conference on Computer Vision, ECCV 2000, Dubline, Ireland*, Part II, pp. 20-36, 2000.

[47] Faugeras O. Three-Dimensional Computer Vision, A Geometric Viewpoint. *MIT Press, Cambridge*, 1993.

[48] Faugeras O. Stratification of three-dimensional vision: projective, affine and metric representations *Journal of Optical Society of America*, Vol. 12, No. 3, pp. 465-484, March 1995.

[49] Faugeras O.D. What can be seen in three dimensions with an uncalibrated stereo rig? *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pp. 563-578, 1992.

[50] Faugeras O.D. and Maybank S. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision (IJCV)*, Vol. 4, pp. 225–246, 1990.

[51] Faugeras O. and Robert L. What can two images tell us about a third one? *International Journal of Computer Vision (IJCV)*, Vol. 18, pp. 5–19, 1996.

[52] Farouki R.T. Curves from motion, motion from curves. In *Curve and Surface Design, P.-J. Laurent, P. Sablonniere, and L.L. Schumaker (Editors)*, Vanderbilt University Press, Nashville, TN, pp.63-90, 2000.

[53] Farouki R.T. and Moon H.P. Bipolar und multipolar coordinates *The Mathematics of Surfaces IX. Proceedings of the ninth IMA conference on the mathematics of surfaces. Cipolla R. and Martin R. (Editors)*, pp. 348-371, Springer London, 2000.

[54] Felsberg M. Low-Level Image Processing with the Structure Multivector. *Technical Report 0203,Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2002.

[55] Felsberg M. and Sommer G. The multidimensional isotropic generalization of quadrature filters in Geometric Algebra, *2nd International Workshop on Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Sommer G. and Zeevi Y.Y. (Editors)*, Kiel, LNCS Vol. 1888, pp. 175-185, Springer-Verlag, Heidelberg, 2000.

[56] Felsberg M. and Sommer G. Structure multivector for local analysis of images. *In Multi-Image Search and Analysis, R.Klette, Th. Huang, G.Gimmel'farb (Editors)*, LNCS Vol. 2032 , Springer, pp. 93-104, 2001.

[57] Fenske A. Affin-invariante Erkennung von Grauwertmustern mit Fourierdeskriptoren. *Mustererkennung 1993*, pp. 75-83, Springer-Verlag, 1993.

[58] Foley J. and van Dam A. and Feiner S. and Hughes J. *Computer Graphics Principles and Practice, 2nd ed. in C.* Addison-Wesley Publishing Company, Reading, Mass., 1996.

[59] Fox D., Burgard W. and Thrun, S. The Dynamic Window Approach to Collision Avoidance. *Technical Report*, IAI-TR-95-13, University of Bonn, 1995.

[60] Franz M.O. and Mallot H.A. Biomimetic Robot Navigation. *Technical Report No. 65, Max-Planck-Institut für biologische Kybernetik*, 1998.

[61] Funda J. and Paul R.P. A computational analysis of screw transformations in robotics. *IEEE Trans. Robotics and Automation*, Vol. 6, pp. 348–356, 1990.

[62] Gallier J. Geometric Methods and Applications for Computer Science and Engineering. *Springer-Verlag*, New York Inc. 2001.

[63] Ghosh B.K., Jankovic M and Wu Y.T. Perspective problems in system theory and its application to machine vision. *Journal of Mathematical Systems, Estimation and Control*, Birkhäuser-Boston, Vol. 4, No. 1, pp. 3–38, 1994.

[64] Gilbert J.E. and Murray M.A.M. Clifford Algebras and Dirac Operators in Harmonic Analysis. *Cambridge University Press*, 1991.

[65] Goddard J.S. Pose and Motion Estimation From Vision Using Dual Quaternion-Based Extended Kalman Filtering. *University of Tennessee, Knoxville*, Ph.D. Thesis, 1997.

[66] Granert O. Poseschätzung Kinematischer Ketten. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

[67] Granlund G. Fourier preprocessing for hand print character recognition. *IEEE Transactions on Computers*, Vol. 21, pp. 195-201, 1972.

[68] Grest D. Ein System zur visuellen landmarkenbasierten Roboternavigation. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

[69] Grimson W. E. L. Object Recognition by Computer. *The MIT Press, Cambridge, MA*, 1990.

[70] Hailu G. and Sommer G. Embedding knowledge in reinforcement learning. *International Conference on Artificial Neural Networks (ICANN)*, pp. 1133-1138, Skovde, Sweden, 1998.

[71] Haralick R.M. and Joo H. 2d-3d pose estimation. *Proceedings of International Conference on Pattern Recognition (ICPR)*, pp. 385-391, 1988.

[72] Hartley R. Triangulation. *Computer Vision and Image Understanding (CVIU),* Vol. 68, pp. 146-157, 1997.

[73] Hartley R. and Pipitone F. Experiments with the subsumption architecture. *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento California, pp. 1652-1658, 1991.

[74] Hauck A. Lanser S and Zierl C. Hierarchical recognition of articulated objects from single perspective views. *IEEE Computer Vision and Pattern Recognition (CVPR)*, Puerto Rico, pp. 870-876, 1997.

[75] Hel-Or Y. and Werman M. Pose estimation by fusing noisy data of different dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 17, No. 2, February 1995.

[76] Hel-Or Y. and Werman M. Constraint fusion for recognition and localization of articulated objects. *International Journal of Computer Vision (IJCV)*, Vol. 19, No. 1, pp. 5–28, 1996.

[77] Hestenes D. The design of linear algebra and geometry. *Acta Applicandae Mathematicae*, Vol. 23, pp. 65–93, 1991.

[78] Hestenes D. Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, Vol. 7, pp.65–77, 1994.

[79] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [164], pp. 3-23, 2001.

[80] Hestenes D. and Sobczyk G. Clifford Algebra to Geometric Calculus. *D. Reidel Publ. Comp., Dordrecht*, 1984.

[81] Hestenes D. and Ziegler R. Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, Vol. 23, pp. 25–63, 1991.

[82] Hilbert and Cohn-Vossen Anschauliche Geometrie. *Springer-Verlag Berlin, Heidelberg*, 2. Auflage 1996.

[83] Holt J.R. and Netravali A.N. Camera calibration problem: Some new results. *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 54, No. 3, pp. 368–383, 1991.

[84] Holt J.R. and Netravali A.N. Uniqueness of solutions to structure and motion from combinations of point and line correspondences. *Journal of Visual Communication and Image Representation*, Vol. 7, No. 2 pp. 126–136, 1996.

[85] Homer H. C. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 13, No. 6, pp. 530–541, 1991.

[86] Hoppen P. Autonome mobile Roboter. *B.I. Wissenschaftsverlag*, Mannheim 1992.

[87] Horaud R., Phong T.Q. and Tao P.D. Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision (IJCV)*, Vol. 15, pp. 225–243, 1995.

[88] Hough P.V.C. Methods and means for recognizing complex patterns. *U.S. Patent 3,069,654, Dec. 18*, 1962.

[89] Huber D.F. and Hebert M. Fully automatic registration of multiple 3D data sets. *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum (CVBVS 2001)*, 2001.

[90] Hwang Y.K. and Ahuja N. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, pp. 23-32, February 1992.

[91] Jähne B. Digitale Bildverarbeitung *Springer-Verlag*, Berling, Heidelberg, New-York, 1997.

[92] Kanatani K. and Ohta N. Optimal robot self-localisation and reliability evaluation. *5th European Conference on Computer Vision*, LNCS Vol. 1407, Springer-Verlag, Heidelberg, pp. 796-808, 1998.

[93] Kauppinen H., Seppänen T. and Pietikäinen M. An experimental comparison of autoregressive and fourier-based descriptors in 2D shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 17, No. 2, pp. 201-207, 1995.

[94] Kernighan B.W. and Ritchie D.M. *The C Programming Language, Second Edition*. Prentice Hall, Inc., 1988.

[95] KiViGraP (Homepage of the Kieler Vision and Grasping Project). http://www.ks.informatik.uni-kiel.de/~kivi/kivi.html.

[96] Klingen B. Fouriertransformation für Ingenieur- und Naturwissenschaften. *Springer-Verlag*, Berlin, Heidelberg, New York, 2001.

[97] Klingspohr H., Block T., Grigat R.-R. A passive real-time gaze estimation system for human-machine interfaces. In: *Computer Analysis of Images and Patterns (CAIP), Sommer G., Daniilidis K., Pauli J. (Editors)*, LNCS Vol. 1296, Springer-Verlag Heidelberg, pp. 718-725, 1997.

[98] Kosaka A. and Kak A.C. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 56, No. 3, pp. 271-329, November, 1992.

[99] Kriegman D.J., Vijayakumar B., and Ponce, J. Constraints for recognizing and locating curved 3D objects from monocular image features. In: *Proceedings of Computer Vision (ECCV '92), G. Sandini, (Editor)*, LNCS 588, Springer-Verlag Heidelberg, pp. 829–833, 1992.

[100] Krüger N., Ackermann M. and Sommer G. Accumulation of object representations utilizing interaction of robot action and perception. *Mustererkennung 2000, Sommer G., Krüger N. and Perwass Ch. (Editors)*, Informatik aktuell, Springer-Verlag Heidelberg, pp. 365-372, 2000.

[101] Krüger N., Wendorff D. and Sommer G. Two models of a vision–based robotic system: Visual haptic attention and accumulation of object representations. In: *International Workshop on Robot Vision; Auckland, New Zealand, Klette R., Sommer G., Peleg S. (Editors)*, LNCS Vol. 1998, Springer-Verlag, Heidelberg, pp. 219-226, 2001.

[102] Krüger N., Jäger T. and Perwass Ch. Extraction of object representations from stereo image sequences utilizing statistical and deterministic regularities in visual data. In: *Proceedings of the 2nd Workshop on Biologically Motivated Computer Vision*, 2002.

[103] Krüger N. and Wörgötter F. Multi modal estimation of collinearity and parallelism in natural image sequences. To appear: *Network: Computation in Neural Systems*, 2003.

[104] Kunze S. Ein Hand-Auge-System zur visuell basierten Lokalisierung und Identifikation von Objekten. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2000.

[105] Lanser S. and Zierl Ch. On the use of topological constraints within object recognition tasks. *In Proceedings of 13th International Conference on Pattern Recognition (ICPR)*, Vol. 1, pp. 580–584, 1996.

[106] Lanser S., Zierl C., Munkelt O., Radig B. MORAL-a vision-based object recognition system for autonomous mobile systems. *Computer Analysis of Images and Patterns (CAIP), Sommer G., Daniilidis K., Pauli J. (Editors)*, LNCS Vol. 1296, Springer-Verlag, Heidelberg, pp. 33-41, 1997.

[107] Lasenby A. and Lasenby J. Surface evolution and representation using geometric algebra. *The Mathematics of Surfaces IX. Proceedings of the ninth IMA conference on the mathematics of surfaces. Cipolla R. and Martin R. (Editors)*, pp. 144-168, Springer London, 2000.

[108] Lee X. A Visual Dictionary of Special Plane Curves. *http://xahlee.org/SpecialPlaneCurves_dir/specialPlaneCurves.html*

[109] Levitt T.S. and Lawton D.T. Qualitative navigation for mobile robots. *Journal of Artificial Intelligence*, No. 44, pp. 305-360, 1990.

[110] Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [164], pp. 27-52, 2001.

[111] Li H., Hestenes D. and Rockwood A. A universal model for conformal geometries In [164], pp. 77-104, 2001.

[112] Li H. and Sommer G. Coordinate-free projective geometry for computer vision. In [164], pp. 415-454, 2001.

[113] Lin C.-S. and Hwang C.-L. New forms of shape invariants from elliptic fourier descriptors. *Pattern Recognition*, Vol. 20, No. 5, pp. 535-545, 1987.

[114] Lorenz F. Lineare Algebra I+II *BI Wissenschaftsverlag* 3. Auflage, 1992.

[115] Lorusso A., Eggert D.W. and Fisher R.B. A comparison of four algorithms for estimating 3D rigid transformations. *Machine Vision and Applications*, Vol. 9, No. 5/6, pp. 272-290, 1997.

[116] Lowe D.G. Solving for the parameters of object models from image descriptions. in *Proc. ARPA Image Understanding Workshop*, pp. 121-127, 1980.

[117] Lowe D.G. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, Vol. 31, No. 3, pp. 355-395, 1987.

[118] Mallot H.A. Sehen und die Verarbeitung visueller Informationen. *Vieweg Verlag*, 1998.

[119] Mallot H.A., Bültoff H.H., Little J.J. and Bohrer S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics* Vol. 64, pp. 177-185, 1994.

[120] Marsland S. and Nehmzow U. Selecting Landmarks from Sonar Scans for Mobile Robot Navigation. *Technical Report Series, Report UMCS-00-8-1.* University of Manchester, 2000.

[121] Maybank S. Theory of reconstruction from image motion. *Springer-Verlag*, pp. 173-215, 1992.

[122] McCarthy J.M. Introduction to Theoretical Kinematics. *MIT Press, Cambridge, Massachusetts, London, England*, 1990.

[123] Mundy J.L. and Zisserman A. Geometric Invariants in Computer Vision. *MIT Press, Cambridge, Massachusetts, USA*, 1992.

[124] Murray R.M., Li Z. and Sastry S.S. A Mathematical Introduction to Robotic Manipulation. *CRC Press*, 1994.

[125] Navab N. and Faugeras O. Monocular pose determination from lines: Critical sets and maximum number of solutions. *Proceedings Computer Vision and Pattern Recognition (CVPR),* New York, pp. 254-260, 1993.

[126] Needham T. Visual Complex Analysis. *Oxford University Press*, 1997

[127] Pauli J. Geometric/photometric consensus and regular shape quasi-invariants for object localization and boundary extraction. *Technical Report 9805, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 1998.

[128] Pauli J. Development of Camera-Equipped Robot Systems. *Technical Report 9904, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2000.

[129] Pauli J. Learning-Based Robot Vision, Principles and Applications. *Springer-Verlag*, 2001.

[130] Pennec X. and N. Ayache. Uniform distribution, distance and expectation problems for geometric features processing. *Journal of Mathematical Imaging and Vision*, Vol. 9, pp. 49-67, 1998.

[131] Perwass C. Applications of Geometric Algebra in Computer Vision. *Cambridge University*, Ph.D. Thesis, 2000. Available at *http://www.perwass.de*

[132] Perwass C. and Hildenbrand D. Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space. An Introductory Tutorial. *To be published* 2003.

[133] Perwass C. and Lasenby J. A novel axiomatic derivation of geometric algebra. Technical Report CUED/F - INFENG/TR.347, Cambridge University Engineering Department, 1999.

[134] Perwass C. and Lasenby L. A unified description of multiple view geometry. In [164], pp. 337-369, 2001.

[135] Porteous I. R. Clifford Algebra and Classical Groups. *Cambridge University Press, Cambridge*, 1995.

[136] Press W.H., Teukolsky S.A., Vetterling W.T., and Flannery B.P. Numerical Recipes in C. *Cambridge University Press*, 1994.

[137] Princen J., Illingworth J. and Kittler J. An optimizing line finder using a Hough transform algorithm. *Computer Vision, Graphics, and Image Processing (CVGIP)*, Vol. 52, pp. 57-77, 1990.

[138] Reiss T.H. Recognizing Planar Objects Using Invariant Image Features. LNCS Vol. 676, *Springer-Verlag*, 1993.

[139] Rabsch T. Tracking von Objekten mittels Posenschätzung und lokaler Suche. *Studienarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2000.

[140] Riseman E. and Hanson A. and Beveridge J. and Kumar R. and Sawhney H. Landmark-based navigation and the acquisition of environmental models. *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles*, Yiannis Aloimonos (Editor), Lawrence Erlbaum Associates, Inc., pp. 317–374, 1997.

[141] Rooney J. A comparison of representations of general screw displacement. *Environment and Planning* , Vol. B5, pp. 45-88, 1978.

[142] Rooney J. On the three types of complex number and plane transformations. *Environment and Planning*, Vol. B5, pp. 89-99, 1978.

[143] Rosenhahn B., Granert O., Sommer G. Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering (AGACSE), Dorst L., Doran C. and Lasenby J. (Editors)*, Birkhäuser Verlag, pp. 373-383, 2001.

[144] Rosenhahn B., Krüger N., Rabsch T. and Sommer G. Tracking with a novel pose estimation algorithm. *International Workshop on Robot Vision, Klette R., Peleg S. and Sommer G. (Editors),* Springer-Verlag Heidelberg, LNCS Vol. 1998, pp. 9-19, 2001.

[145] Rosenhahn B. and Lasenby J. Constraint Equations for 2D-3D Pose Estimation in Conformal Geometric Algebra. Technical Report CUED/F - INFENG/TR.396, Cambridge University Engineering Department, 2000.

[146] Rosenhahn B., Perwass Ch. and Sommer G. Pose Estimation of 3D Free-form Contours. *Technical Report 0207, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2002.

[147] Rosenhahn B., Perwass Ch. and Sommer G. Free-form pose estimation by using twist representations. *Algorithmica (special issue)*, 2003, in print.

[148] Rosenhahn B., Perwass Ch. and Sommer G. Pose estimation of 3D free-form contours in conformal geometry. In *Proceedings of Image and Vision Computing (IVCNZ), D. Kenwright (Editor)*, Wickliffe Ltd, Dunedin, New Zealand, pp. 29-34, 2002.

[149] Rosenhahn B., Perwass C. and Sommer G. Modeling adaptive deformations during free-form pose. estimation. In *Proceedings of Computer Analysis of Images and Patterns (CAIP 2003), N. Petkov and M.A.*

*Westenberg (Eds.)*, Springer-Verlag, Berling Heidelberg, LNCS 2756, pp. 664-672, 2003.

[150] Rosenhahn B. and Sommer G. Pose Estimation in Conformal Geometric Algebra. Part I: The stratification of mathematical spaces. Part II: Real-Time pose estimation using extended feature concepts. *Technical Report 0206*, University Kiel, 2002. *Accepted: Journal of Mathematical Imaging and Vision (to appear 2004).*

[151] Rosenhahn B. and Sommer G. Adaptive pose estimation for different corresponding entities. In *Pattern Recognition, 24th DAGM Symposium, L. Van Gool (Editor)*, Springer-Verlag, Berling Heidelberg, LNCS Vol. 2449, pp. 265-273, 2002.

[152] B. Rosenhahn and G. Sommer. Pose estimation of cycloidal curves by using twist representations. To appear *CLIFFORD ALGEBRAS: Application to Mathematics, Physics, and Engineering*, R. Ablamowicz, (Editor), Progress in Mathematical Physics, Birkhäuser Verlag, Boston, 2003.

[153] Rosenhahn B., Zhang Y., and Sommer G.. Performance of constraint based pose estimation algorithms. *Mustererkennung 2000, Sommer G., Krüger N. and Perwass Ch. (Editors)*, Informatik aktuell, Springer-Verlag, Heidelberg, pp. 277-284, 2000.

[154] Rosenhahn B., Zhang Y. and Sommer G. Pose estimation in the language of kinematics. In: *Second International Workshop, Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Sommer G. and Zeevi Y.Y. (Editors)*, LNCS Vol. 1888, Springer-Verlag, Heidelberg, pp. 284-293, 2000.

[155] Rosenhahn C., Hayes S., Rosenhahn B., Eckert H. Structural organization of Arsenic Selenide liquids: New results from liquid state NMR. In *Journal of Non Crystaline Solids*, Vol. 284, pp. 1-8, 2001.

[156] Ruf A. Closing the loop between articulated motion and stereo vision: a projective approach *These pour obtenir le grade de DOCTEUR de I'INPG, Institut National Polytechnique (INP)*, Grenoble 2001.

[157] Ruf A. and Horaud R. Vision-based guidance and control of robots in projective space. *Proceedings, 6th European Conference on Computer Vision (ECCV), Part II, Vernon D. (Editor)*, LNCS Vol. 1843, Springer-Verlag Heidelberg, pp. 50-66, 2000.

[158] Rusinkiewicz S. and Levoy M. Efficient variants of the ICP algorithm. Available at http://www.cs.princeton.edu/ smr/papers/fasticp/. Presented at *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[159] RWI, Homepage of Real World Interface. http://www.irobot.com/rwi/index.asp

[160] Schmitz O. and Koberstein J. Fernsteuerung eines Roboterarmes durch Gesten. *Fortgeschrittenenpraktikum, WS01/02 and SS02, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2002.

[161] Selig J.M. Some Remarks on the Statistics of Pose Estimation. *Technical Report SBU-CISM-00-25*, South Bank University London, 2000.

[162] IEEE Robotics and Automation Society http://www.service-robots.org/IEEE-start.php

[163] Shevlin F. Analysis of orientation problems using Plücker lines. *International Conference on Pattern Recognition (ICPR)*, Brisbane, Vol. 1, pp.685–689, 1998.

[164] Sommer G., editor. Geometric Computing with Clifford Algebra. *Springer-Verlag*, Heidelberg, 2001.

[165] Sommer G. Algebraic aspects of designing behavior based systems. *Algebraic Frames for the Perception-Action Cycle, AFPAC'97, Sommer G. and Koenderink J.J. (Editors)*, Springer-Verlag Heidelberg, LNCS Vol. 1315, pp. 1–28, 1997.

[166] Sommer G. The global algebraic frame of the perception-action cycle. *Handbook of Computer Vision and Applications, Vol. 3*, Academic Press, San Diego, pp. 221–264, 1999.

[167] Sommer G., Rosenhahn B., and Zhang Y. Pose estimation using geometric constraints. *In Klette R., Huang Th. and Gimmel'farb G. (Editors), Multi-Image Search and Analysis*, LNCS Vol. 2032, Springer-Verlag, Heidelberg, pp. 153–170, 2001.

[168] Sommer G., Rosenhahn B. and Zhang Y. Pose estimation using geometric constraints. *Technical Report 2003, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2000.

[169] Stäubli Roboter RX90-CS7 Instruction Manual *D280.190.12.C, November*, 1992.

[170] Stark H. Zielgerichtete Navigation und Kollisionsvermeidung eines mobilen Roboters. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 1998.

[171] Stark K. A Method for Tracking the Pose of Known 3D Objects Based on an Active Contour Model. *Technical Report TUD / FI 96 10*, TU Dresden, 1996.

[172] Stoer J. Numerische Mathematik 1+2 *Springer Verlag*, Heidelberg, 7. Auflage, 1994.

[173] Tello R. Fourier descriptors for computer graphics. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 5, pp. 861-865, 1995.

[174] Thrun S. and A. Bucken W. and Burgard D. and Fox T. and Frohlinghaus D. and Henning and T. Hofmann and M. Krell and T. Schmidt Map learning and high-speed navigation in RHINO *AI-Based Mobile Robots: Case Studies of Successful Robot Systems*, Boston, Mass., MIT Press, 1998.

[175] Thrun S. and Fox D. and Burgard D. and Dellaert F. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence Journal*, Vol. 128, No. 1-2, pp. 99–141, 2001.

[176] Tsotsos J. Behaviorist intelligence and the scaling problem. *Artificial Intelligence*, Vol. 75, pp. 135-160, 1995.

[177] Ude A. Filtering in a unit quaternion space for model-based object tracking. *Robotics and Autonomous Systems*, Vol. 28, No. 2-3, pp. 163-172, August 1999.

[178] Ude A. Nonlinear least squares optimization of unit quaternion functions for pose estimation from corresponding features. *International Conference on Pattern Recognition, Brisbane*, Vol. 1, pp. 425-427, 1998.

[179] Walker M.W. and Shao L. Estimating 3-d location parameters using dual number quaternions. *Computer Vision, Graphics, and Image Processing (CVGIP): Image Understanding*, Vol. 54, No. 3, pp. 358–367, 1991.

[180] Wei Guo-Qing and Song De Ma Implicit and explicit camera calibration: Theory and experiments. *IEEE trans. on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 16, No. 5, pp. 469–480, May 1994.

[181] Weik S. and Liedtke C.-E. Hierarchical 3D pose estimation for articulated human body models. In: *International Workshop on Robot Vision, Klette R., Peleg S. and Sommer G. (Editors)*, Springer-Verlag Heidelberg, LNCS Vol. 1998, pp. 27-34, 2001.

[182] Weng J., Huang T. and Ahuja N. Motion and structure from two perspective views: Algorithms, error analysis and error estimation. *IEEE trans. on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 11, pp. 451-476, 1989.

[183] Winkler S., Wunsch P. and Hirzinger G. A feature map approach to real-time 3D object pose estimation from single 2D perspective views. *Mustererkennung 1997, Paulus E. and Wahl F.M. (Editors)*, Informatik aktuell, Springer-Verlag, Heidelberg, pp.129-136, 1997.

[184] Yaglom M. Felix Klein and Sophus Lie. *Birkhäuser*, Boston, 1988

[185] Yaglom M. Complex numbers in Geometry. *Academic Press, Leicester*, 1968.

[186] Zahn C.T. and Roskies R.Z. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, Vol. 21, No. 3, pp. 269-281, 1972.

[187] Zang Z. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision (IJCV)*, Vol. 13, No. 2, pp. 119-152, 1999.

[188] Zerroug, M. and Nevatia, R. Pose estimation of multi-part curved objects. *Image Understanding Workshop (IUW)* pp. 831-835, 1996.

[189] Zhang Y., Rosenhahn B. and Sommer G. Extended Kalman filter design for motion estimation. In: *Second International Workshop, Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Sommer G. and Zeevi Y.Y. (Editors)*, LNCS Vol. 1888, Springer-Verlag, Heidelberg, pp. 339-348, 2000.

[190] Zhang Y., Sommer G. and E. Bayro-Corrochano. The motor extended Kalman filter for dynamic rigid motion estimation from line observations. In [164], pp. 503-530, 2001.

[191] Zhang Z. Parameter estimation techniques: a tutorial with application to conic fitting. In *Image and Vision Computing*, Vol. 15, pp. 59-76, 1997.

[192] Zhang, Z. and O. Faugeras. 3D Dynamic Scene Analysis. *Springer Verlag*, 1992.

# INDEX


$O(n)$, 155
$SE(3)$, 19, 34, 50
$SE(n)$, 156
$SO(3)$, 33
$SO(n)$, 155
$se(3)$, 157
$so(3)$, 157
$\mathbb{H}$, 32
1twist curve, 95
2twist curve, 96
3twist curve, 96

affine space, 19, 153
affine transformation, 154
algebraic curves, 92
astroid, 93

B21, 176
basis, 152
behavior based robotics, 177
bipolar coordinates, 11
bracket, 28

calibration, 160
cardioid, 92, 97
Chasles, 53
circle, 45, 88
Conformal geometric algebra, 39
conformal transformation, 20
contour interpolation, 166
coupled twists, 91, 119
cross product rule, 30
cycloidal curves, 14, 91

deltoid, 93
Denavit-Hartenberg parameterization, 83
DFT, 164
dual, 28
dual shuffle product, 28

ellipse, 93
ellipse interpolation, 113
ellipsoid, 97
envelope, 99
epitrochoid, 92
Euclidean geometric algebra, 29
Euclidean space, 19, 153
Euler representation, 33
evolute, 100
extended object concepts, 83
extrinsic camera parameters, 160

field, 151
focal length, 159
Fourier descriptors, 11, 117, 120
Fourier transformation, 117

general rotation, 51
geometric algebra, 25
gradient descent, 14, 23
Grassmann, 25
group, 155

Hamilton, 25
Hamilton relation, 32
Hamming distance, 171
Hesseform of planes, 31, 36, 48
hierarchical pose estimation, 106
homogeneous, 32
Hough transformation, 174

Hough transformation, 15, 24
hypotrochoid, 92

ICP algorithm, 15, 124, 125
increasing degree method, 127
inner product, 27
intercept theorem, 40
interpolation, 163
intrinsic camera parameters, 160

join, 28

Kalman filter, 23
kinematic chain, 83, 106
kinematic space, 20
knowledge based robotics, 177

Lagrange identity, 31
Lie algebra, 156
Lie group, 156
line, 31, 36, 47
local search, 15, 170
low-pass approximation, 127
Lowe, 8

matching, 167, 173
meet, 28
Minkowski plane, 43
motor, 51
motor algebra, 44
multipolar coordinates, 11

navigation, 2
nephroid, 93
null cone, 44

outer product, 27

period, 162
periodic function, 162
phase, 162
phase vectors, 118
pinhole camera, 159
Plücker line, 31, 36, 47
plane, 31, 36, 47
point, 31, 35, 43
point pair, 48
pose, 4, 17
pose estimation, 4
prismatic joint, 83
projective geometric algebra, 35
projective space, 20, 154
projective split, 58, 61

quaternion, 32

ray-tracing, 99
reverse, 28
revolute joint, 83
rigid body motion, 50
rigid motion, 4, 19, 155
rose, 93
rotation, 32
rotor, 32, 50
RX-90, 104

screw, 53
screw motion, 53
sphere, 45, 88
spiral, 98
stereographic projection, 39
stratification hierarchy, 20
SVD, 23

tracking, 167, 171
translator, 50
trigonometric interpolation, 117, 163
trochoid, 94
twist, 51
twist surface, 96

vector space, 152