INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Pose Estimation in Conformal Geometric Algebra

Part I: The Stratification of Mathematical Spaces

Part II: Real-time Pose Estimation using Extended Feature Concepts

Bodo Rosenhahn, Gerald Sommer

Bericht Nr. 0206
November 2002

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

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

Pose Estimation in Conformal Geometric Algebra

Part I: The Stratification of Mathematical Spaces

Part II: Real-time Pose Estimation using Extended Feature Concepts

Bodo Rosenhahn, Gerald Sommer

Bericht Nr. 0206
November 2002

e-mail: {bro,gs}@ks.informatik.uni-kiel.de

Pose Estimation in Conformal Geometric Algebra

Part I: The Stratification of Mathematical Spaces

Part II: Real-time Pose Estimation using Extended Feature Concepts

Bodo Rosenhahn and Gerald Sommer

Cognitive Systems Group

Institute of Computer Science and Applied Mathematics

Christian-Albrechts-University of Kiel

D-24105 Kiel, Germany

{bro,gs}@ks.informatik.uni-kiel.de

Part I pp. 1 – 50

Part II pp. 51 – 96

Pose Estimation in Conformal Geometric Algebra

Part I: The Stratification of Mathematical Spaces

Bodo Rosenhahn and Gerald Sommer
Cognitive Systems Group
Institute of Computer Science and Applied Mathematics
Christian-Albrechts-University of Kiel
D-24105 Kiel, Germany
{bro,gs}@ks.informatik.uni-kiel.de

Abstract

2D-3D pose estimation means to estimate the relative position and orientation of a 3D object with respect to a reference camera system. This work has its main focus on the theoretical foundations of the 2D-3D pose estimation problem: We discuss the involved mathematical spaces and their interaction within higher order entities. To cope with the pose problem (how to compare 2D projective image features with 3D Euclidean object features), the principle we propose is to reconstruct image features (e.g. points or lines) to one dimensional higher entities (e.g. 3D projection rays or 3D reconstructed planes) and express constraints in the 3D space. It turns out that the stratification hierarchy [9] introduced by Faugeras is involved in the scenario. But since the stratification hierarchy by Faugeras is based on pure point concepts a new algebraic embedding is required when dealing with higher order entities. The conformal geometric algebra (CGA) [22] is well suited to solve this problem, since it subsumes the involved mathematical spaces. Operators are defined to switch entities between the algebras of the conformal space and its Euclidean and projective subspace. This leads to another interpretation of the stratification hierarchy, which is not restricted to be based solely on point concepts. This work summarizes the theoretical foundations needed to deal with the pose problem. Therefore it contains mainly basics of Euclidean, projective and conformal geometry. Since especially conformal geometry is not well known in computer science, we recapitulate the mathematical concepts in some detail. We believe that this geometric model is useful also for many other computer vision tasks and has been ignored so far. Applications of these foundations are presented in part II.

Keywords : 2D-3D pose estimation, stratification hierarchy, conformal geometric algebra.

CONTENTS

1. INTRODUCTION

In this work we are concerned with the theoretical foundations of an algorithmic approach for simultaneous 2D-3D pose estimation from correspondences of different entities. Pose estimation itself is a basic visual task [12], and several approaches for monocular pose estimation exist, which relate the position of a 3D object to a reference camera coordinate system [1, 37, 20, 33]. Nearly all papers concentrate on one specific type of correspondences. But many situations are conceivable in which a system has to gather information from different hints or has to consider different reliabilities of measurements. While from the first situation the necessity follows to relate the correspondences of quite different geometric entities, the second problem necessitates the use of weighted mixtures of correspondences. To cope algebraically with these combined informations, is in general very hard. For example, some algorithms assume point correspondences between 3D model and 2D image data and relate 3D points to 3D projection lines [33]. Other algorithms assume line correspondences and relate 3D lines to 3D (reconstructed) planes [19, 20]. Several algorithms use information of the image plane to relate points to entities like circles [21]. All these papers use different algebraic embeddings. Matrix, quaternion and dual-quaternion algebras can be found to describe the situations in different geometries (Euclidean, affine or projective) [9, 28, 32].

One work concerning the combination of different kinds of correspondences can be found in [18]. There only point and line correspondences are treated.

In [28, 35] we started to embed the pose estimation problem for point, line and plane correspondences in the kinematic framework. We continued in [29] by applying a conformal [22] embedding, which appears much more compact and natural. This enables us to formalize the monocular pose estimation problem for kinematic chains [30].

Our work is separated in two parts, part I (this article) and part II. Part I deals with the foundations of the pose estimation problem and formalize the pose scenario by using the language of geometric algebras. It turns out, that the conformal geometric algebra (CGA) provides a new model dealing with projective and kinematic geometry which is not based on point concepts

leading to a *new* stratification hierarchy. In Part II we then continue with application of these foundations to the pose estimation problem of different corresponding entities.

The main attribute of this contribution is to give an overview of the geometric scenario for 2D-3D pose estimation and their algebraic embedding in conformal geometric algebra (CGA) [22]. The contribution is organized as follows: The second section describes the pose estimation scenario in the context of the stratification hierarchy. Then, geometric algebras are introduced. Therefore, we start with the algebra of the Euclidean space, continue with the algebra of the projective space and end up in the algebra of the conformal space which subsumes the former ones. In the fourth section, the relations of projective and conformal geometry will be developed. This will be used in the part II to formalize the 2D-3D pose estimation problem in one algebraic context.

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

This section introduces the foundations of the 2D-3D pose estimation problem. Therefore, the general scenario is explained firstly. Then the involved mathematical spaces are explained and thirdly, the main principles 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 we describe the following situation: We assume

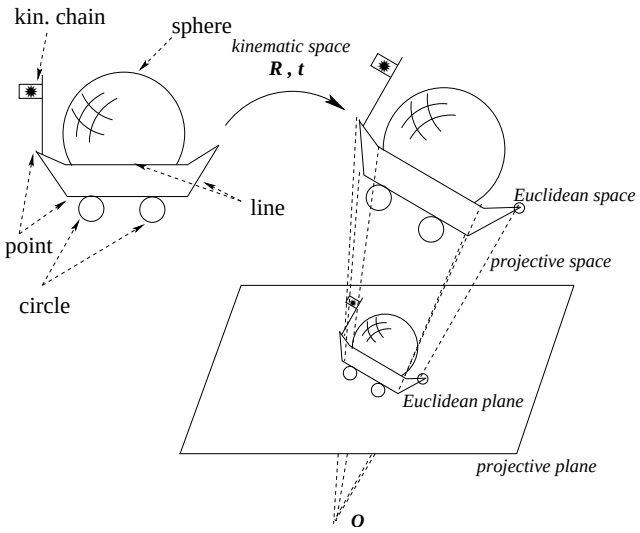

Fig. 2.1: The scenario. The solid lines describe the assumptions: the camera model, the model of the object (consisting of points, lines, circles, spheres and kinematic chains) and corresponding extracted entities on the image plane. The dashed lines describe the pose of the model, which leads to the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

3D points, 3D lines, 3D spheres, 3D circles or kinematic chain segments

as features or components of an object or reference model. Further, we extract corresponding features in an image of a calibrated camera. The aim is to find the rotation $\mathcal{R}$ and translation $\mathcal{T}$ of the object, which leads to the best fit of the reference model with the actual extracted entities. One main question is, how to define a geometric error measure with that respect. Though it is clear by intuition, a mathematic formalization is not easy and not unique. Comparing model features to image features leads to sets of constraint equations which have to be solved and model the involved geometry in an implicit manner.

The method how to establish the correspondences is out of the scope of this paper. The reader should consult e.g. [4] as an example to solve the matching problem in this context.

2.2 The Stratification Hierarchy and Pose Estimation

In the scenario of figure 2.1 four mathematical frameworks can be identified: The first one is the projective plane of a camera, embedded in the second framework, a 3D projective space. In this 3D projective space it is possible to project or reconstruct entities. The third one is the framework of kinematics, a special affine map (the map of the *direct affine isometries*) [10], which can be used to describe rigid body motions. A set of entities with the property that the distances between any two of them never varies is called a *rigid body*, and a transformation with the property of preserving distances during transformation is called a *rigid body motion*. A rigid body motion corresponds to the Euclidean transformation group $SE(3)$. Although being a transformation by itself, it subsumes rotation and translation. To distinguish between two rigid body motions, a distance measure on the manifold has be be defined [7, 36]. But this is no simple task in general. Instead, the distance of two geometric entities in Euclidean space can be used to derive a measure of motions. This necessitates as a fourth framework, the Euclidean space or Euclidean plane. The basic definition of these spaces are the following [10]: The Euclidean space is a vector space V with a symmetric positive definite bilinear form (which induces a Euclidean norm). The kinematic space is an affine space with the group of rigid motions as special affine transformation. The projective space is the set of $(V\backslash\{0\})/\sim$ of equivalence classes with

$$\forall u, v \in V\backslash\{0\} : u \sim v \Leftrightarrow \exists \lambda \in \mathbb{R} : v = \lambda u.$$

Mathematically, a projective space $\mathcal{P}(V)$ is a set of equivalence classes of vectors in V. The spirit of 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 $a = (u)_\sim$

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

Tab. 2.1: Stratification of mathematical spaces

a *point* (the entire equivalence class $(u)_\sim$ is collapsed into a single object, viewed as a point).

The idea is to end up later in the Euclidean space. On that way it is possible to cope geometrically with the problem of noisy data 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 only in the very last step in a distance measure of the Euclidean space. Before this step, we want to use the other spaces to represent partial problems in a suitable way. The above mentioned spaces of the pose estimation scenario are exactly the spaces of the stratification hierarchy which Faugeras introduced in 1995 [9]. The three main representations he is considering are the projective, affine and metric ones. All strata are involved in the 2D-3D pose estimation problem.

In our approach, we are using geometric algebras instead of vector calculus to represent and handle different mathematical spaces of geometric meaning. The maximum sized algebra over Euclidean space so far used by us is an algebra to handle conformal transformations [15]. A transformation is said to be conformal if it (locally) preserves shape. This means it preserves angles. The conformal geometric algebra (CGA) contains the algebras for projective and Euclidean geometry as subalgebras, thus leading to another formalization of the stratification hierarchy, we propose in this contribution. Table 2.1 shows the different stratification hierarchies. The stratification hierarchy proposed by Faugeras has its roots in the vector space concepts and assumes points as the represented basic geometric entities. All other geometric entities are derived as subspaces of point sets without having an own algebraic existence. 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. So this way to stratify the vision space is clearly motivated by the underlying point concepts of the vector spaces.

In geometric algebras instead, we do have besides point concepts so-called multivector concepts to model geometry. In the next section we will explain why it is necessary also to extend geometric algebras to homogeneous models. But this leads to a different stratification of the space since this stratification

is not based on pure point concepts any more. Instead, the new stratification concept contains algebras for the Euclidean, projective and conformal space.

2.3 Principles of Solving the Pose Estimation Problem

The 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 in 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 projectively reconstruct the object features from the image data and to compare the (by one dimension higher) entities with the 3D object features. Both approaches have advantages and disadvantages. Here we 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 the denominator. The equations are not linear any more and are not easy 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 bad conditioned equations. To avoid such problems, orthographic projections (see e.g. [6]) are used, but then the camera model is not perspective any more. Since the second strategy uses projective reconstructed data, this problem does not occur there. But the problem is that the distance measures in the 3D space is different to those in the image plane: Though the distance of two image points may be constant, the distance of two 3D projectively reconstructed points varies with the distance of the points to the optical center of the camera. This necessitates for degenerate situations[1] that the (from the image and object features generated) constraint equations must be adapted with respect to the projective depth. Table 2.2 summarizes the main principles of solving the pose problem in an implicit manner.

In our approach (similar to [37]) we projectively reconstruct the 3D data from image data and compare the one dimensional higher entities (their projective equivalence classes) with the 3D object features. There are three main arguments why we decided for the second strategy which is based on the stratification concepts above: Firstly, we want to describe the constraints

[1] 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 objects influence the equations to a lesser extend then the far object features.

strategy	linear	geometric distance measure	full perspective
2D 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

as simply as possible and want to gain real-time performance. For this, the projectively reconstructed data are easier to handle in the 3D kinematic space than the projected data in the 2D projective space. The second advantage of our approach is that the error measures are formalized in the 3D Euclidean space and are directly connected to a spatial distance measure. This is in contrast to other approaches, where the minimization of estimating errors of the rigid body motion has to be computed directly on the manifold of the geometric transformation [7, 36]. The third argument is that the depth dependence of the 3D constraints can be adapted in each situation. As will be later shown (see part II) our constraints can be scaled, and therefore transformed in depth-depending constraints comparable to the situation observed in the 2D image plane.

Since CGA can be used to formalize kinematics and since it contains the algebras for projective and Euclidean geometry as subalgebras, it is well suited to be used in this context. Therefore, the whole scenario is formalized in CGA: That are the entities, the kinematic chains, the transformations of the entities and the constraints for collinearity, coplanarity and tangentiality of involved entities.

3. INTRODUCTION TO GEOMETRIC ALGEBRAS

What we currently call *geometric algebra* [15] is tightly related to Clifford algebra. Both in fact represent families of algebras which depend on both the chosen vector spaces the algebras are derived from and the chosen kind of product defining the special algebra. A nice historic introduction of Clifford's contribution of inventing a geometric extension of the real number system to such which provides a complete algebraic representation of directed numbers can be found in [38].

Clifford (or geometric) algebras have the properties of dense symbolic representations of higher order entities and of linear operations acting on those, coupled with strong under-pinned mathematical concepts. It is nice that 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. This is depicted in figure 3.1. In geometric algebras there are strong relations between algebraic and geometric entities. Furthermore, both the object concepts and the operations acting on those are represented in one unique mathematical language.

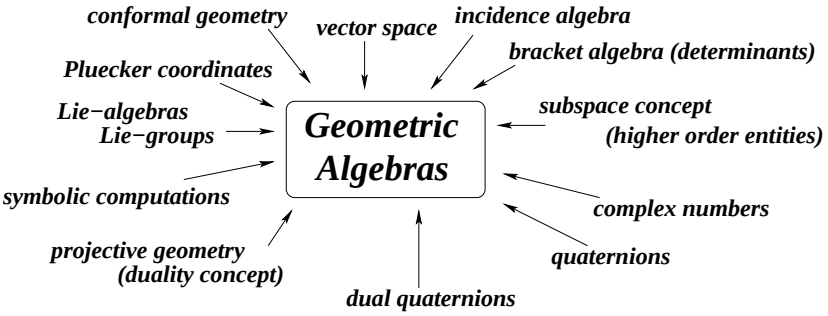

Fig. 3.1: Some mathematical concepts fused in geometric algebras.

We will now continue with a general introduction to geometric algebras

and will proceed with algebras to model the Euclidean, projective and conformal space. A more extended introduction into geometric algebras can be found in [14, 15, 13, 17, 34]. See also the courses on web, e.g. [8].

In general, a geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ is a linear space of 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, or scalars as grade zero entities. A geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ results in a constructive way from a vector space $\mathbb{R}^{p,q,r}$, endowed with the signature (p,q,r), by application of a geometric product. The geometric product of two multivectors $\boldsymbol{A}$ and $\boldsymbol{B}$ is denoted as $\boldsymbol{AB}$. The geometric product consists of an outer ($\wedge$) and an inner ($\cdot$) product, whose roles are to increase or to decrease the order of the algebraic entities, respectively. For further computations, we also use both the commutator $\underline{\times}$ and the anticommutator $\overline{\times}$ product for any two multivectors,

$$\boldsymbol{AB} = \frac{1}{2}(\boldsymbol{AB}+\boldsymbol{BA}) + \frac{1}{2}(\boldsymbol{AB}-\boldsymbol{BA}) =: \boldsymbol{A}\overline{\times}\boldsymbol{B} + \boldsymbol{A}\underline{\times}\boldsymbol{B}. \tag{3.1}$$

The reader should consult [26] 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.

To be more detailed, we define the geometric product of a geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ for two basis vectors $\mathbf{e}_i$ and $\mathbf{e}_j$ as

$$\mathbf{e}_i\mathbf{e}_j = \begin{cases} 1 & \text{for} \quad i=j \in \{1,\ldots,p\} \\ -1 & \text{for} \quad i=j \in \{p+1,\ldots,p+q\} \\ 0 & \text{for} \quad i=j \in \{p+q+1,\ldots,n\} \\ \mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i & \text{for} \quad i \neq j \end{cases} \tag{3.2}$$

A vector space with signature (p,q,r), $q \neq 0$, $r \neq 0$, is called pseudo-Euclidean. If $r \neq 0$, then its metric is degenerate. Although the dual-quaternions, which have some importance in kinematics, are isomorph to a degenerate geometric algebra, see [2, 3], we will in the following only consider non-degenerate geometric algebras $\boldsymbol{\mathcal{G}}_{p,q}$ where $r = 0$. Besides, we will write $\boldsymbol{\mathcal{G}}_n$ if $q = 0$, that is, there is a Euclidean metric.

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

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

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

Here $\alpha = \boldsymbol{u} \cdot \boldsymbol{v}$ represents a scalar, which is of grade zero, i.e. $\alpha \in \langle \boldsymbol{\mathcal{G}}_{p,q} \rangle_0$ with $\langle . \rangle_s$ is the operator to separate the grade-s entities of the linear space $\boldsymbol{\mathcal{G}}_{p,q}$. Besides $\boldsymbol{B} = \boldsymbol{u} \wedge \boldsymbol{v}$ represents a bivector, i.e. $\boldsymbol{B} \in \langle \boldsymbol{\mathcal{G}}_{p,q} \rangle_2$.

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

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

with

$$\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \sum_{i=1}^{s} (-1)^{i-1} \boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_{i-1} \wedge (\boldsymbol{u}_r \cdot \boldsymbol{v}_i) \wedge \boldsymbol{v}_{i+1} \wedge \ldots \wedge \boldsymbol{v}_s. \tag{3.6}$$

We will make this more explicit in the next subsections.

The geometric algebra $\boldsymbol{\mathcal{G}}_{p,q}$, with $p+q=n$ leads to an n-blade as element of maximum grade, the pseudo-scalar $\boldsymbol{P}$. The unit n-blade is called *unit pseudo-scalar* $\boldsymbol{I}$. Since every algebra derived from vector spaces of different signature contains its own unit pseudo-scalar, we later use indices to distinct between the different pseudo-scalars in the different algebras, e.g. we use $\boldsymbol{I}_E$, $\boldsymbol{I}_P$ or $\boldsymbol{I}_C$ for the unit pseudo-scalars of the algebra for the Euclidean, projective or conformal space, respectively.

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

$$[\boldsymbol{P}] \;=\; \boldsymbol{P}\boldsymbol{I}^{-1}. \tag{3.7}$$

For the bracket determined by n vectors, we write

$$\begin{aligned} [\boldsymbol{v}_1 \ldots \boldsymbol{v}_n] &= [\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n] \\ &= (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n)\boldsymbol{I}^{-1}. \end{aligned} \tag{3.8}$$

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

We define the *dual* $\boldsymbol{X}^\star$ of an r-blade $\boldsymbol{X}$ by

$$\boldsymbol{X}^\star \;=\; \boldsymbol{X}\boldsymbol{I}^{-1}. \tag{3.9}$$

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

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

$$(\boldsymbol{A} \vee \boldsymbol{B})^\star \;:=\; \boldsymbol{A}^\star \wedge \boldsymbol{B}^\star. \tag{3.10}$$

The shuffle product is the common factor of $\boldsymbol{A}$ and $\boldsymbol{B}$ with highest grade. The outer product of two blades $\boldsymbol{A}$ and $\boldsymbol{B}$ defines the *join*. The *join* $\boldsymbol{A} \dot{\wedge} \boldsymbol{B}$ is the pseudo-scalar of the space given by the sum of spaces spanned commonly by $\boldsymbol{A}$ and $\boldsymbol{B}$.

The shuffle product applied on the join of two blades can be used to estimate intersections of entities and will later be called the *meet* product.

Now we will proceed to introduce the algebras for the Euclidean, projective and conformal space.

3.1 The Euclidean Geometric Algebra

The algebra $\mathcal{G}_3$, which is derived from $\mathbb{R}^3$, i.e. $n = p = 3$, is the smallest and simplest one, we want to introduce here. This algebra is suitable to represent entities and operations in the 3D Euclidean space. Therefore, we call it EGA as abbreviation for Euclidean geometric algebra. We start with the three orthonormal basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of the 3D Euclidean space. The geometric algebra of the 3D Euclidean space consists of $2^3 = 8$ basis vectors:

$$\mathcal{G}_3 = span\{1\ , \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3,\ \ \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12},\ \ \mathbf{e}_{123} = \boldsymbol{I}_E\} \tag{3.11}$$

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 $\mathbf{e}_{123} = \mathbf{e}_1 \mathbf{e}_2 \mathbf{e}_3 = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 = \boldsymbol{I}_E$ is a trivector, called Euclidean unit pseudo-scalar, which squares to -1 and commutes with scalars, vectors and bivectors. To make more clear the above introduced rules of the geometric product, we will formulate the geometric product of two vectors as an example:

$$\begin{aligned}
\boldsymbol{u}\boldsymbol{v} &= (u_1\mathbf{e}_1 + u_2\mathbf{e}_2 + u_3\mathbf{e}_3)(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) \\
&= u_1\mathbf{e}_1(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) + u_2\mathbf{e}_2(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) + \\
&\quad + u_3\mathbf{e}_3(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + v_3\mathbf{e}_3) \\
&= u_1v_1 + u_2v_2 + u_3v_3 + (u_1v_2 - u_2v_1)\mathbf{e}_{12} + (u_3v_1 - u_1v_3)\mathbf{e}_{31} + \\
&\quad + (u_2v_3 - u_3v_2)\mathbf{e}_{23} \\
&= \boldsymbol{u} \cdot \boldsymbol{v} + \boldsymbol{u} \wedge \boldsymbol{v}
\end{aligned} \tag{3.12}$$

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 a bivector, representing the outer product of two vectors. The bivector corresponds to the dual of the vector which results from the cross product ($\times$) of two vectors (in matrix calculus). The inner product of a bivector ($\boldsymbol{a} \wedge \boldsymbol{b}$) with a vector $\boldsymbol{c}$ leads to another

vector,

$$\begin{aligned}(\boldsymbol{a}\wedge\boldsymbol{b})\cdot\boldsymbol{c} &\overset{3.6}{=} (\boldsymbol{a}\cdot\boldsymbol{c})\wedge\boldsymbol{b}-\boldsymbol{a}\wedge(\boldsymbol{b}\cdot\boldsymbol{c})\\ &= (\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b}-(\boldsymbol{b}\cdot\boldsymbol{c})\boldsymbol{a},\end{aligned} \tag{3.13}$$

and thus, we get the equivalent formulation of the famous cross product rule for the 3D case[1] ,

$$(\boldsymbol{a}\times\boldsymbol{b})\times\boldsymbol{c} = \langle\boldsymbol{a},\boldsymbol{c}\rangle\boldsymbol{b}-\langle\boldsymbol{b},\boldsymbol{c}\rangle\boldsymbol{a}. \tag{3.14}$$

The inner product of two bivectors leads to a scalar,

$$\begin{aligned}(\boldsymbol{a}\wedge\boldsymbol{b})\cdot(\boldsymbol{a}\wedge\boldsymbol{b}) &\overset{3.5}{=} ((\boldsymbol{a}\wedge\boldsymbol{b})\cdot\boldsymbol{c})\cdot\boldsymbol{d}\\ &\overset{3.13}{=} ((\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b}-(\boldsymbol{b}\cdot\boldsymbol{c})\boldsymbol{a})\cdot\boldsymbol{d}\\ &= (\boldsymbol{a}\cdot\boldsymbol{c})(\boldsymbol{b}\cdot\boldsymbol{d})-(\boldsymbol{b}\cdot\boldsymbol{c})(\boldsymbol{a}\cdot\boldsymbol{d}),\end{aligned} \tag{3.15}$$

and we get (for the 3D case) the *Lagrange identity* for the cross products of 3D vectors,

$$\langle(\boldsymbol{a}\times\boldsymbol{b}),(\boldsymbol{c}\times\boldsymbol{d})\rangle = \langle\boldsymbol{a},\boldsymbol{c}\rangle\langle\boldsymbol{b},\boldsymbol{d}\rangle-\langle\boldsymbol{b},\boldsymbol{c}\rangle\langle\boldsymbol{a},\boldsymbol{d}\rangle. \tag{3.16}$$

Note that the outer product is more general than the cross product, since it can be applied in spaces of any dimension and of 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 $\mathcal{G}_3$. A point, representing a position in the 3D space, can simply be expressed by a linear combination of the three basis vectors,

$$\boldsymbol{u} = u_1\mathbf{e}_1+u_2\mathbf{e}_2+u_3\mathbf{e}_3. \tag{3.17}$$

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 [5],

$$\begin{aligned}\boldsymbol{l} &= \boldsymbol{r}+\boldsymbol{x}\wedge\boldsymbol{r}\\ &= \boldsymbol{r}+\boldsymbol{m}.\end{aligned} \tag{3.18}$$

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

A plane can be represented by an entity one grade higher then the line. In terms of the Hesse distance d from the origin to the plane (coded by the Euclidean pseudo-scalar) and the unit bivector direction $\boldsymbol{n}$ from the origin to the plane, a plane is defined by

$$\boldsymbol{p} \;=\; \boldsymbol{n} + \boldsymbol{I}_E d. \tag{3.19}$$

Thus, a plane is an inhomogeneous multivector, consisting of a bivector and a trivector.

If we compare the representations of these three entities in EGA, we recognize, that those of lines and planes are more complicated than that of points. This has its reason in the fact that points are the basic geometric entities of the Euclidean space and all other entities have to be derived from points by means of certain operations (i.e. outer products) which are not defined in a vector space. In vector spaces, subspace concepts can be used to define these entities as (infinite) sets of points.

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. $(\boldsymbol{I}_E\mathbf{e}_3)\mathbf{e}_2 = \mathbf{e}_{123}\mathbf{e}_3\mathbf{e}_2 = \mathbf{e}_{12}\mathbf{e}_2 = \mathbf{e}_1$, or $(\boldsymbol{I}_E\mathbf{e}_1)\mathbf{e}_2 = \mathbf{e}_{123}\mathbf{e}_1\mathbf{e}_2 = \mathbf{e}_{23}\mathbf{e}_2 = -\mathbf{e}_1$, etc. Note, 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 so they can easily be identified with the unit vectors $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ of the quaternion algebra $\mathbb{H}$ with the famous Hamilton relations $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. We have the isomorph y $\boldsymbol{\mathcal{G}}_3^+ \simeq \mathbb{H}$ with $\boldsymbol{\mathcal{G}}_3^+$ as the even-grade subalgebra of $\boldsymbol{\mathcal{G}}_3$.

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 $\boldsymbol{R}\widetilde{\boldsymbol{R}} = 1$. Here $\widetilde{\boldsymbol{R}}$ stands for the reverse for $\boldsymbol{R}$. 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

$$\boldsymbol{R} \;=\; \underbrace{u_0}_{scalar} + \underbrace{u_1\mathbf{e}_{23} + u_2\mathbf{e}_{31} + u_3\mathbf{e}_{12}}_{bivectors}, \tag{3.20}$$

$$\widetilde{\boldsymbol{R}} \;=\; \underbrace{u_0}_{scalar} - \underbrace{u_1\mathbf{e}_{23} - u_2\mathbf{e}_{31} - u_3\mathbf{e}_{12}}_{bivectors}. \tag{3.21}$$

If we use the Euler representation of a rotor,

$$\begin{aligned} \boldsymbol{R} \;&=\; \exp\left(-\frac{\theta}{2}\boldsymbol{n}\right) \\ &=\; \cos\left(\frac{\theta}{2}\right) - \boldsymbol{n}\sin\left(\frac{\theta}{2}\right), \end{aligned} \tag{3.22}$$

it 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 $\theta \in \mathbb{R}$ is representing the amount of 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 $\boldsymbol{x}$,

$$\boldsymbol{x}' \;=\; \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}}. \tag{3.23}$$

A rotor is representing the group $SO(3)$ in EGA. Thus, the operation concatenates according to a left-sided product $\boldsymbol{R} = \boldsymbol{R}_2\boldsymbol{R}_1$ yielding a new rotor. From this follows

$$\boldsymbol{x}' \;=\; \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}} = (\boldsymbol{R}_2\boldsymbol{R}_1)\,\boldsymbol{x}\left(\widetilde{\boldsymbol{R}_1}\widetilde{\boldsymbol{R}_2}\right). \tag{3.24}$$

In contrast to rotation matrices of $\mathbb{R}^3$, rotors are working not only on points, but for all types of geometric objects, and is defined independent on the grade and the dimension of the space.

The exponential function of multivectors $\boldsymbol{m}$ can also be expressed via its series expression,

$$\exp(\boldsymbol{m}) \;=\; \sum_{k=0}^{\infty} \frac{\boldsymbol{m}^k}{k!}. \tag{3.25}$$

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

$$\begin{aligned} \frac{\partial \boldsymbol{R}}{\partial \theta} \;&=\; \frac{\partial \exp(-\frac{\theta}{2}\boldsymbol{n})}{\partial \theta} = -\frac{1}{2}\boldsymbol{n}\exp\left(-\frac{\theta}{2}\boldsymbol{n}\right) \\ &=\; -\frac{1}{2}\sin\left(\frac{\theta}{2}\right) - \frac{1}{2}\boldsymbol{n}\cos\left(\frac{\theta}{2}\right). \end{aligned} \tag{3.26}$$

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

$$\frac{\partial \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}}}{\partial \theta} \;=\; \frac{\partial \boldsymbol{R}}{\partial \theta}\boldsymbol{x}\widetilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x}\frac{\partial \widetilde{\boldsymbol{R}}}{\partial \theta} \tag{3.27}$$

In contrast to rotations, there exist no multiplicative way to formalize translation 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

$$\boldsymbol{x}' \;=\; \boldsymbol{x} + \boldsymbol{t}. \tag{3.28}$$

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 translations follow the rule $\boldsymbol{t} = \boldsymbol{t}_1 +$

$\boldsymbol{t}_2$. Another problem concerns the linearity of both operations. A rotation, $\mathcal{R}$, is a linear operation. Let be $\boldsymbol{x}$ and $\boldsymbol{y}$ any multivectors of $\boldsymbol{\mathcal{G}}_3$, then $\mathcal{R}\{\boldsymbol{x}+\boldsymbol{y}\} = \mathcal{R}\{\boldsymbol{x}\} + \mathcal{R}\{\boldsymbol{y}\}$. But translation behaves not linear. For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, representing points of $\langle\boldsymbol{\mathcal{G}}_3\rangle_1$, it follows $\mathcal{T}\{x+y\} \neq \mathcal{T}\{x\} + \mathcal{T}\{y\}$.

These different behaviors cause problems in representing the rigid motion of an object in EGA as linear operation. A rigid motion in Euclidean space is a mapping $\mathcal{D} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ which preserves distances between points and angles between vectors. In general, the movement of a rigid body, that is 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

$$\boldsymbol{x}' \;=\; \boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}} + \boldsymbol{t}. \tag{3.29}$$

A spatial rigid 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 $\mathcal{D} = (\boldsymbol{R}, \boldsymbol{t}) = (\boldsymbol{R_2}, \boldsymbol{t_2})(\boldsymbol{R_1}, \boldsymbol{t_1}) = (\boldsymbol{R}_2\boldsymbol{R}_1, \boldsymbol{R}_2\boldsymbol{t}_1 + \boldsymbol{t_2})$. But regrettably, because of the non-linear behavior of the translation, the displacement is no linear operation in $\boldsymbol{\mathcal{G}}_3$, neither for points nor for other entities. Fortunately, there are other algebraic embeddings which result in linearization with respects to points or other entities. While so-far either point or line based transformations for rigid displacements have been distinguished [27], we will introduce in this paper a third category which is based on spheres, see section 3.3.

3.2 The Projective Geometric Algebra

By using homogeneous coordinates we increase the dimension of the vector space by one and the corresponding algebra is of dimension $2^4 = 16$. The elements we gain are now scalars, vectors, bivectors, trivectors and the pseudoscalar. We use $\boldsymbol{\mathcal{G}}_{3,1}$ to represent the projective space. Here the additional basis vector $\mathbf{e}_-$ denotes the homogeneous component indicating the direction of projection. Because $\mathbf{e}_-^2 = -1$, this basis vector induces a Minkowski metric. The algebra $\boldsymbol{\mathcal{G}}_{3,1}$ contains the following elements,

$$\begin{aligned}\boldsymbol{\mathcal{G}}_{3,1} \;=\; span\{1\;, &\mathbf{e}_-, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3,\;\; \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12},\quad \mathbf{e}_{-1}, \mathbf{e}_{-2}, \mathbf{e}_{-3},\\ &\mathbf{e}_{123}, \mathbf{e}_{-23}, \mathbf{e}_{-31}, \mathbf{e}_{-12},\;\; \mathbf{e}_{-123} = \boldsymbol{I}_P\}.\end{aligned} \tag{3.30}$$

Note that e.g. $\mathbf{e}_{-123} \equiv \mathbf{e}_- \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3$, $\mathbf{e}_{-123}^2 = -1$.

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

In contrast to the Euclidean geometric algebra $\mathcal{G}_3$, in the projective geometric algebra $\mathcal{G}_{3,1}$ (PGA) we can simply represent points, lines and planes as r-blades, i.e. homogeneous multivectors of grade r. In that case the above mentioned duality operator 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 $\mathcal{G}_3$, can be represented in $\mathcal{G}_{3,1}$ by

$$\boldsymbol{X} = \boldsymbol{x} + \mathbf{e}_-. \tag{3.31}$$

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

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \\ &= (\boldsymbol{x}_1 + \mathbf{e}_-) \wedge (\boldsymbol{x}_2 + \mathbf{e}_-) \\ &= \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 + (\boldsymbol{x}_1 - \boldsymbol{x}_2)\mathbf{e}_- \\ &= \boldsymbol{m} + \boldsymbol{r}\mathbf{e}_-. \end{aligned} \tag{3.32}$$

The line $\boldsymbol{L}$ contains the moment $\boldsymbol{m}$ and direction $\boldsymbol{r}$. Therefore, it corresponds directly to the Plücker representation [5]. 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

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \wedge \boldsymbol{X}_3 \\ &= (\boldsymbol{x}_1 + \mathbf{e}_-) \wedge (\boldsymbol{x}_2 + \mathbf{e}_-) \wedge (\boldsymbol{x}_3 + \mathbf{e}_-) \\ &= \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 \wedge \boldsymbol{x}_3 + (\boldsymbol{x}_1 - \boldsymbol{x}_2) \wedge (\boldsymbol{x}_1 - \boldsymbol{x}_3)\mathbf{e}_- \\ &= d\boldsymbol{I}_E + \boldsymbol{n}\mathbf{e}_- \end{aligned} \tag{3.33}$$

This representation corresponds to the Hesse description of planes, formalizing a plane by the normal $\boldsymbol{n}$ of the plane, and the Hesse distance d of the plane to the origin.

As can be seen, the generation of the higher order entities is much more natural than in the algebra of the Euclidean space because it results from the incidence algebra of points.

There are two basic operations of the incidence algebra, the *join* and the *meet*.

The outer product of two blades is non-vanishing iff their supports have zero intersection. This can be used to prove an incidence relation [17], e.g. a point $\boldsymbol{X}$ is on a line $\boldsymbol{L}$ iff

$$\boldsymbol{X} \wedge \boldsymbol{L} \;=\; 0. \tag{3.34}$$

For blades $\boldsymbol{A}$ and $\boldsymbol{B}$ we use the previous defined shuffle product and the join, to express the *meet* operations:

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ two arbitrary blades and let $\boldsymbol{J} = \boldsymbol{A} \dot{\wedge} \boldsymbol{B}$, then the meet, symbolized by $\vee$, is defined as

$$(\boldsymbol{A} \vee \boldsymbol{B}) \;:=\; \left(\boldsymbol{A}\boldsymbol{J}^{-1} \wedge \boldsymbol{B}\boldsymbol{J}^{-1}\right)\boldsymbol{J}. \tag{3.35}$$

The meet is the common factor of $\boldsymbol{A}$ and $\boldsymbol{B}$ with highest grade. The meet defines a generalized intersection operation. It is the shuffle product of two blades, applied on their join as pseudo-scalar. Because of the similarity between the shuffle product and the meet product, we define no extra symbol for the meet product.

Thus the meet and wedge operators provide the desired operations in the algebra of subspaces of a vector space. The wedge product 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 re-transform e.g. a projective point to a Euclidean point the *projective split* [15] has to be applied.

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 provided by the subspace concepts. From this results a nice formulation of the duality concept in projective geometry and to compact descriptions of joins and meets of subspaces, just by applying a suitable operator.

In PGA projective transformations can be expressed. These transformations are more general than Euclidean transformations, since they include also other transformations like scaling or shearing. Since we are only interested in Euclidean transformations, we have to restrict the projective transformations in a second processing step. So we need an algebraic embedding which enables the restriction of the transformations on a Euclidean transformation in a better way. The common used algebra so far is the *dual quaternion* algebra, which is isomorph to the motor algebra $\boldsymbol{\mathcal{G}}^{+}_{3,0,1}$ [2]. But since it contains null spaces, the duality concepts of projective geometry cannot be applied any more[2]. The aim is now, to proceed to the conformal algebra,

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

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 artificially generated, 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

We use the *conformal geometric algebra* [22] to model the geometry of our scenario for pose estimation.

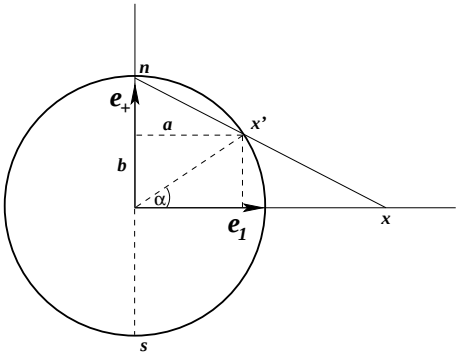

Fig. 3.2: Visualization of a stereographic projection for the 1D case: Points on the line are projected on the circle. Note that the point at infinity projects to the north pole $\boldsymbol{n}$, and the origin projects to the south pole $\boldsymbol{s}$.

3.3.1 Stereographic projection

The idea behind conformal geometry is to interpret points as *stereographic projected* points. Simply speaking, a stereographic projection is one way to make 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 nice geometric description and is visualized for the 1D case in figure 3.2: 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. A visualization for the 2D case is shown in figure 3.5. Before introducing a formalization in terms of geometric algebra, we want to repeat the basic formulas for projecting points in space

on the sphere and vice versa, e.g. given in [24]. To simplify the calculations, we will restrict ourselves to the 1-D case, as shown in figure 3.2. We assume two orthonormal basis vectors $\{\mathbf{e}_1, \mathbf{e}_+\}$ and assume the radius of the circle as $\rho = 1$. Note that $\mathbf{e}_+$ is an additional vector to the one-dimensional vector space $\mathbf{e}_1$ with $\mathbf{e}_+^2 = \mathbf{e}_1^2 = 1$.

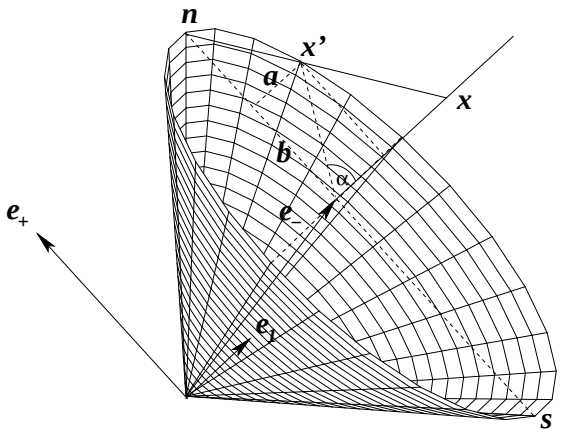

Fig. 3.3: Visualization of the homogeneous model for stereographic projections for the 1D case. All stereographic projected points are on a cone, which is a null-cone in the Minkowski space. Note that in comparison to figure 3.2, the coordinate axes are rotated and perspective drawn.

To project a point $\boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+$ on the sphere onto the $\mathbf{e}_1$-axis, the interception theorems can be applied to obtain

$$\boldsymbol{x} = \left(\frac{a}{1-b}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{3.36}$$

To project a point $x\mathbf{e}_1$ ($x \in \mathbb{R}$) onto the circle we have to estimate the appropriate factors $a, b \in [0, \ldots, 1]$. The vector $\boldsymbol{x}'$ can be expressed as

$$\begin{aligned} \boldsymbol{x}' &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+, \end{aligned} \tag{3.37}$$

and using homogeneous coordinates this leads to a homogeneous representation of the point on the circle as

$$\boldsymbol{x}' = x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}_3. \tag{3.38}$$

Thus, the vector $\boldsymbol{x}$ is mapped to

$$\boldsymbol{x} \quad \Rightarrow \quad \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_3. \tag{3.39}$$

We define $\mathbf{e}_3$ to have a negative signature, and therefore replace $\mathbf{e}_3$ with $\mathbf{e}_-$, whereby $\mathbf{e}_-^2 = -1$. This has the advantage that in addition to using a homogeneous representation of points, we are also working in a Minkowski space. Euclidean points, stereographically projected onto the circle in figure 3.2, are then represented by the set of null vectors in our new space. That is, we have the mapping

$$\boldsymbol{x} \quad \Rightarrow \quad \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_-, \tag{3.40}$$

with

$$(\boldsymbol{x}')^2 = a^2 + b^2 - 1 = 0 \tag{3.41}$$

since (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 [22] 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 n-dimensional Euclidean space is represented by a spinor in $\boldsymbol{\mathcal{G}}_{n+1,1}$, the conformal geometric algebra. Figure 3.3 visualizes the homogeneous model for stereographic projections for the 1D case.

Substituting the expressions for a and b from equation (3.37) into equation (3.40), we get

$$\boldsymbol{x}' \quad = \quad x\mathbf{e}_1 + \frac{1}{2}\left(x^2 - 1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2 + 1\right)\mathbf{e}_-. \tag{3.42}$$

This homogeneous representation of a point is used as point representation in the conformal geometric algebra. We will show this in the next section. Note that the stereographic projection leads to points on a sphere. Therefore, we can use (special) rotations on this sphere to model e.g. translations in the world or rigid body motion as coupled rotation/translation. Since we also use a homogeneous embedding, we have furthermore the possibility to model projective geometry.

3.3.2 Definition of the conformal geometric algebra

To introduce CGA we follow [22] and start with a *Minkowski plane*, $\boldsymbol{\mathcal{G}}_{1,1}$, whose vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the

properties

$$\mathbf{e}_+^2 = 1 \quad \mathbf{e}_-^2 = -1 \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0 \tag{3.43}$$

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

$$\mathbf{e}_0 := \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} := \mathbf{e}_- + \mathbf{e}_+ \tag{3.44}$$

These vectors can be interpreted as the origin, $\mathbf{e}_0$, of the coordinate system and the point at infinity, $\mathbf{e}$, respectively. Note that this is in consistency with figure 3.3: $\mathbf{e}_0$ corresponds to the south pole and $\mathbf{e}$ corresponds to the north pole in homogeneous coordinates. Furthermore, we define $\boldsymbol{E} := \mathbf{e} \wedge \mathbf{e}_0 = \mathbf{e}_+ \wedge \mathbf{e}_-$.
For these elements the following straightforwardly proved properties can be summarized as

$$\begin{aligned} \mathbf{e}_0^2 = \mathbf{e}^2 = 0 \quad \mathbf{e} \cdot \mathbf{e}_0 = -1 \quad \boldsymbol{E} = \mathbf{e}_+\mathbf{e}_- \\ \boldsymbol{E}^2 = 1 \quad \boldsymbol{E}\mathbf{e} = -\mathbf{e} \quad \boldsymbol{E}\mathbf{e}_0 = \mathbf{e}_0 \\ \mathbf{e}_+\boldsymbol{E} = \mathbf{e}_- \quad \mathbf{e}_-\boldsymbol{E} = \mathbf{e}_+ \quad \mathbf{e}_+\mathbf{e} = \boldsymbol{E} + 1 \\ \mathbf{e}_-\mathbf{e} = -(\boldsymbol{E} + 1) \quad \mathbf{e} \wedge \mathbf{e}_- = \boldsymbol{E} \quad \mathbf{e}_+ \cdot \mathbf{e} = 1 \end{aligned} \tag{3.45}$$

The role of the Minkowski plane is to generate null vectors, and so to extend a 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 $\mathbb{R}^{4,1}$. Its 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. We denote the conformal unit pseudo-scalar as

$$\boldsymbol{I}_C \;=\; \mathbf{e}_{+-123} = \boldsymbol{E}\boldsymbol{I}_E. \tag{3.46}$$

In this algebra we consider points of the so-called null cone, which fulfill the properties

$$\{\underline{\boldsymbol{x}} \in \mathbb{R}^{4,1} | \underline{\boldsymbol{x}}^2 = 0, \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1\}. \tag{3.47}$$

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

$$\underline{\boldsymbol{x}} \;=\; \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0. \tag{3.48}$$

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

$$\begin{aligned} \underline{\boldsymbol{x}} &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0 \\ &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2(\mathbf{e}_+ + \mathbf{e}_-) + \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \\ &= \boldsymbol{x} + \left(\frac{1}{2}\boldsymbol{x}^2 - \frac{1}{2}\right)\mathbf{e}_+ + \left(\frac{1}{2}\boldsymbol{x}^2 + \frac{1}{2}\right)\mathbf{e}_-. \end{aligned} \tag{3.49}$$

This is exactly the homogeneous representation of a stereographic projected point, given in (3.42). The basis vectors $\{\mathbf{e}, \mathbf{e}_0\}$ only allow for a more compact representation of vectors than when using $\{\mathbf{e}_+, \mathbf{e}_-\}$.

We will now analyze new characteristic properties of the points, and so of the generated entities from these *points*.

3.3.3 Geometric entities in conformal geometric algebra

The use of a certain geometric algebra induces an involved metric and therewith a *basis geometric entity* from which the other entities are derived. In $\mathcal{G}_3$, the algebra of the Euclidean space, the basis entities are points, and lines and planes are formulated as certain sets of points. In the motor algebra $\mathcal{G}^+_{3,0,1}$, an algebra to model kinematics [2], the basis entities are lines, expressed in terms of the Plücker coordinates [5], and points and planes are written in these terms. In conformal geometric algebra, $\mathcal{G}_{4,1}$, the spheres are the basis entities [24] from which the other entities are derived. It turns out that the above introduced point representation is nothing more than a degenerate sphere.

To introduce primitive geometric entities in CGA we will start by introducing the representation of spheres in CGA. Then we will proceed to the other entities. A more detailed introduction can be found in [22].

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

$$\begin{aligned} (\boldsymbol{x}-\boldsymbol{p})^2 &= \rho^2 \\ \Leftrightarrow \boldsymbol{x}^2 - (\boldsymbol{x}\boldsymbol{p}+\boldsymbol{p}\boldsymbol{x}) + \boldsymbol{p}^2 &= \rho^2. \end{aligned} \tag{3.50}$$

The basis entities of the 3D conformal space are spheres $\underline{\boldsymbol{s}}$, containing the center $\boldsymbol{p}$ and the radius ρ, $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0$. 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 $\mathcal{G}_{4,1}$ equation (3.50) can therefore be represented more compact:

$$\begin{aligned} (\boldsymbol{x}-\boldsymbol{p})^2 &= \rho^2 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} &= 0. \end{aligned} \tag{3.51}$$

This can easily be verified,

$$\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} = (\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0) \cdot (\boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0)$$

$$\begin{aligned} &= -\frac{1}{2}(\boldsymbol{x}^2 + \boldsymbol{p}^2 - \rho^2) + \boldsymbol{x} \cdot \boldsymbol{p} \\ &= -\frac{1}{2}((\boldsymbol{x} - \boldsymbol{p})^2 - \rho^2). \end{aligned} \tag{3.52}$$

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

$$\underline{\boldsymbol{s}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} \wedge \underline{\boldsymbol{d}}, \tag{3.53}$$

and a point $\underline{\boldsymbol{x}}$ is on a sphere $\underline{\boldsymbol{s}}$ iff $\underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star = 0$. Note: To test incidence of a point with an entity can be expressed by the *inner product null-space* or *outer product null-space*, dependent on the representation or dual representation of the entity. This follows from the easy relationship (see e.g. [15])

$$\begin{aligned} \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star &= 0. \end{aligned} \tag{3.54}$$

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

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

$$\underline{\boldsymbol{x}} \in \underline{\boldsymbol{z}} \;\Leftrightarrow\; \underline{\boldsymbol{x}} \in \underline{\boldsymbol{s}}_1 \text{ and } \underline{\boldsymbol{x}} \in \underline{\boldsymbol{s}}_2. \tag{3.55}$$

Since $\underline{\boldsymbol{s}}_1$ and $\underline{\boldsymbol{s}}_2$ can be assumed as linear independent, we can write

$$\begin{aligned} \underline{\boldsymbol{x}} &\in \underline{\boldsymbol{z}} \\ \Leftrightarrow (\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}}_1)\underline{\boldsymbol{s}}_2 - (\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}}_2)\underline{\boldsymbol{s}}_1 &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underbrace{(\underline{\boldsymbol{s}}_1 \wedge \underline{\boldsymbol{s}}_2)}_{\underline{\boldsymbol{z}}} &= 0 \\ \Leftrightarrow \underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{z}} &= 0 \end{aligned} \tag{3.56}$$

This means that algebraically a circle can be expressed as the join of two spheres. Figure 3.4 visualizes the generation of a circle as intersection of two spheres. The intersection of the circle with a third sphere leads to a point pair[3].

In the dual form circles are geometrically defined by three points on it,

$$\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}. \tag{3.57}$$

[3] This figure is taken from the visualization tool for Clifford algebra, CLUDraw [8].

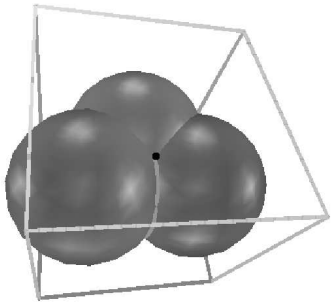

Fig. 3.4: A circle can be expressed as intersection of two spheres. Intersecting the circle with a third sphere leads to two points (only one of these two points is visible).

Entity	Representation	Grade	Dual representation	Grade
Sphere	$\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}-\rho)^2\mathbf{e} + \mathbf{e}_0$	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	$\underline{\boldsymbol{P}} = \boldsymbol{n}\boldsymbol{I}_E - d\mathbf{e}$	1	$\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$	4
	$\boldsymbol{n} = (\boldsymbol{a}-\boldsymbol{b}) \wedge (\boldsymbol{a}-\boldsymbol{c})$			
	$d = (\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c})\boldsymbol{I}_E$			
Line	$\underline{\boldsymbol{L}} = \boldsymbol{r}\boldsymbol{I}_E + \mathbf{e}\boldsymbol{m}\boldsymbol{I}_E$	2	$\underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$	3
	$\boldsymbol{r} = \boldsymbol{a} - \boldsymbol{b}$			
	$\boldsymbol{m} = \boldsymbol{a} \wedge \boldsymbol{b}$			
Circle	$\underline{\boldsymbol{z}} = \underline{\boldsymbol{s}_1} \wedge \underline{\boldsymbol{s}_2}$	2	$\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$	3
	$\underline{\boldsymbol{P}_z} = \underline{\boldsymbol{z}} \cdot \mathbf{e}$, $\underline{\boldsymbol{L}_z}^\star = \underline{\boldsymbol{z}} \wedge \mathbf{e}$			
	$\underline{\boldsymbol{p}_z} = \underline{\boldsymbol{P}_z} \vee \underline{\boldsymbol{L}_z}$, $\rho = \frac{\underline{\boldsymbol{z}}^2}{(\mathbf{e} \wedge \underline{\boldsymbol{z}})^2}$			
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.

Evaluating the outer products of three points leads to

$$\underline{\boldsymbol{z}}^\star \;=\; \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} = A + A^-\mathbf{e} + A^+\mathbf{e}_0 + A^\pm\boldsymbol{E}, \tag{3.58}$$

with

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

The dual form of lines are represented by the outer product of two points on the line and the point at infinity (see [24]), $\underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$. Since the outer product of three points determines a circle [22], the line can be interpreted as a circle passing through the point at infinity.

Similar to lines, dual planes can then be defined by the outer product of three points on the plane and the point at infinity, $\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$. A

plane is a degenerate sphere, containing the point at infinity.

An overview of the definitions of the entities, their dual representations and their grades are 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.

The dual lines and planes are given, similar to $\boldsymbol{\mathcal{G}}_{3,1}$, by the Plücker coordinates of lines (direction $\boldsymbol{r}$ and moment $\boldsymbol{m}$) and the Hesse formulation (normal $\boldsymbol{n}$ and directed distance $d\boldsymbol{I}_E$) of planes, respectively.

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

3.3.4 Conformal transformations

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

$$\sigma\underline{\boldsymbol{x}}' \;=\; G\underline{\boldsymbol{x}}(G^{\star})^{-1}, \tag{3.59}$$

where G is a versor and σ a scalar. Since the null cone is invariant under G, i.e. $(\underline{\boldsymbol{x}}')^2 = \underline{\boldsymbol{x}}^2 = 0$, we have to apply a scale factor σ to ensure $\underline{\boldsymbol{x}}' \cdot \mathbf{e} = \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1$. Table 3.2, taken from [22], summarizes the conformal transformations. The first row shows the type of operation performed with the versor product. The second row shows as example the result of a transformation acting on a point. The third row shows the versor, which has to be applied and the last row shows the scaling parameter 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 we see, any conformal transformation covers several more simple geometric transformations. More explanations of the conformal group can also be found in [24, 11]. It is shown in e.g. [13], that in $\boldsymbol{\mathcal{G}}_3$, the rotations are generated by reflections and similarly one can ask, what does a reflection mean for the stereographic projected point. This is visualized in figure 3.5 for the 2D case: A reflection of a point on the sphere with respect to the 2D plane leads to a new point on the sphere, which corresponds to the inverse of the point on the 2D plane. This means, that the basic operation is an inversion, and the other operations are derived from it. In figure 3.5 it is also shown, what a translation of a point on the 2D plane means for a corresponding point: A translation corresponds to a special rotation along an axis in the 2D plane. It is also easy to imagine, that a rotation in 2D plane is exactly the same for a stereographic projected point. This means, that a rotation can be estimated

Type	$G(\boldsymbol{x})$ on $\mathbb{R}^n$	Versor in $\mathbb{R}_{n+1,1}$	σ
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	$\boldsymbol{x} - \boldsymbol{a}$	$\boldsymbol{T}_a = 1 + \frac{1}{2}\boldsymbol{a}\mathbf{e}$	1
Transversion	$\frac{\boldsymbol{x}-\boldsymbol{x}^2\boldsymbol{a}}{\sigma(x)}$	$\boldsymbol{K}_a = 1 + \boldsymbol{a}\mathbf{e}_0$	$1 - 2\boldsymbol{a}\cdot\boldsymbol{x} + \boldsymbol{x}^2\boldsymbol{a}^2$
Dilation	$\lambda\boldsymbol{x}$	$\boldsymbol{D}_\lambda = \exp(-\frac{1}{2}\boldsymbol{E}(\ln\lambda))$	λ^{-1}
Involution	$\boldsymbol{x}^\star = -\boldsymbol{x}$	$\boldsymbol{E}$	-1

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

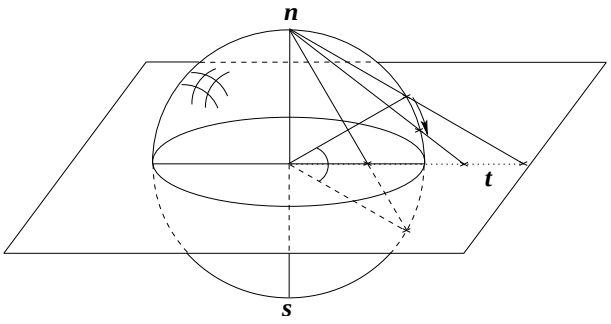

Fig. 3.5: Visualization of an inversion and translation for a stereographic projected point in the 2D case.

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 $\boldsymbol{\mathcal{G}}_{4,1}$. This is the reason, why kinematics can be described in this model in a linear manner.

We will now concentrate on expressing rotations and translations in CGA.

3.3.5 Rigid motions in conformal geometric algebra

This section concerns the formulation of *rigid body motions* in CGA. 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. To describe a Euclidean transformation in a linear manner it makes necessary to have access on a multiplicative coupling of rotation and translation. Since the conformal transformation contains the Euclidean transformation, we can use the conformal group to express rigid body motions. Note: Though the conformal group is more general than the Euclidean group, for our pose estimation scenario, it is sufficient to concen-

trate only on this subset of transformations.

So far, we can use rotors as elements of $\mathcal{G}_3^+$ to formalize 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 $\mathcal{G}_3$ (see section 3.1.2), rotations in $\mathcal{G}_{4,1}$ are represented by rotors, $\boldsymbol{R} = \exp\left(\frac{\theta}{2}\boldsymbol{l}\right)$. The components of the rotor $\boldsymbol{R} \in \mathcal{G}_{4,1}^+$ are, similar to section 3.1.2, the unit bivector $\boldsymbol{l}$ which represents the dual of the rotation axis, and the angle θ, 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 $\widetilde{\boldsymbol{R}}$. E.g., a rotation of a point can be written as $\underline{\boldsymbol{x}}' = \boldsymbol{R}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{R}}$.

If we want to translate an entity with respect to a translation vector $\boldsymbol{t} \in \langle \mathcal{G}_3 \rangle_1$, we can use a so called *translator*, $\boldsymbol{T} \in \mathcal{G}_{4,1}^+$, $\boldsymbol{T} = (1 + \frac{\mathbf{e}\boldsymbol{t}}{2}) = \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right)$. As mentioned previously, a translator is a special rotor and given in a null space since $\mathbf{e}^2 = 0$. Similar to rotations we can translate entities by multiplying the entity from the left with the translator $\boldsymbol{T}$ and with its reverse $\tilde{\boldsymbol{T}}$ from the right,

$$\underline{\boldsymbol{x}}' = \boldsymbol{T}\underline{\boldsymbol{x}}\tilde{\boldsymbol{T}}. \tag{3.60}$$

To express a rigid body motion, we can apply rotors and translators consecutively. We denote such an operator,

$$\boldsymbol{M} = \boldsymbol{R}\boldsymbol{T}, \tag{3.61}$$

it is a special even-grade multivector, as a motor, which is an abbreviation of "moment and vector"[3]. The rigid body motion of e.g. a point $\underline{\boldsymbol{X}}$ can be written as

$$\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}, \tag{3.62}$$

see also [16].

This formalization of a rigid displacement can not only be applied to points or lines (see [27]), 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 [3, 2], the algebra $\mathcal{G}_{3,0,1}^+$, which is formulating kinematics in a space composed of lines and which is isomorph to the dual quaternion algebra. Although equation (3.61) 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 we do not have to make any sign changes, depending on the entity, where the motor has to act on.

This makes several case decisions in the previous formalizations of kinematics unnecessary and thus, the calculations will become more easy.

The reason for that increased symmetry of a motor action lies in our chosen algebraic embedding.

3.3.6 Twist and screw transformations

We will now derive a further definition of a motor in CGA based on the so-called *twist transformation*. Following e.g. [23], every rigid body motion can be expressed by a twist transformation, which is a rotation about a line in space (in general not passing the origin). This results from the fact, that for every $g \in SE(3)$ exists a $\xi \in se(3)$ and a $\theta \in \mathbb{R}$ such that $g = \exp(\xi\theta)$[4]. In CGA we use the rotors and translators to express twist transformations in the space. To model a rotation of a point $\underline{\boldsymbol{X}}$ around a line $\underline{\boldsymbol{L}}$ in the space, the general idea is to translate both, the point and the line, to the origin, to perform a rotation and to translate them back. So an alternative definition of a motor $\boldsymbol{M} \in \mathcal{G}^{+}_{4,1}$ describing a twist transformation has the general form

$$\boldsymbol{M} \;=\; \boldsymbol{T}\boldsymbol{R}\tilde{\boldsymbol{T}}, \tag{3.63}$$

denoting the inverse translation, rotation and back translation, respectively. Using the exponential form of the translators and rotors, we can write[5]

$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R}\tilde{\boldsymbol{T}} \\
&= \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\exp\left(-\frac{\theta}{2}\boldsymbol{l}\right)\exp\left(-\frac{\mathbf{e}\boldsymbol{t}}{2}\right) \\
&= \left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\exp\left(-\frac{\theta}{2}\boldsymbol{l}\right)\left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right) \\
&= \exp\left(\left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\left(-\frac{\theta}{2}\boldsymbol{l}\right)\left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right)\right) \\
&= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right).
\end{aligned} \tag{3.64}$$

This formulation corresponds to the one for a general motor given in [22]. We use an exponential representation of this motor since then it is

[4] $SE(3)$ denotes the *special Euclidean* group, $SE(3) = \{(\mathcal{R}, \boldsymbol{t}) : \mathcal{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3\}$ and $se(3)$ denotes its corresponding Lie algebra, $se(3) = \{(\omega, \boldsymbol{v}) : \omega \in so(3), \boldsymbol{v} \in \mathbb{R}^3\}$, see more details in [23].

[5] In the fourth equation we make use of the nice 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 proved by induction on the series expression of the exponential function.

more easy to estimate its derivative. We need to estimate derivatives in the later introduced linearization process to get a good conditioned system of equations.

It is interesting to mention that the exponential part of the motor $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}\tilde{\boldsymbol{T}}$ consists directly of the line components, we want to rotate our entities around. To show this, we will first recall the description of a dual line $\underline{\boldsymbol{L}}^\star$:

$$\begin{aligned} \underline{\boldsymbol{L}}^\star &= \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \\ &= \mathbf{e}(\boldsymbol{a} \wedge \boldsymbol{b}) + (\boldsymbol{b} - \boldsymbol{a})\boldsymbol{E}. \end{aligned} \tag{3.65}$$

Using the unit direction $\boldsymbol{n}$ and the plumb-point $\boldsymbol{t}$ of the origin to the line, we can write the line as

$$\underline{\boldsymbol{L}}^\star = \mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n}) + \boldsymbol{n}\boldsymbol{E}. \tag{3.66}$$

Multiplying with $\boldsymbol{I}_C$ (from the right) leads to

$$\begin{aligned} (\mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n}) + \boldsymbol{n}\boldsymbol{E})\boldsymbol{I}_C &= (\mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n})\boldsymbol{E}\boldsymbol{I}_E + \boldsymbol{n}\boldsymbol{E})\boldsymbol{E}\boldsymbol{I}_E \\ &= (\mathbf{e}(\boldsymbol{t} \wedge \boldsymbol{n})\boldsymbol{I}_E + \boldsymbol{n}\boldsymbol{I}_E \\ &= \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{n}\boldsymbol{I}_E) + \boldsymbol{n}\boldsymbol{I}_E \\ &= \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l}) + \boldsymbol{l}, \end{aligned} \tag{3.67}$$

since the direction $\boldsymbol{n}$ of the line corresponds to the rotation axis $\boldsymbol{l} = \boldsymbol{n}\boldsymbol{I}_E$.

This means that we only have to multiply the five dimensional pseudo-scalar $\boldsymbol{I}_C = \boldsymbol{E}\boldsymbol{I}_E$ to get the line. Vice versa: Given the dual line $\underline{\boldsymbol{L}}^\star$ (with unit direction) in the space, the corresponding motor describing a twist transformation around this line is given by

$$\begin{aligned} \boldsymbol{M} &= \exp\left(-\frac{\theta}{2}\boldsymbol{I}_C\underline{\boldsymbol{L}}^\star\right) \\ &= \exp\left(-\frac{\theta}{2}\underline{\boldsymbol{L}}\right), \end{aligned} \tag{3.68}$$

Note: The line $\underline{\boldsymbol{L}}$ must then be scaled with respect to the direction, $\|\boldsymbol{r}\| = 1$, since the scaling of the line is directly connected to the amount of the rotation θ. We will use the representation $\boldsymbol{M} = \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l})\right)\right)$ for the linearization of the constraint equations in part II.

Screw transformations can also 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 about an axis combined with a translation parallel to that axis, see also [27, 23]. This is called a *screw transformation.* A

screw transformation is defined by a rotation axis $\boldsymbol{l}$ (a bivector), a pitch h and a magnitude θ. The *pitch* of the screw is the ratio of translation to rotation, $h := \frac{d}{\theta}$ $(\theta \neq 0)$. If $h \to \infty$, then the corresponding screw transformation consists of a pure translation along the axis of the screw. The principle of a screw transformation is visualized in figure 3.6. To model

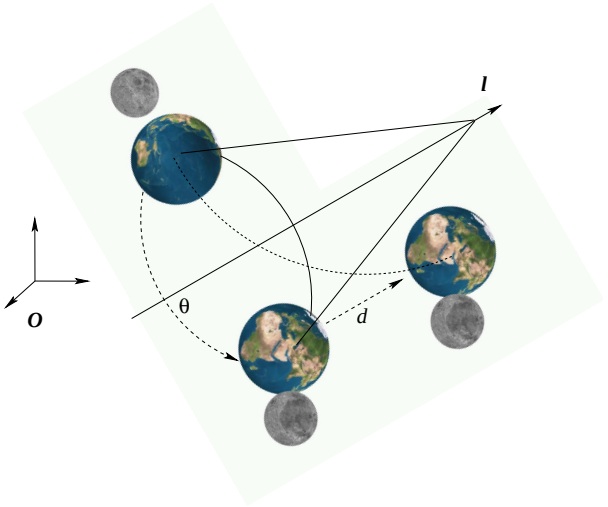

Fig. 3.6: Visualization of a screw transformation

a screw transformation, we only have to translate the entity after a twist motion with respect to the rotation axis, leading to a motor representation

$$\boldsymbol{M} \;=\; \boldsymbol{T}_{\lambda \boldsymbol{l}} \boldsymbol{T}_{\boldsymbol{t}} \boldsymbol{R}_{\boldsymbol{l}} \widetilde{\boldsymbol{T}_{\boldsymbol{t}}}. \tag{3.69}$$

The bivector $\boldsymbol{l}$ denotes the rotation plane $\boldsymbol{R}_{\boldsymbol{l}}$ is acting on, and $\boldsymbol{t}$ denotes the translation vector of the translator $\boldsymbol{T}$. The vector $\lambda \boldsymbol{l}$ in $\boldsymbol{T}_{\lambda \boldsymbol{l}}$ denotes the pitch during the screw transformation. Note, that here the dual of the bivector (leading to a vector) is acting in the translator as translation vector. The reader should consult [2], to find a more detailed analysis of screws, given in the motor algebra, which can also be adopted to the conformal geometric algebra.

For our pose estimation algorithm, we prefer the interpretation of a motor $\boldsymbol{M}$ as a twist, since we make use of the property $\boldsymbol{g}\ \exp(\xi)\ \tilde{\boldsymbol{g}} = \exp(\boldsymbol{g}\xi\tilde{\boldsymbol{g}})$ for $\boldsymbol{g}\tilde{\boldsymbol{g}} = 1$. This symmetry property enables us to linearize our equations more easily, than interpreting $\boldsymbol{M}$ as $\boldsymbol{M} = \boldsymbol{R}\boldsymbol{T}$, or $\boldsymbol{M} = \boldsymbol{T}_{\lambda \boldsymbol{l}} \boldsymbol{T}_{\boldsymbol{t}} \boldsymbol{R}_{\boldsymbol{l}} \widetilde{\boldsymbol{T}_{\boldsymbol{t}}}$.

4. THE POSE PROBLEM IN CONFORMAL GEOMETRY

Let us recall figure 2.1 for the 2D-3D pose estimation problem: We assume the knowledge of a 3D object model and observe it in an image of a calibrated camera. The aim is to find the rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ of the object, which leads to the best fit of the reference model with the actual extracted entities. To describe the pose scenario, it is crucial to interact entities between mathematical spaces involved in the pose problem.

4.1 Interacting Entities in Euclidean, Projective and Conformal Geometry

So far we are able to use CGA for the formalization of involved entities and their transformations. To formalize the scenario of figure 2.1 in a suitable way, the aim is now to describe the interaction of projective and conformal geometry. As mentioned earlier, the interaction of the different strata of the hierarchy is only poorly lit in the last years. E.g., Ruf [32] concerns this problem, but only for point features in the framework of matrix calculus. We want to extend the problem to more general object features and use in this context the conformal geometric algebra. To enable interaction between strata we use algebras for the projective and Euclidean space, respectively, as subalgebras of the CGA. It turns out that it is possible to switch between entities between conformal and projective representations by using multiplicative operators.

The main strategy to estimate the pose of the rigid object of figure 2.1 is very simple. It is summarized in figure 4.1 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 projective reconstructed and converted in a conformal representation. Then the model features are transformed in the conformal space. To get a distance measure in the Euclidean

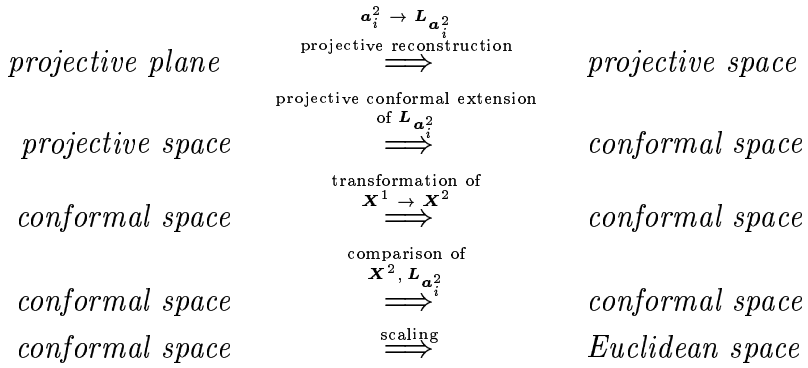

Fig. 4.1: Involved geometric spaces of the 2D-3D pose estimation problem.

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

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

space, in the last step, the transformed model entities and reconstructed image features are compared by suitable scaled constraint equations.

We can summarize the involved mathematical spaces and their corresponding geometric algebras in the following manner: The Euclidean framework can be represented by using the algebra $\mathcal{G}_{3,0}$, and $\mathcal{G}_{3,1}$ can be used to represent the projective space [17]. The projective plane is represented by the algebra $\mathcal{G}_{2,1}$. One way of defining a kinematic space is given by the motor algebra $\mathcal{G}^+_{3,0,1}$ [3, 2]. Another way is given by embedding the kinematics into the 3D conformal space represented by $\mathcal{G}_{4,1}$ [22]. Table 4.1 gives an overview of representations of points using different algebras. As can be seen, the relation

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

is valid, but only for $\mathcal{G}_{3,0}$ 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 $\mathcal{G}_{3,0}$ the *direction* of

modeling the pose problem is consistent with the increasing possibilities by using higher geometric algebras: Reconstruct from the projective plane one dimensional higher entities and work in the projective or conformal space, respectively. In these spaces we have more possibilities of expressing geometry. Therefore the modeling of the pose problem follows the direction

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

In the following, we will introduce operators which not only relate linear spaces of the considered algebras but guarantee the mapping between the algebraic properties. This means, we define operators 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 enables us to pick up the advantages of each algebra, and so to use the better suited algebra for each subproblem.

4.2 Change of Representations of Geometric Entities

In this section it will be shown, how to transform these representations. The operators between the conformal and projective space will be denoted as *conformal projective split* and *projective conformal extension*, according to the *projective split* [17] which enables a change between the projective and the Euclidean space and the *conformal split* [16, 17] which enables a change between the Euclidean and conformal space. This means, 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.

We will start with the first two spaces, the conformal space to describe kinematics and conformal geometry, and the projective space which can be considered as subspace of the former one. The two operators to switch between geometric algebras representing these spaces are summarized in the following theorems:

Theorem 4.2.1 *To change an entity Θ given in the projective representation, Θ_p, to the conformal representation, Θ_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:*

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

Note: Since circles and spheres are no entities of the projective space, we can not transform them between the projective and conformal space. This

leads to remarkable consequences for the pose estimation problem, discussed in the later sections.

For the proof of the theorem it is sufficient to show the simple 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 make the involved geometry more clear, we will describe the representation changes of points, lines and planes 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, we describe how to switch representations from the conformal space into the projective space.

Theorem 4.2.2 *To change an entity Θ, given in the conformal representation, Θ_c, to the projective representation, Θ_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:*

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

For the proof it is sufficient to show the following identity
$\Theta_p = \mathbf{e}_+ \cdot (\mathbf{e} \wedge \Theta_p)$,

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

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

The transformation 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, these are only artificially generated representations which are not generated by the algebra itself. As a consequence, only for points the transformation can be described in a suitable way. The transformations are leaned on Hestenes' formalization in [17] and can be written in the following way,

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

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

Euclidean space		projective space		conformal space
$\boldsymbol{\mathcal{G}}_{3,0}$	$\subseteq$	$\boldsymbol{\mathcal{G}}_{3,1}$	$\subseteq$	$\boldsymbol{\mathcal{G}}_{4,1}$
Θ_e	$\underset{\longleftarrow}{\overset{\boldsymbol{x}+\mathbf{e}_-}{\longrightarrow}}$ $\dfrac{(\boldsymbol{X}\wedge\mathbf{e}_-)\cdot\mathbf{e}_-}{\boldsymbol{X}\cdot\mathbf{e}_-}$	Θ_p	$\underset{\mathbf{e}_+\cdot\Theta_c}{\overset{\mathbf{e}\wedge\Theta_p}{\longrightarrow}}$ $\longleftarrow$	Θ_c
Θ_e	$\underset{\longleftarrow}{\longrightarrow}$	$\mathbf{e}\wedge(\boldsymbol{x}+\mathbf{e}_-)$ $\dfrac{((\mathbf{e}_+\cdot\underline{\boldsymbol{X}})\wedge\mathbf{e}_-)\cdot\mathbf{e}_-}{(\mathbf{e}_+\cdot\underline{\boldsymbol{X}})\cdot\mathbf{e}_-}$	$\underset{\longleftarrow}{\longrightarrow}$	Θ_c

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

To estimate a rigid body motion of an entity given in projective geometry, we change its representation in a conformal one, compute the rigid body motion and go back to the projective space: Let Θ_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, $\boldsymbol{T}=\left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)$ and $\widetilde{\boldsymbol{T}}=\left(1-\frac{\mathbf{e}\boldsymbol{t}}{2}\right)$. Then a multiplicative formulation of the translated

entity in the projective space is given by

$$\Theta'_p = \underbrace{\underbrace{\mathbf{e}_+ \cdot (\boldsymbol{T}(\mathbf{e} \wedge \underbrace{\Theta_p}_{\text{projective}})\tilde{\boldsymbol{T}})}_{\text{conformal}}}_{\text{projective}} \tag{4.10}$$

To compute joins and meets of entities, given in conformal geometric algebra, we change their representations to the projective space, perform the incidence operation and go back to the conformal space. As an 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

$$\underline{\boldsymbol{L}}^\star \vee_c \underline{\boldsymbol{P}}^\star = \underbrace{\mathbf{e} \wedge \underbrace{((\mathbf{e}_+ \cdot \underbrace{\underline{\boldsymbol{L}}^\star}_{\text{conformal}}) \vee (\mathbf{e}_+ \cdot \underbrace{\underline{\boldsymbol{P}}^\star}_{\text{conformal}}))}_{\text{projective}}}_{\text{conformal}} \tag{4.11}$$

$$\tag{4.12}$$

To explicitly compute the coordinates of the intersection point of two lines, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, given in the projective space, we intersect these lines in the projective space and use the projective split to get the intersection point in the geometric algebra of the Euclidean space,

$$\boldsymbol{x} = \underbrace{\frac{1}{\underbrace{\boldsymbol{L}_1 \vee \boldsymbol{L}_2}_{projective} \cdot \mathbf{e}_-} \left(((\underbrace{\boldsymbol{L}_1 \vee \boldsymbol{L}_2}_{projective}) \wedge \mathbf{e}_-) \cdot \mathbf{e}_- \right)}_{Euclidean} \tag{4.13}$$

These examples show, how to interact between the Euclidean, projective and conformal framework.

4.2.1 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. Our constraint equations will lead to equations of the following structure (here just for point correspondences),

$$\lambda((\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) \cdot \mathbf{e}_+ = 0. \tag{4.14}$$

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

$$\lambda((\boldsymbol{M} \underbrace{\underline{\boldsymbol{X}}}_{\substack{\text{object point}\\ \text{in conformal space}}} \widetilde{\boldsymbol{M}}) \underline{\times} \underbrace{\mathbf{e} \wedge (\underbrace{\boldsymbol{O}}_{\substack{\text{optical}\\ \text{center}}} \wedge \underbrace{\boldsymbol{x}}_{\substack{\text{image}\\ \text{point}}})}_{\substack{\text{projection ray,}\\ \text{reconstructed from the image point}\\ \text{in conformal space}}}) \cdot \mathbf{e}_+ = 0. \tag{4.15}$$

$$\underbrace{\underbrace{\underbrace{\lambda((\boldsymbol{M} \underbrace{\underline{\boldsymbol{X}}}_{CS} \widetilde{\boldsymbol{M}})}_{CS} \underline{\times} \underbrace{\mathbf{e} \wedge (\underbrace{\underbrace{\boldsymbol{O}}_{PS} \wedge \underbrace{\boldsymbol{x}}_{PP}}_{PS})}_{CS}) \cdot \mathbf{e}_+ = 0.}_{ES}}_{\substack{\text{geometric distance measure between 3D line and 3D point}}}}_{\substack{\text{collinearity of the transformed object}\\ \text{point with the reconstructed line}}}}_{\substack{\text{rigid motion of the object point}}} \tag{4.16}$$

Here does PP abbreviate *projective plane*, PS *projective space*, CS *conformal space* and ES the *Euclidean space*. Furthermore, will part II show that the used commutator and anti-commutator products can be used to describe a geometric distance measure, to ensure good conditioned equations in the presence of noise. This will become more clear in part II.

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, etc.

5. SUMMARY AND DISCUSSION

This work is concerned with the theoretical foundations of the 2D-3D pose estimation problem. Firstly, Faugeras' stratification hierarchy is identified as an important concept in the pose estimation problem. But since it is based solely on point concepts we introduce the conformal geometric algebra which provides a homogeneous model for stereographic geometry and is therefore well suited to deal with projective geometry on the one hand and kinematics on the other hand. The multivector concepts of geometric algebras lead to a new stratification hierarchy which contains as highest algebra the conformal geometric algebra. Since conformal geometry is not well known for solving computer vision problems, we introduce the stereographic projections and sphere concepts in some detail in the context of the pose problem.

The usefulness of this approach is as preview shown in section 4.2.1: we gain compact constraint equations with a strict modeling of all geometric aspects. We will apply this in part II for simultaneous pose estimation of different image and object features, e.g. containing points, lines, planes, circles, spheres or kinematic chains. Our very recent work [31] concerns further extensions of this approach e.g. by modeling cycloidal curves and free-form contours.

Acknowledgments

We would like to thank Oliver Granert, Daniel Grest, Norbert Krüger and Christian Perwass for fruitful discussions and hints for performing this work. This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).

BIBLIOGRAPHY

[1] Araujo H., Carceroni R.L. and Brown C.M. A fully projective formulation to improve the accuracy of Lowe's pose-estimation algorithm in *Computer Vision and Image Understanding CVIU*, Vol.70, pp. 227-238, 1998

[2] 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* Vol. 13, pp. 79-100, 2000.

[3] Bayro-Corrochano E. The geometry and algebra of kinematics. In [34], pp. 457-472, 2001.

[4] Beveridge J.R. Local search algorithms for geometric object recognition: Optimal correspondence and pose. *Technical Report CS 93–5, University of Massachusetts*, 1993.

[5] Blaschke W. Kinematik und Quaternionen, Mathematische Monographien 4. *Deutscher Verlag der Wissenschaften*, 1960.

[6] Bregler C. and Malik J. Tracking people with twists and exponential maps. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Santa Barbara, California, pp.8-15 1998.

[7] 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.

[8] CLU Library, A C++ Library for Clifford Algebra. available at http://www.perwass.de/CLU *Cognitive Systems Group, University Kiel,* 2001.

[9] 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.

[10] Gallier J. Geometric Methods and Applications for Computer Science and Engineering. *Springer Verlag*, New York, 2001.

[11] Gilbert J.E. and Murray M.A.M. Clifford Algebras and Dirac Operators in Harmonic Analysis. *Cambridge University Press*, 1991.

[12] Grimson W. E. L. Object Recognition by Computer. *The MIT Press, Cambridge, MA*, 1990.

[13] Hestenes D. Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, Vol. 7, pp.65–77, 1994.

[14] Hestenes D. and Sobczyk G. Clifford Algebra to Geometric Calculus. *D. Reidel Publ. Comp., Dordrecht*, 1984.

[15] Hestenes D. The design of linear algebra and geometry *Acta Applicandae Mathematicae*, Vol. 23, pp.65–93, 1991.

[16] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [34], pp. 3-23, 2001.

[17] Hestenes D. and Ziegler R. Projective geometrie with Clifford algebra. *Acta Applicandae Mathematicae*, Vol. 23, pp.25–63, 1991.

[18] 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.7:2, pp. 126–136, 1996.

[19] Homer H.H. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13:6, pp.530–541, 1991.

[20] 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*, Vol. 15, pp. 225–243, 1995.

[21] Klingspohr H., Block T. and Grigat R.-R. A passive real-time gaze estimation system for human-machine interfaces. *Computer Analysis of Images and Patterns (CAIP), Sommer G., Daniilidis K. and Pauli J. (Eds.)*, LNCS 1296, Springer-Verlag Heidelberg, pp. 718-725, 1997.

[22] Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [34], pp. 27-52, 2001.

[23] Murray R.M., Li Z. and Sastry S.S. A Mathematical Introduction to Robotic Manipulation. *CRC Press*, 1994.

[24] Needham T. Visual Complex Analysis. *Oxford University Press*, 1997

[25] Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T. Numerical Recipes in C. Cambridge University Press, 1993.

[26] Perwass C. and Lasenby J. A novel axiomatic derivation of geometric algebra. Technical Report CUED/F - INFENG/TR.347, Cambridge University Engineering Department, 1999.

[27] Rooney J. A comparison of representations of general screw displacement. *Environment and Planning* , Vol. B5, pp. 45-88, 1978.

[28] 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, Sommer G. and Zeevi Y.Y. (Eds.)*, LNCS 1888, Springer-Verlag Heidelberg, pp.284- 293, 2000.

[29] 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.

[30] Rosenhahn B., Granert O., Sommer G. Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering, Birkhäuser Verlag, Dorst L., Doran C. and Lasenby J. (Eds.)*, pp. 373-383, 2001.

[31] 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.

[32] 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. (Ed.)*, LNCS 1843, Springer-Verlag Heidelberg, pp.50-66, 2000.

[33] Shevlin F. Analysis of orientation problems using Plücker lines. *International Conference on Pattern Recognition, Brisbane*, Vol. 1, pp.685–689, 1998.

[34] Sommer G., editor. Geometric Computing with Clifford Algebra. *Springer Verlag Heidelberg*, 2001.

[35] Sommer G., Rosenhahn B., and Zhang Y. Pose estimation using geometric constraints. In: *Multi-Image Search and Analysis, R.Klette, Th.*

Huang, G.Gimmel'farb (Eds.), LNCS 2032, Springer-Verlag, Heidelberg, pp. 153–170, 2001.

[36] 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.

[37] Walker M.W. and Shao L. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, Vol. 54:3, pp.358–367, 1991.

[38] Yaglom I.M. Felix Klein and Sophus Lie. *Birkhäuser*, Boston, 1988

Pose Estimation in Conformal Geometric Algebra

Part II: Real-time Pose Estimation using Extended Feature Concepts

Bodo Rosenhahn and Gerald Sommer
Cognitive Systems Group
Institute of Computer Science and Applied Mathematics
Christian-Albrechts-University of Kiel
D-24105 Kiel, Germany
{bro,gs}@ks.informatik.uni-kiel.de

Abstract

Part II uses the foundations of part I to define constraint equations for 2D-3D pose estimation of different corresponding entities. Most articles on pose estimation concentrate on specific types of correspondences, mostly between point, and only rarely line correspondences. The first aim of this part is to extend pose estimation scenarios to correspondences of an extended set of geometric entities. In this context we are interested to relate the following (2D) image and (3D) model types: 2D point/3D point, 2D line/3D point, 2D line/3D line, 2D conic/3D circle, 2D circle/3D sphere. Furthermore, to handle articulated objects, we describe kinematic chains in this context in a similar manner. We ensure that all constraint equations end up in a distance measure in the Euclidean space, which is well posed in the context of noisy data. We also discuss the numerical estimation of the pose. We propose to use linearized twist transformations which result in well conditioned and fast solvable systems of equations. The key idea is not to search for the representation of the Lie group, describing the rigid body motion, but for the representation of their generating Lie algebra. This leads to real-time capable algorithms.

Keywords : 2D-3D pose estimation, pose constraints, kinematic chains, circles, spheres, twists.

CONTENTS

1. INTRODUCTION

This contribution concerns the 2D-3D pose estimation problem of 3D free-form curves. Pose estimation itself is one of the oldest computer vision problems and algebraic solutions with different camera models have been proposed for several variations of this problem. Pioneering work was done in the 80's and 90's by Lowe [20, 21], Grimson [11] and others. In their work point correspondences are used. Other works concerning lines or line segments can be found in e.g. [39, 18]. Works concerning extensions to kinematic chains can be found in [4, 12]. Nearly all papers concentrate on one specific type of correspondences. But many situations are conceivable in which a system has to gather information from different hints or has to consider different reliabilities of measurements. This is the main aspect of this work: We describe a scenario for adaptive pose estimation of simultaneously used different entities, without loosing linearity, well conditioned equations and real-time capability. This work contains the second part of our research on pose estimation. The first part discussed the scenario and the mathematical foundations for the pose problem. This part uses these foundations to deal with the pose estimation problem:

Section 2 starts with the pose constraints to relate 3D point, line and plane features. In section 3 we extend this formalization to kinematic chains, 3D circles and 3D spheres. The aim is to model all different kinds of entities, their transformations and relations in one algebraic framework to get constraint equations which can be used in a similar manner and can be solved simultaneously. Furthermore, we explain how to use the constraints in a noise adaptive way. This means, we control the influence of a constraint to the whole system of equations. This is only possible if the constraints describe (in their implicit formulations) equivalent geometries. In this context every constraint results in a Hesse distance error measure between the involved entities. In the experimental part, section 4, we present several experiments which visualize the properties of our algorithms.

2. COLLINEARITY AND COPLANARITY CONSTRAINTS IN CONFORMAL GEOMETRIC ALGEBRA

So far we have introduced (see part I) the entities, their transformations and the interaction of entities between Euclidean, projective and conformal geometry. The aim is now to formalize the pose estimation problem in an implicit way using a set of geometric constraints which describe an error measure to be minimized.

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

$$\begin{aligned} \mathbf{e} \wedge \boldsymbol{X} &= \underline{\boldsymbol{X}}^{\star} \\ \mathbf{e} \wedge \boldsymbol{L} &= \underline{\boldsymbol{L}}^{\star} \\ \mathbf{e} \wedge \boldsymbol{P} &= \underline{\boldsymbol{P}}^{\star}. \end{aligned} \tag{2.1}$$

In the next sections we will only work in the dual representation of the entities and therefore, from now on we will neglect the $\star$-sign in the equations.

In this section, we will derive constraints for collinearity and coplanarity to relate points, lines and planes. The constraints will be given in the conformal space. They are then translated in an error measure of the Euclidean space. While this section only concerns the relation of points, lines and planes, the following sections will regard the constraints to relate the other entities as circles and spheres.

Table 2.1 gives an overview of the formulations of the constraints for collinearity and coplanarity of points, lines and planes in conformal geometric algebra, which were first presented in [36, 31]. Now we will analyze the geometry of the constraints introduced in table 2.1.

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

2.1 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

$$
\begin{aligned}
0 &= \underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{L}} \\
&= \frac{1}{2}\left(\underline{\boldsymbol{X}\boldsymbol{L}} - \underline{\boldsymbol{L}\boldsymbol{X}}\right) \\
&= \frac{1}{2}\left((\boldsymbol{E} + \mathbf{e}\boldsymbol{x})(\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m}) - (\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})(\boldsymbol{E} + \mathbf{e}\boldsymbol{x})\right) \\
&= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{E}\boldsymbol{r} + \boldsymbol{E}\mathbf{e}\boldsymbol{m} + \boldsymbol{E}^2\boldsymbol{r} - \mathbf{e}\boldsymbol{m}\boldsymbol{E} - \boldsymbol{r}\boldsymbol{E}\mathbf{e}\boldsymbol{x} - \boldsymbol{r}\right) \\
&= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{r} - \boldsymbol{m}\mathbf{e} - \boldsymbol{m}\mathbf{e} - \boldsymbol{r}\boldsymbol{x}\mathbf{e}\right) \\
&= \frac{1}{2}\left(-(2\boldsymbol{m} - (\boldsymbol{x}\boldsymbol{r} - \boldsymbol{r}\boldsymbol{x}))\mathbf{e}\right) \\
&= -(\boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r})\mathbf{e} \\
\Leftrightarrow 0 &= (\boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r})\mathbf{e} \cdot \mathbf{e}_+ = \boldsymbol{m} - \boldsymbol{x}\underline{\times}\boldsymbol{r}
\end{aligned}
\tag{2.2}
$$

The product $\underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{L}}$ is an element in the null space, since $\mathbf{e}^2 = 0$. By estimating the inner product with $\mathbf{e}_+$, we can change this expression to an equation in the non-null space. Note, that this is consistent with table 4.2 of part I. Estimating 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 [3].

So far the constraint equation is given unscaled. Following part I, we have to apply a scaling parameter $\lambda \in \mathbb{R}$ to express a distance measure in the Euclidean space. In this case let $\lambda = \frac{1}{\|\boldsymbol{r}\|}$. That means we scale the

equation with the inverse norm of the direction of the line. Then we get

$$\begin{aligned} 0 &= \boldsymbol{m} - \boldsymbol{x} \underline{\times} \boldsymbol{r} \\ \Leftrightarrow 0 &= \lambda(\boldsymbol{m} - \boldsymbol{x} \underline{\times} \boldsymbol{r}) \\ \Leftrightarrow 0 &= \lambda\boldsymbol{m} - \boldsymbol{x} \underline{\times} (\lambda\boldsymbol{r}) \\ \Leftrightarrow 0 &= \boldsymbol{m}' - \boldsymbol{x} \underline{\times} \boldsymbol{r}'. \end{aligned} \tag{2.3}$$

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 2.1 shows the scenario.

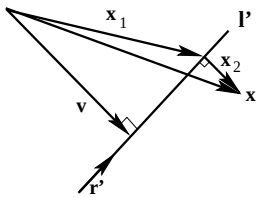

Fig. 2.1: The Euclidean line $\boldsymbol{l}'$ consists of the direction $\boldsymbol{r}'$ and the moment $\boldsymbol{m}' = \boldsymbol{v} \underline{\times} \boldsymbol{r}'$. Further, there exists a decomposition $\boldsymbol{x} = \boldsymbol{x_1} + \boldsymbol{x_2}$ with $\boldsymbol{x_1} \in \boldsymbol{l}'$ and $\boldsymbol{x_2} \perp \boldsymbol{l}'$ so that $\boldsymbol{m}' = \boldsymbol{v} \underline{\times} \boldsymbol{r}' = \boldsymbol{x}_1 \underline{\times} \boldsymbol{r}'$.

Then we can calculate

$$\begin{aligned} \|\boldsymbol{m}' - \boldsymbol{x} \underline{\times} \boldsymbol{r}'\| &= \|\boldsymbol{m}' - (\boldsymbol{x_1} + \boldsymbol{x_2}) \underline{\times} \boldsymbol{r}'\| \\ &= \|\boldsymbol{m}' - \boldsymbol{x_1} \underline{\times} \boldsymbol{r}' - \boldsymbol{x_2} \underline{\times} \boldsymbol{r}'\| \\ &= \|\boldsymbol{x_2} \underline{\times} \boldsymbol{r}'\| = \|\boldsymbol{x_2}\|. \end{aligned} \tag{2.4}$$

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.

2.2 Point-Plane Constraint

Evaluating the point-plane-constraint of a point $\underline{\boldsymbol{X}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{X}} = \boldsymbol{E} + \mathbf{e}\boldsymbol{x}$, coplanar to a plane $\underline{\boldsymbol{P}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{P}} = \boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E$, leads to

$$\begin{aligned} 0 &= \underline{\boldsymbol{X}} \underline{\times} \underline{\boldsymbol{P}} \\ &= \frac{1}{2}(\underline{\boldsymbol{X}\boldsymbol{P}} - \underline{\boldsymbol{P}\boldsymbol{X}}) \\ &= \frac{1}{2}((\boldsymbol{E} + \mathbf{e}\boldsymbol{x})(\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E) - (\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E)(\boldsymbol{E} + \mathbf{e}\boldsymbol{x})) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\left(\mathbf{e}\boldsymbol{x}\boldsymbol{E}\boldsymbol{n} + \boldsymbol{E}\mathbf{e}d\boldsymbol{I}_E + \boldsymbol{n} - \mathbf{e}d\boldsymbol{I}_E\boldsymbol{E} - \boldsymbol{E}\boldsymbol{n}\mathbf{e}\boldsymbol{x} - \boldsymbol{n}\right)\\
&= \frac{1}{2}\left(-\boldsymbol{x}\boldsymbol{n}\mathbf{e} + d\boldsymbol{I}_E\mathbf{e} + \boldsymbol{n} + d\boldsymbol{I}_E\mathbf{e} - \boldsymbol{n}\boldsymbol{x}\mathbf{e} - \boldsymbol{n}\right)\\
&= \frac{1}{2}\left(2d\boldsymbol{I}_E - (\boldsymbol{x}\boldsymbol{n} + \boldsymbol{n}\boldsymbol{x})\right)\mathbf{e}\\
&= (d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n})\mathbf{e}\\
\Leftrightarrow 0 &= (d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n}))\mathbf{e}\cdot\mathbf{e}_+ = d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n}). \qquad (2.5)
\end{aligned}
$$

Note here that the anticommutator product of the bivector $\boldsymbol{n}$ and the vector $\boldsymbol{x}$ results in 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 estimating 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. Then we get

$$
\begin{aligned}
0 &= d\boldsymbol{I}_E - (\boldsymbol{x}\overline{\times}\boldsymbol{n})\\
\Leftrightarrow 0 &= \lambda(d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n})\\
\Leftrightarrow 0 &= \lambda d\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}(\lambda\boldsymbol{n})\\
\Leftrightarrow 0 &= d'\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}' \qquad (2.6)
\end{aligned}
$$

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 2.2 shows the scenario. Then we can calculate

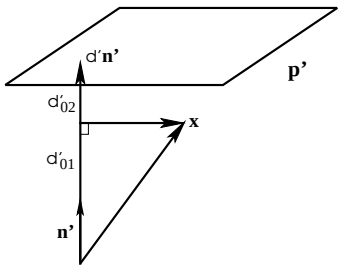

Fig. 2.2: The Euclidean plane $\boldsymbol{p}'$ is represented by the normal $\boldsymbol{n}'$ and the Hesse distance d'. The value d' can be interpreted as a sum $d' = d'_{01} + d'_{02}$ so that $d'_{01}\boldsymbol{n}'$ corresponds to the orthogonal projection of $\boldsymbol{x}$ onto $\boldsymbol{n}'$.

$$
\begin{aligned}
d'\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}' &= (d'_{01} + d'_{02})\boldsymbol{I}_E - \boldsymbol{x}\overline{\times}\boldsymbol{n}'\\
&= d'_{02}\boldsymbol{I}_E. \qquad (2.7)
\end{aligned}
$$

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 $\boldsymbol{p}'$.

2.3 Line-Plane Constraint

Evaluating the line-plane-constraint of a line $\underline{\boldsymbol{L}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{L}} = \boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m}$, coplanar to a plane $\underline{\boldsymbol{P}} \in \boldsymbol{\mathcal{G}}_{4,1}$, $\underline{\boldsymbol{P}} = \boldsymbol{E}\boldsymbol{n} + \mathbf{e}\boldsymbol{I}_E d$, leads to

$$\begin{aligned}
0 &= \underline{\boldsymbol{L}}\,\overline{\times}\,\underline{\boldsymbol{P}} \\
&= \frac{1}{2}(\underline{\boldsymbol{L}\boldsymbol{P}} + \underline{\boldsymbol{P}\boldsymbol{L}}) \\
&= \frac{1}{2}((\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})(\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E) + (\boldsymbol{E}\boldsymbol{n} + \mathbf{e}d\boldsymbol{I}_E)(\boldsymbol{E}\boldsymbol{r} + \mathbf{e}\boldsymbol{m})) \\
&= \frac{1}{2}(\mathbf{e}\boldsymbol{m}\boldsymbol{E}\boldsymbol{n} + \boldsymbol{r}\boldsymbol{E}\mathbf{e}\boldsymbol{I}_E d + \boldsymbol{r}\boldsymbol{n} + \mathbf{e}\boldsymbol{I}_E d\boldsymbol{r}\boldsymbol{E} + \boldsymbol{E}\boldsymbol{n}\mathbf{e}\boldsymbol{m} + \boldsymbol{E}\boldsymbol{n}\boldsymbol{r}\boldsymbol{E}) \\
&= \frac{1}{2}(\boldsymbol{m}\boldsymbol{n}\mathbf{e} + \boldsymbol{r}\boldsymbol{I}_E d\mathbf{e} + \boldsymbol{r}\boldsymbol{n} + \boldsymbol{I}_E d\boldsymbol{r}\mathbf{e} - \boldsymbol{n}\boldsymbol{m}\mathbf{e} + \boldsymbol{n}\boldsymbol{r}) \\
&= \frac{1}{2}((\boldsymbol{r}\boldsymbol{n} + \boldsymbol{n}\boldsymbol{r}) + (2\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\boldsymbol{n} - \boldsymbol{n}\boldsymbol{m})\mathbf{e}) \\
&= \frac{1}{2}((\boldsymbol{r}\boldsymbol{n} + \boldsymbol{n}\boldsymbol{r}) + 2(\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e}) \\
&= \boldsymbol{r}\overline{\times}\boldsymbol{n} + (\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e} \qquad (2.8)
\end{aligned}$$

Thus, the constraint of 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 seen in equation (2.8) since $\mathbf{e}^2 = 0$.

Again the constraint equation is given unscaled. Let be $\lambda = \frac{1}{\|\boldsymbol{n}\|\|\boldsymbol{r}\|}$. Then we get

$$\begin{aligned}
0 &= \boldsymbol{r}\overline{\times}\boldsymbol{n} + (\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e} \\
\Leftrightarrow 0 &= \lambda(\boldsymbol{r}\overline{\times}\boldsymbol{n} + (\boldsymbol{r}\boldsymbol{I}_E d + \boldsymbol{m}\underline{\times}\boldsymbol{n})\mathbf{e}) \\
\Leftrightarrow 0 &= \boldsymbol{r}'\overline{\times}\boldsymbol{n}' + (\boldsymbol{r}'\boldsymbol{I}_E d' + \boldsymbol{m}'\underline{\times}\boldsymbol{n}')\mathbf{e}, \qquad (2.9)
\end{aligned}$$

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 $\boldsymbol{r}' \not\perp \boldsymbol{n}'$, the non-null part leads to

$$\boldsymbol{r}'\overline{\times}\boldsymbol{n}' = \|\boldsymbol{r}'\|\|\boldsymbol{n}'\|\cos(\alpha) = \cos(\alpha), \qquad (2.10)$$

where α is the angle between $\underline{\boldsymbol{L}}'$ and $\underline{\boldsymbol{P}}'$, see figure 2.3. If $\boldsymbol{r}' \perp \boldsymbol{n}'$, we have $\boldsymbol{r}'\overline{\times}\boldsymbol{n}' = 0$. 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 to be estimated, $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}\tilde{\boldsymbol{T}} = \boldsymbol{R}'_1 + \mathbf{e}\boldsymbol{R}'_2$, 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 of the unknown rotation from equations of the unknown translation without the limitations known from the embedding of the problem in Euclidean space [6]. This is useful since the two smaller systems of equations are faster to solve than one larger system of equations. To analyze the null part of the constraint we interpret the moment $\boldsymbol{m}'$ of the line representation $\underline{\boldsymbol{L}}' = \boldsymbol{E}\boldsymbol{r}' + \mathbf{e}\boldsymbol{m}'$ 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 [28], we can estimate

$$\begin{aligned} \boldsymbol{n}'\underline{\times}\boldsymbol{m}' &= -(\boldsymbol{s}\underline{\times}\boldsymbol{r}')\underline{\times}\boldsymbol{n}' \\ &= (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}' - \boldsymbol{s}\overline{\times}(\boldsymbol{r}'\overline{\times}\boldsymbol{n}'). \end{aligned} \tag{2.11}$$

Now we can evaluate

$$d\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}') = d\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}' + \boldsymbol{s}\overline{\times}(\boldsymbol{r}'\overline{\times}\boldsymbol{n}'). \tag{2.12}$$

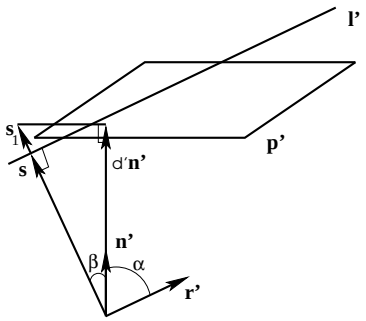

Fig. 2.3: The Euclidean plane $\boldsymbol{p}'$ is represented by its normal $\boldsymbol{n}'$ and the Hesse distance d'. Furthermore, we choose $\boldsymbol{s} \in \boldsymbol{l}'$ with $\boldsymbol{s} \perp \boldsymbol{r}'$. The angle of $\boldsymbol{r}'$ and $\boldsymbol{n}'$ is α and the angle of $\boldsymbol{s}$ and $\boldsymbol{n}'$ is β. We choose the vector $\boldsymbol{s_1}$ with $\boldsymbol{s} \parallel \boldsymbol{s_1}$ so that $d\boldsymbol{n}'$ is the orthogonal projection of $(\boldsymbol{s} + \boldsymbol{s_1})$ onto $\boldsymbol{n}'$.

Figure 2.3 shows the scenario. Further, we can 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 we can evaluate

$$\begin{aligned} d'\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}') &= d'\boldsymbol{I}_E\boldsymbol{r}' - \|\boldsymbol{s}\|\cos(\beta)\boldsymbol{r}' + \cos(\alpha)\boldsymbol{s} \\ &= \|\boldsymbol{s}_1\|\cos(\beta)\boldsymbol{r}' + \cos(\alpha)\boldsymbol{s}. \end{aligned} \tag{2.13}$$

Thus, the error of the null part of the motor is constituted by the sum of the vector $\boldsymbol{s}$, scaled by the angle α, and the direction vector $\boldsymbol{r}'$, scaled by the norm of $\boldsymbol{s_1}$ and the angle β.
If $\boldsymbol{r}' \perp \boldsymbol{n}'$, then $\boldsymbol{n}' \parallel \boldsymbol{s}$ and, thus, we will find

$$\begin{aligned} \|d'\boldsymbol{I}_E\boldsymbol{r}' - (\boldsymbol{n}'\underline{\times}\boldsymbol{m}')\| &= \|d'\boldsymbol{I}_E\boldsymbol{r}' + \boldsymbol{s}\overline{\times}(\boldsymbol{r}'\overline{\times}\boldsymbol{n}') - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}'\| \\ &= \|d'\boldsymbol{I}_E\overline{\times}\boldsymbol{r}' - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\overline{\times}\boldsymbol{r}'\| \\ &= \|d'\boldsymbol{I}_E - (\boldsymbol{s}\overline{\times}\boldsymbol{n}')\|. \end{aligned} \tag{2.14}$$

This means, in agreement with the point-plane constraint, that the above difference measure corresponds the Hesse distance of the line to the plane.

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

2.4 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 *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 $\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$ and $\underline{\boldsymbol{L}}' = \boldsymbol{M}\underline{\boldsymbol{L}}\widetilde{\boldsymbol{M}}$. 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 we 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 confound in the scenario and it avoids extra fonds. The projective reconstruction of these entities can be written as $\boldsymbol{L}_x = \boldsymbol{O} \wedge \boldsymbol{x} \in \boldsymbol{\mathcal{G}}_{3,1}$ and $\boldsymbol{P}_l = \boldsymbol{O} \wedge \boldsymbol{l} \in \boldsymbol{\mathcal{G}}_{3,1}$. The point $\boldsymbol{O} \in \boldsymbol{\mathcal{G}}_{3,1}$ denotes the optical center of the camera. Then we can apply the $\mathbf{e}\wedge$-operator to change the representations from the projective to the conformal space, and combine it with the commutator and anticommutator products to express the collinearity and coplanarity of the involved entities.

Thus, the constraint equations of pose estimation read
point-line constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x})}_{\substack{\text{projection ray,}\\ \text{reconstructed from the image point}}}}_{\substack{\text{collinearity of the transformed object}\\ \text{point with the reconstructed line}}}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (2.15)$$

point-plane constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{l})}_{\substack{\text{3D plane,}\\ \text{reconstructed from the image line}}}}_{\substack{\text{coplanarity of the transformed object}\\ \text{point with the reconstructed plane}}}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (2.16)$$

line-plane constraint:

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{L}}}_{\text{object line}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object libe}}\ \overline{\times}\ \underbrace{\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{l})}_{\substack{\text{3D plane,}\\ \text{reconstructed from the image line}}}}_{\substack{\text{coplanarity of the transformed object}\\ \text{line with the reconstructed plane}}}\Big) \;=\; 0. \quad (2.17)$$

Note, that there is no $\cdot\mathbf{e}_+$ operation involved in the line-plane constraint. The reason is, that the constraint equation is partitioned into one equation on the non-null part and one equation on the null part of the constraint equation as explained in the last section.

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

$$\lambda\Big(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{CS}\ \widetilde{\boldsymbol{M}})}_{CS}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\underbrace{\underline{\boldsymbol{O}}}_{PS}\wedge\underbrace{\boldsymbol{x}}_{PP})}_{CS}}_{ES}\Big)\cdot\mathbf{e}_+ \;=\; 0. \quad (2.18)$$

Here does PP abbreviate *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 described in an implicit way. Note, that the stratification hierarchy of the involved entities is strictly kept within these equations. Furthermore are the equations compact and

therefore easy to interpret. Additionally do the geometric analysis of the constraints assure well conditioned equations and help to interpret effects of the constraints discussed in the experimental part. The constraints behave robust in case of noisy data and linearization and iteration enables 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 [5, 38], in our approach a distance in the Euclidean space constitutes the error measure.

This is the now the complete formulation and analysis of the constraint equations already shown in part I, chapter 4.

3. POSE ESTIMATION WITH EXTENDED OBJECT CONCEPTS

This section concerns the development of constraint equations to relate 3D kinematic chains, circles and spheres with corresponding extracted 2D image features. similar to the previous section we will formalize constraint equations in the 3D space, which contain a geometric distance measure.

3.1 Pose Estimation of Kinematic Chains

So far we have parameterized the 3D pose constraint equations of a rigid object. Let be given a rigid object by a set of entities as points and lines. Assume that a second rigid object 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 θ_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 [7]. Each joint defines a new coordinate system, and the coordinate transformations between joints can be expressed by suitable motors $\boldsymbol{M}_j$. This means, an entity given in the coordinate system of the jth joint can be translated in an entity of the base coordinate system by transforming it with the motors $\boldsymbol{M}_0, \ldots, \boldsymbol{M}_j$.

Such objects are also called *kinematic chains*. With kinematic chains we mean flexible linked rigid objects which can only change their pose in mutual dependence. Examples are robot arms [37] or human body movements, see e.g. figure 3.1. 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 an other base coordinate system, all involved joint coordinate systems must traced. This is visualized in figure 3.1. For short notations of the single transformations in CGA between the joints we define

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

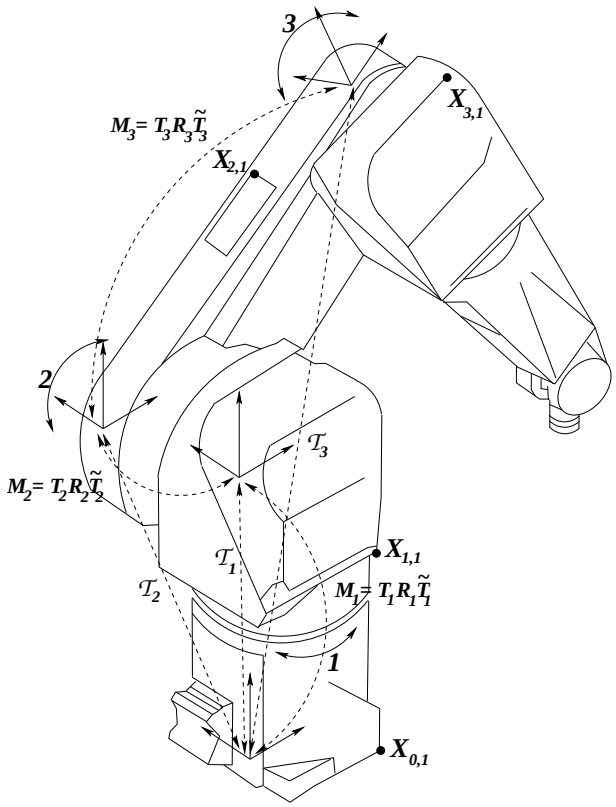

Fig. 3.1: The RX-90 robot arm. The internal joint transformations $\boldsymbol{M}_j$ and the global transformation $\mathcal{T}_j$ are visualized.

$$= \quad \boldsymbol{M}_1 \ldots \boldsymbol{M}_j \underline{\boldsymbol{X}}_{j,i_j} \widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1 : j = 1, \ldots, n. \quad (3.1)$$

The function $\mathcal{T}_0$ with the motor $\boldsymbol{M}_0$ describes the identity for points which are not subject to internal transformations. We call them *base* points. Since the motor $\boldsymbol{M}_0$ is just the identity it will be neglected in the further equations. The function $\mathcal{T}_j$ formalizes the transformation of an attached joint j with respect to the basis coordinate system in an inductive manner. In the general case, the transformation of a point $\underline{\boldsymbol{X}}_{j,i_j}$ of a j-th joint to the base coordinate system is represented by a sequence of such 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 $n+1$ such functions $\mathcal{T}_j$,

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

3.2 Constraint Equations of Kinematic Chains

Now we will combine the introduced representation of kinematic chains in CGA with the pose estimation constraints derived in section 2. This is very simple now because everything is formulated in the same algebra. Note, that the constraints are presented unscaled, so the $\lambda(.) \cdot \mathbf{e}_+$ operation is not extra written.

The general unknown pose corresponds to a motor $\boldsymbol{M}$. For the base points $\underline{\boldsymbol{X}}_{0,i_0}$ the constraint equations reduce for a suitable projection ray $\underline{\boldsymbol{L}}_{0,i_0} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0})$ to

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{X}}_{0,i_0}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{0,i_0}) &= 0. \end{aligned} \tag{3.2}$$

The general constraint equation for a point $\underline{\boldsymbol{X}}_{j,i_j}$ at the j-th joint leads to

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{X}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j\underline{\boldsymbol{X}}_{j,i_j}\widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{j,i_j}) &= 0. \end{aligned} \tag{3.3}$$

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

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{X}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{X}}_{0,i_0}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0 \end{aligned} \tag{3.4}$$

for the base points, and the general constraint equation for a point at the jth joint leads to

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{X}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j\underline{\boldsymbol{X}}_{j,i_j}\widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0. \end{aligned} \tag{3.5}$$

We can also describe kinematic chains by lines and combine them with the line-plane-constraint. For this, only 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 anticommutator product. For the base lines we get

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_0\{\underline{\boldsymbol{L}}_{0,i_0}\})\widetilde{\boldsymbol{M}}) \overline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}\underline{\boldsymbol{L}}_{0,i_0}\widetilde{\boldsymbol{M}}) \overline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{0,i_0}) &= 0, \end{aligned} \tag{3.6}$$

and for a line on the j-th joint we get

$$\begin{aligned} (\boldsymbol{M}(\mathcal{T}_j\{\underline{\boldsymbol{L}}_{j,i_j}, \boldsymbol{M}_j\})\widetilde{\boldsymbol{M}}) \mathbin{\overline{\times}} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0 \\ \Leftrightarrow (\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j \underline{\boldsymbol{L}}_{j,i_j} \widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \mathbin{\overline{\times}} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{l}_{j,i_j}) &= 0. \quad (3.7) \end{aligned}$$

A detailed description of kinematic chains in CGA and the construction of pose estimation constraints can be found in [29].

3.3 Pose Estimation using Constraints for Circles and Spheres

In this section constraint equations are derived to relate 3D circles to 2D conics and 3D spheres to 2D circles. We start with an analysis of involved problems and then present a suitable solution approach.

3.3.1 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 conic or an image circle (the image of a circle or sphere, respectively) leads to a cone. Indeed, we can not formalize cones as single entities in conformal geometric algebra. But to enable the above mentioned comparison, we formalize constraint equations for tangentiality of 3D circles or spheres to projection rays, reconstructed from image points of the corresponding image entity. We denote the spatial tangentiality of a 3D circle $\underline{\boldsymbol{z}}$ to a 3D line $\underline{\boldsymbol{L}}$ as circle-line and the tangentiality of a 3D sphere $\underline{\boldsymbol{s}}$ to a 3D line $\underline{\boldsymbol{L}}$ as sphere-line constraint. Figure 3.2 visualizes the idea.

It is very 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 only holds 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 minima or no minimum in non-degenerate and degenerate cases. In figure 3.3 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

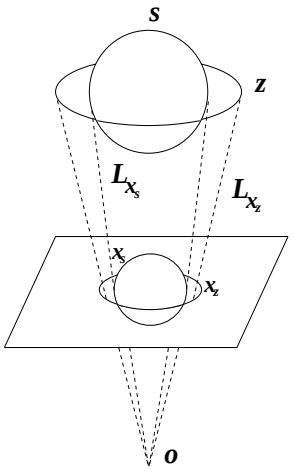

Fig. 3.2: Visualization of the circle-line and sphere-line constraint in conformal geometric algebra.

minima. The third line is passing the *inside* of the circle and is not parallel to the plane in which the circle lies. This results in one global and one local minimum. From that relations result two possible strategies: First, we make a case decision, depending on the geometric situation. This is hard to implement and to combine with our previous derived constraint equations. Second, we can parameterize the circle in a suitable way. This will be done in the following section.

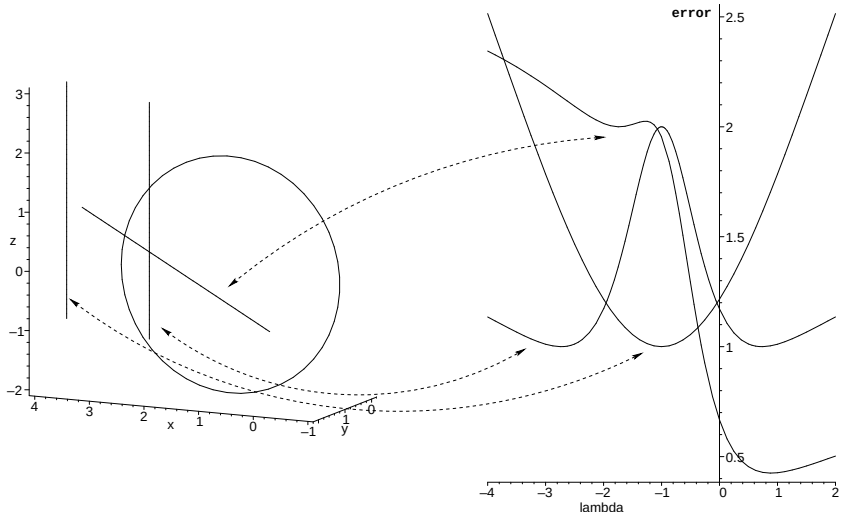

Fig. 3.3: Different geometric relations of lines to circles leads to different kinds of error functions for parameterized lines.

Comparing spheres with lines is in ideal geometry also no problem. 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 section 2. The line $\underline{\boldsymbol{L}}'$ is tangential to $\underline{\boldsymbol{s}} = \underline{\boldsymbol{p}} - \rho^2\mathbf{e}$ iff

$$\|(\underline{\boldsymbol{L}}' \underline{\times} \underline{\boldsymbol{p}}) \cdot \mathbf{e}_0\| - \rho \;=\; 0. \tag{3.8}$$

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, which are not nice to handle if we want to estimate the unknown motor $\boldsymbol{M}$ in the equation

$$\sqrt{((\boldsymbol{M}\underline{\boldsymbol{p}}\widetilde{\boldsymbol{M}} \underline{\times} \underline{\boldsymbol{L}}') \cdot \mathbf{e}_0)^2} - \rho \;=\; 0. \tag{3.9}$$

We made experiments with these kind of equations and implemented a Newton-Raphson method to solve the equations. But there are two main problems: First, the convergence rate is very slow and the algorithm often converges against the wrong minimum (the algorithm needs about 5 seconds to estimate the pose). Second, we loose the possibility to combine them 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.

3.3.2 Operational definition of circles and spheres using twists

We will first repeat the general description of circles and spheres, as introduced in part I and then generate an operational definition of these entities. In the next section we will continue with the formulations of the circle-line and sphere-line constraints.

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

$$\underline{\boldsymbol{z}} \;=\; \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} = A + A^{-}\mathbf{e} + A^{+}\mathbf{e}_0 + A^{\pm}\boldsymbol{E} \tag{3.10}$$

with suitable multivectors A, A^{-}, A^{+} and $A^{\pm}$, see part I.

A circle can also be understood as a twist $\underline{\boldsymbol{L}}_z$ and a point $\underline{\boldsymbol{X}}_z$ on the circle. From the dual representation of the circle, this information is very

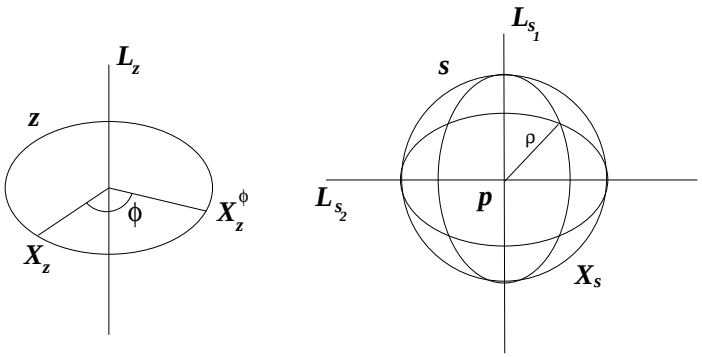

Fig. 3.4: Circles and spheres parameterized with twists.

easy to extract since the generating twist parameters are directly given. The twist transformation corresponds to a suitable parameterized motor $\boldsymbol{M}_\phi$,

$$\boldsymbol{M}_\phi \;=\; \exp\left(\frac{\phi}{2}\left(A^+ + \mathbf{e}A^\pm\right)\right). \tag{3.11}$$

The points on the circle are simply given by

$$\underline{\boldsymbol{X}}_z^\phi \;=\; (\boldsymbol{M}_\phi \underline{\boldsymbol{X}}_z \widetilde{\boldsymbol{M}}_\phi) \quad : \quad \phi \in [0, \ldots, 2\pi]. \tag{3.12}$$

Figure 3.4 (left) visualizes the geometry. The circle results as the orbit of the unique motor, which moves a certain point and is constraint by the points $\underline{\boldsymbol{a}}$, $\underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{c}}$ laying on the circle.

Now we continue with the formalization of a sphere. The general expression of a sphere leads to

$$\underline{\boldsymbol{s}} = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} \wedge \underline{\boldsymbol{d}} \;=\; (\underline{\boldsymbol{p}} - \frac{1}{2}\rho^2 \mathbf{e})\boldsymbol{I}_C^{-1}. \tag{3.13}$$

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

The idea is to formalize spheres in an operational manner as two coupled twists. 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 3.4 (right) visualizes the idea. The corresponding motors are denoted as $\boldsymbol{M}_{\phi_1}$ and $\boldsymbol{M}_{\phi_2}$,

$$\begin{aligned} \boldsymbol{M}_{\phi_1} &= \exp\left(\frac{\phi_1}{2}\left(\mathbf{e}_{12} + \mathbf{e}(\boldsymbol{p} \cdot \mathbf{e}_{12})\right)\right), \\ \boldsymbol{M}_{\phi_2} &= \exp\left(\frac{\phi_2}{2}\left(\mathbf{e}_{31} + \mathbf{e}(\boldsymbol{p} \cdot \mathbf{e}_{31})\right)\right). \end{aligned} \tag{3.14}$$

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

$$\underline{\boldsymbol{X}}_s^{\phi_1,\phi_2} \;=\; (\boldsymbol{M}_{\phi_1}\boldsymbol{M}_{\phi_2}\underline{\boldsymbol{X}}_s\widetilde{\boldsymbol{M}}_{\phi_2}\widetilde{\boldsymbol{M}}_{\phi_1}) \quad : \quad \phi_1,\phi_2 \in [0,\ldots,2\pi]. \tag{3.15}$$

This principle of coupling two motors is virtual in contrast to kinematic chains, which correspond the coupling of physical objects. The principle of virtual coupling can be further extended to construct more complex orbits of twists and, thus, to enable pose estimation of more complex objects, see [32].

3.3.3 The constraint equations of circles and spheres to lines

So far we have developed the formalization of circles and spheres as orbits of twists. We will use these representations to express incidence of circles $\underline{\boldsymbol{z}}$ and spheres $\underline{\boldsymbol{s}}$ to 3D lines $\underline{\boldsymbol{L}}$.

While in the constraint equations of section 2 the motors are the only unknowns to be estimated, now we have higher loads because of the parameterization of the features or entities of pose estimation.

We will start with the formalization of a suitable circle-line constraint. To relate the circle $\underline{\boldsymbol{z}}$ to a line $\underline{\boldsymbol{L}} = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})$, we only need to estimate the unknown angle ϕ, 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

$$(\boldsymbol{M}_\phi\underline{\boldsymbol{X}}_z\widetilde{\boldsymbol{M}}_\phi) \,\underline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;=\; 0. \tag{3.16}$$

In this equation, the angle ϕ 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 ϕ,

$$(\boldsymbol{M}(\boldsymbol{M}_\phi\underline{\boldsymbol{X}}_z\widetilde{\boldsymbol{M}}_\phi)\widetilde{\boldsymbol{M}}) \,\underline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;=\; 0. \tag{3.17}$$

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

$$(\boldsymbol{M}_{\phi_1}\boldsymbol{M}_{\phi_2}\underline{\boldsymbol{X}}_s\widetilde{\boldsymbol{M}}_{\phi_2}\widetilde{\boldsymbol{M}}_{\phi_1}) \,\underline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \;=\; 0. \tag{3.18}$$

In this constraint equation ϕ_1 and ϕ_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 ϕ_1 and ϕ_2,

$$(\boldsymbol{M}(\boldsymbol{M}_{\phi_1}\boldsymbol{M}_{\phi_2}\underline{\boldsymbol{X}}_s\widetilde{\boldsymbol{M}}_{\phi_2}\widetilde{\boldsymbol{M}}_{\phi_1})\widetilde{\boldsymbol{M}})\,\underline{\times}\,\mathbf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x}) \;=\; 0. \qquad (3.19)$$

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

4. REAL-TIME POSE ESTIMATION

This section concerns the numerical estimation of the pose parameters and presents experimental results.

4.1 Estimation of Motion Parameters

In the last sections, several constraint equations to relate object informations to image informations 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), this is a non-trivial task, especially in the case of real-time estimations.

4.1.1 Linearization in the tangential space

The idea is to gain linear equations with respect to the generators of the motor. We use the exponential representation of motors and apply the Taylor series expression of first order for approximation. This leads to a mapping of the above mentioned global motion transformation to a twist representation, which enables incremental changes of pose. That means, we do not search for the parameters of the Lie group $SE(3)$ to describe the rigid body motion [10], but for the parameters which generate their Lie algebra $se(3)$ [24]. This results in linear equations in the generators of the unknown 3D rigid body motion. In this section we derive the linearization of the motors. For the sake of simplicity we will do that in the case of point transformations.

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

$$\begin{aligned} \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))\right)\underline{\boldsymbol{X}}\exp\left(\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))\right) \\ &\approx (1-\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})))\underline{\boldsymbol{X}}(1+\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))) \end{aligned}$$

$$\approx \quad \boldsymbol{E} + \mathbf{e}(\boldsymbol{x} + \theta(\boldsymbol{l} \cdot \boldsymbol{x}) - \theta(\boldsymbol{t} \cdot \boldsymbol{l})). \tag{4.1}$$

Setting $\boldsymbol{v} := \theta(\boldsymbol{l} \cdot \boldsymbol{x})$ and $\boldsymbol{m} := \theta(\boldsymbol{t} \cdot \boldsymbol{l})$ leads to

$$\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \quad \approx \quad \boldsymbol{E} + \mathbf{e}(\boldsymbol{x} + \boldsymbol{v} - \boldsymbol{m}). \tag{4.2}$$

By combining this approximation of the motion with the previously derived constraints (e.g. the point-line constraint), we get

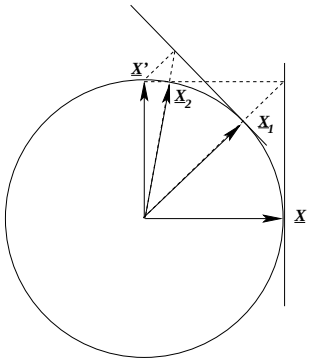

Fig. 4.1: Principle of the convergence rate for the iteration of a point $\underline{\boldsymbol{X}}$ rotated around 90 degree to a point $\underline{\boldsymbol{X}}'$. $\underline{\boldsymbol{X}}_1$ is the result of the first iteration and $\underline{\boldsymbol{X}}_2$ is the result of the second iteration.

$$\begin{aligned}
0 &= \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \times \underline{\boldsymbol{L}} \\
\Leftrightarrow 0 &= \exp\left(\frac{\theta}{2}(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l}))\right) \underline{\boldsymbol{X}} \exp\left(-\frac{\theta}{2}(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l}))\right) \times \underline{\boldsymbol{L}} \\
\Leftarrow\approx\Rightarrow 0 &= (\boldsymbol{E} + \mathbf{e}(\boldsymbol{x} + \boldsymbol{v} - \boldsymbol{m})) \times \underline{\boldsymbol{L}} \\
\Leftrightarrow 0 &= \lambda(\boldsymbol{E} + \mathbf{e}(\boldsymbol{x} + \boldsymbol{v} - \boldsymbol{m})) \times \underline{\boldsymbol{L}}.
\end{aligned} \tag{4.3}$$

Because of the approximation ($\Leftarrow\approx\Rightarrow$) the unknown motion parameters $\boldsymbol{v}$ 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 general rotation (five unknowns for the location of the twist and one unknown angle). In the last step we scale the linearized constraints with a suitable factor λ to express an Euclidean distance measure as explained in section 2. This means, everything so far happens unscaled only in the very last step we scale the constraint equation and go to the Euclidean space, as one of the strata of the hierarchy described in part I.

The linear equations can be solved for a set of correspondences by applying e.g. the householder method [27]. From the solution of the system of

equations, the motion parameters $\boldsymbol{R}, \boldsymbol{t}$ can easily be recovered by evaluating $\theta := \|\boldsymbol{v}\|$, $\boldsymbol{l} := \frac{\boldsymbol{v}}{\theta}$ and $(\boldsymbol{t} \cdot \boldsymbol{l}) := \frac{-\boldsymbol{m}}{\theta}$.

This procedure is iterated to converge to the whole rigid motion. Figure 4.1 visualizes the principle of such an approximation and iteration: The aim is to rotate a point $\underline{\boldsymbol{X}}$ around 90 degree 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) we get $\underline{\boldsymbol{X}}_1$ 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 4.1 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 the next section.

Note, that basically this estimation procedure corresponds to a gradient descend method in the 3D space.

4.1.2 Generating an example system of equations

In this section we will derive a system of equations for point, line and plane correspondences to visualize the type of equations which are obtained.

Let us assume two points

$$\boldsymbol{P}_1 = (p_{11}, p_{12}, p_{13}) \tag{4.4}$$
$$\boldsymbol{P}_2 = (p_{21}, p_{22}, p_{23}), \tag{4.5}$$

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

$$\boldsymbol{L} = \{\boldsymbol{L}_{d1} = (Ld_{11}, Ld_{12}, Ld_{13}), \tag{4.6}$$
$$\boldsymbol{L}_{m1} = (Lm_{11}, Lm_{12}, Lm_{13})\} \tag{4.7}$$

and one plane (containing a normal $\boldsymbol{P}_{d1}$ and Hesse distance hab_1),

$$\boldsymbol{Pl}_1 = \{\boldsymbol{P}_{d1} = (Pd_{11}, Pd_{12}, Pd_{13}), hab_1\}. \tag{4.8}$$

Let us further assume that $\boldsymbol{P}_1$ corresponds to $\boldsymbol{Li}_1$ and $\boldsymbol{P}_2$ corresponds to $\boldsymbol{Pl}_1$.

Then the matrix for the system of equations take the form

$$Ax = b, \tag{4.9}$$

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 takes the form

$$\begin{aligned} b \;=\; & (-p_{12}Ld_{13} + p_{13}Ld_{12} + Lm_{11}, -p_{13}Ld_{11} + p_{11}Ld_{13} + Lm_{12}, \\ & -p_{11}Ld_{12} + p_{12}Ld_{11} + Lm_{13}, \\ & -hab_1 + Pd_{11}p_{21} + Pd_{12}p_{22} + Pd_{13}p_{23})^T. \end{aligned} \tag{4.10}$$

The system of equations contains as unknowns the six twist parameters for which the equations are solved for. The matrices involving kinematic chains, circles and spheres take a comparable form, just modified with additional unknowns.

Note, that though the point correspondences give three equations the 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 gives every point-plane constraint exactly one equation. So at least six correspondences are necessary to get a unique solution.

4.1.3 Solving the Equations system

Many algorithms can be found in the literature to estimate coefficients of non-linear equations systems. A comparison of four approaches for pose estimation are made by Lorusso et.al. in [22]. 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 [39] 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 descend method within this context. A gradient descend method will be proposed in this work. Therefore we will now study the convergence rate of the gradient descend method for the case

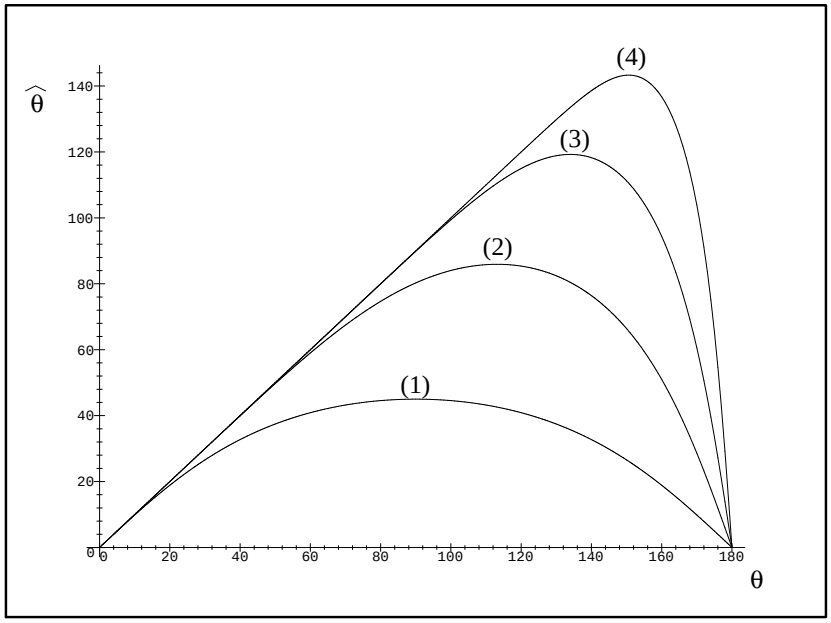

Fig. 4.2: Convergence rate of iterations for arbitrary angles between 0 and 180 degrees. The expected angles θ are on the x-axis and the estimated angles $\hat{\theta}$ are on the y-axis. The iterations $(1)\ldots(4)$ are overlaid.

of one unknown angle θ. The result is demonstrated in figure 4.2. The x-axis represents the wanted angle θ, the y-axis shows the estimated angle $\hat{\theta}$. Four iterations are overlaid. The functions are very characteristic and it can be 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 4.1. All angles, except that of 180 degree converge during the iteration, and for the most cases only a few iterations are sufficient to get a good approximation. In situations where only small rotations are assumed, for the most cases, two or three iterations are sufficient.

A comparison of this gradient descend method with a standard SVD-approach or Kalman filter will be done in the first experiment of the next section. There also the adaptive use of pose constraints is presented in more

detail.

4.2 Pose Estimation Experiments

This section shows experimental results which demonstrate that the theoretical approaches for pose estimation developed so far are extremely useful. The first experiment concerns the numerical analysis of the pose estimation algorithm and compares results of the gradient descend method with an SVD-approach and a Kalman filter. The second experiment concerns pose estimation of rigid objects containing points and lines. We show results on real images and explain how to combine the constraints and how to use them in a noise adaptive manner. Then, experiments with kinematic chains are presented. Finally, we describe experiments with *complicated* objects, which contain all different entities we have introduced so far. All information is used to estimate the pose and kinematic chain parameters of the objects simultaneously. The assumptions for our experiments are the following:

1. Corner features in the image are either manually extracted, or estimated by tracked point markers.

2. Edge features in the image are either reconstructed from two corners or estimated by applying a Hough transformation.

3. Image points on circles or conics are either extracted manually or by a contour algorithm on a silhouette.

4. We use a monocular (calibrated) camera. Only the projection matrix is given, we need no separation of the matrix into intrinsic and extrinsic camera parameters.

5. The 3D (Euclidean) object model is given in terms of feature sets on the object model (corners, lines, kinematic chain locations, etc.)

As explained in the previous section, since we only iterate linear equations containing always six unknowns for the rigid body motion and a few additional ones for the kinematic chains, the pose estimation itself can be carried out in real-time. So far, we are able to estimate the pose of an unknown object by given correspondences and projection matrices in the frame rate of 20 frames per second (fps) on a SUN Ultra 10 and a frame rate of 100 fps on a Linux 2 GHz machine. Note, that the equations are good conditioned with respect to the number of extracted and used image and object features.

4.2.1 Pose estimation of simple rigid objects

There exist several ways to estimate the motion parameters. In earlier works we concerned this problem and we estimated the motion parameters either on the Lie group $SE(3)$ itself (by using an SVD approach), or by using an extended Kalman filter (EKF) [36]. In our first experiment, we compare the noise sensitivity of these three methods (the two older one, and the gradient descend method presented in the last section), 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). Therefore we add a Gaussian noise on ex-

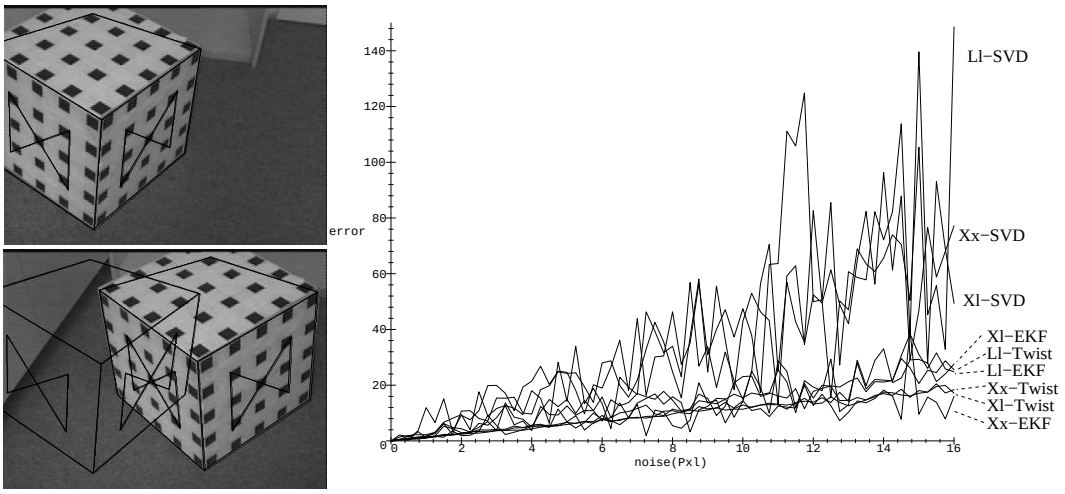

Fig. 4.3: The scenario of the first experiment. In the first image the calibration is performed and the 3D object model is projected on the image. Then the camera moved and corresponding line segments are extracted. For comparison reasons, the initial pose is overlaid. The diagram shows the performance comparison of different methods in case of noisy data.

tracted image points in a virtual scenario (see figure 4.3). Then we estimate the rigid body motion, and use the translational error between the ground truth and the disturbed values as error measure. The result is depicted in figure 4.3. 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.

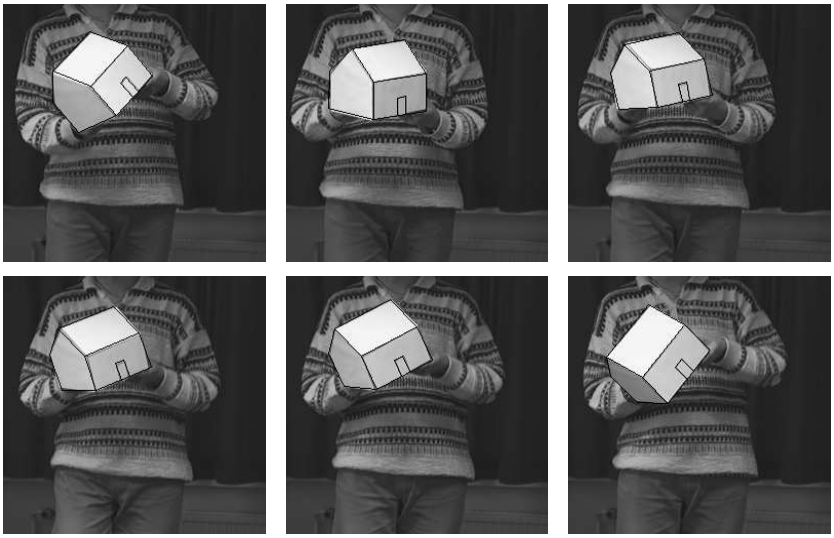

Fig. 4.4: Tracking a model house consisting of points and lines.

This occurs because we selected certain points directly by hand and derived from these the line subspaces. So the quality of the line subspaces is directly connected to the quality of the point extraction. The result of this investigation is, that for noise corresponding to a distribution function, the Kalman filter or twist approach for pose estimation should be used. There are two main reasons, why we further prefer the twist approach for pose estimation instead of the EKF: Firstly, the Kalman filter is sensitive to outliers (see e.g. figure 4.8), leading to non-converging results. Secondly, Kalman filters must be designed for special situations or scenarios. So the design of a general Kalman filter, dealing with different entities in a weighted manner is hard to implement. Instead, this can be done very easily in the twist approach since the linearized constraint equations of any entity can just be scaled and put in one system of equations. Figure 4.4 shows results of an automatic tracking algorithm developed and analyzed in [30]. 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 [2]. We use this approach for self localization and robot navigation tasks.

4.2.2 Adaptive use of pose estimation constraints

Image preprocessing algorithms sometimes enable a characterization of the quality of extracted image data (see e.g. [9]). The resulting question is

how to deal with noisy extracted image data. The idea to cope with this

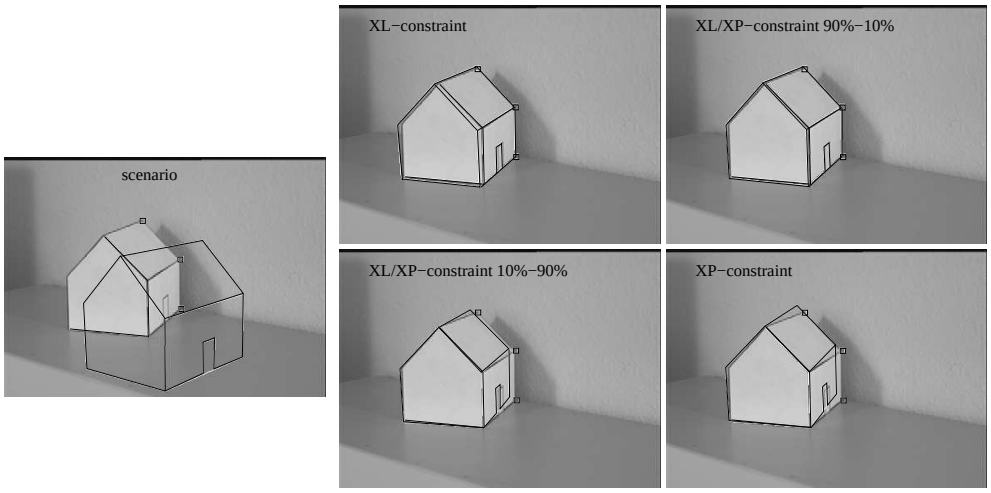

Fig. 4.5: Different weights of constraints for pose estimation.

problem in the context of pose estimation is very 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

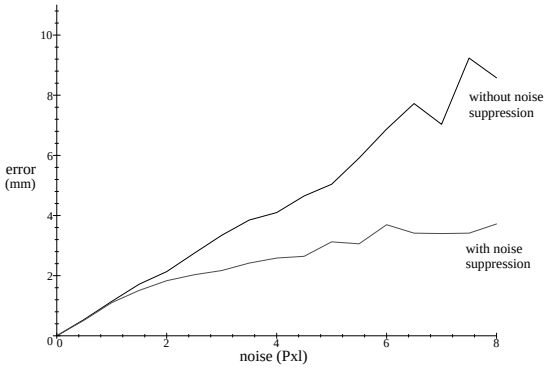

Fig. 4.6: Comparison of the results with and without noise suppression.

possible to individually scale the weights of the equations within the whole system of equations of an observed object. Figure 4.5 shows an example: We have only three extracted image points and three extracted image lines at hand (see left image). We can use both types of information separately to estimate the pose of the object. Since we have only a few information for

each type of correspondences, 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,

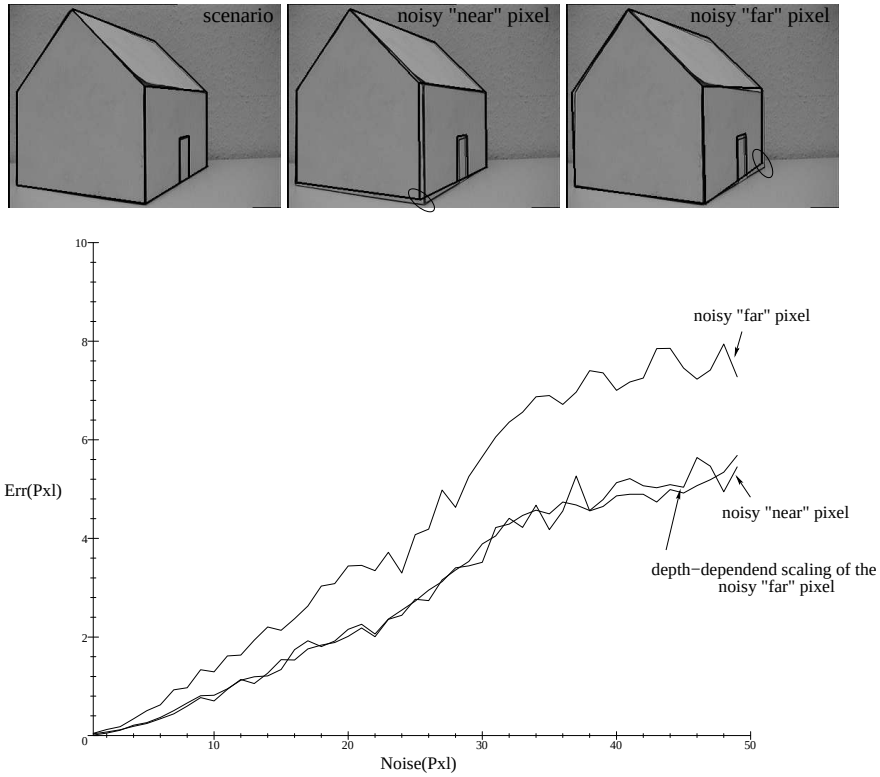

Fig. 4.7: Depth dependence of the constraints.

we can put both constraint equations in one single system of equations and solve the unknowns by using all available image information simultaneously. Furthermore, we are able to choose different weights of the constraints. The change of the estimated pose is visualized in the other images of figure 4.5. This experiment demonstrates that the presented approach enables to model adaptive observer behavior in a cognitive manner with respect to both the choice of image features at hand and with respect to take into account the trustworthiness of the data.

In an other experiment we simulate the possibility of noise adaptive use of the pose estimation constraints. For this we add a Gaussian noise on some of the extracted image points. Although we know from the problem of Gaussian noise modeling on the unit sphere [5], we will omit these problems here. In this experiment we work with six image features and add on two of them the Gaussian noise. Then we solve the constraint equations 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, we use those without noise as ground truth. We repeat the experiment for every noise level several times to get a smooth error function and choose for every noise level the mean value. The result is visualized in figure 4.6. It is easy to see, that we can use the constraints in a noise adaptive manner.

Figure 4.7 visualizes the *depth-dependence* of the used 3D constraints. As discussed in part I, there is a difference of 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 effect and its correction is analyzed in figure 4.7. For this experiment, we calibrate a scene with a model house. Then we pick out two points of the model, put a Gaussian noise on their corresponding image points and estimate their pose separately. The two chosen points differ in their relative depth with respect to the image plane as can be seen in figure 4.7. Then we estimate the absolute image error. This means, the error measure is now connected to the observation of the pose in the image plane. The graph in figure 4.7 shows the influence of the disturbed 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 disturbed 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 formalize the pose problem in an implicit manner, the constraints can be scaled with respect to their relative depth. This is shown in the third error curve of figure 4.7.

4.2.3 Pose estimation of kinematic chains

In the next experiment (see e.g. figure 4.8), we use as more complex object model the RX-90 robot arm [37]. Figure 4.8 shows some examples of a sequence containing 42 images. In this image sequence the first joint is moving in 5 degrees steps from 0 to 25 degrees. Then the second joint is moving in 5 degrees steps from 0 to 60 degrees. This is also shown in figure 4.9. We estimate the pose of the robot and the angles of the kinematic chain via tracked points markers. Figure 4.9 shows the joint angles estimated

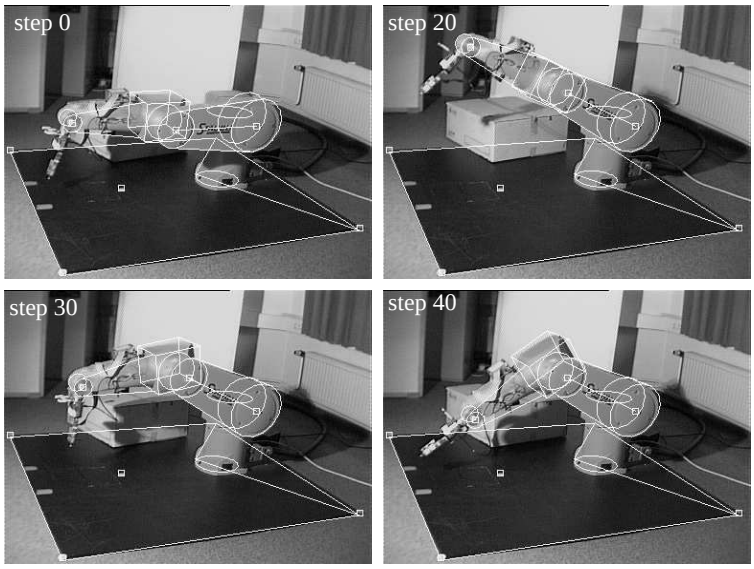

Fig. 4.8: Images of a tracked robot arm taken from a sequence with 40 images.

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) we could prove that (for not degenerate cases) the parameters during the iterations converge against the ground truth. The errors we gain in these experiments are dependent on the calibration quality, the lens distortions and the accuracy of the color marker detection.

Figure 4.10 shows an other image sequence. There we visualize the stability of our algorithm in the context of moved color markers which corresponds to impossible kinematics of the robot. Two things can be seen. First, we model the geometry of the robot within our constraints and the model will not be distorted. Instead, the algorithm leads to a spatially best fit of the model to the extracted image data. Second, we have no hierarchical approach for pose estimation of piecewise rigid objects as in [12, 40] but a pose estimation based on the model of a kinematic chain. Hierarchical pose estimation means that the pose problem is separated in subproblems 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

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

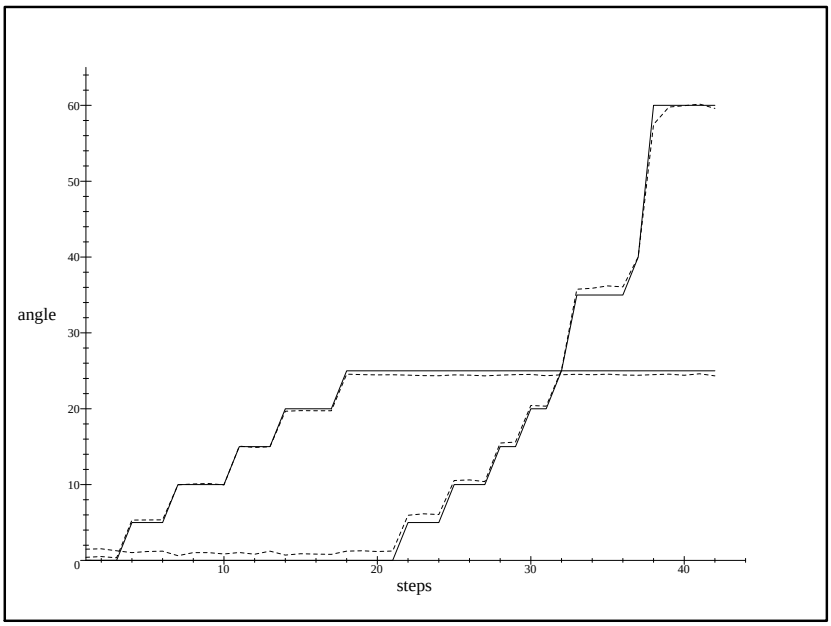

Fig. 4.9: Joint angles estimated and overlaid with the ground truth. The solid lines show the ground truth and the dashed lines show the estimated values.

the model in a second processing step. There are two main arguments why we do not recommend this method: Firstly, the geometry of the whole object is not modeled within the constraint equations. That necessitates the second processing step to ensure no 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 [4]. Second, each point of a kinematic chain contributes with two linear 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 not neglected due to redundancy of the algorithm.

4.2.4 Simultaneous pose estimation with different kinds of correspondences

This section concerns the use of more extended object concepts for pose estimation. In figure 4.11 pose estimation results of an object containing

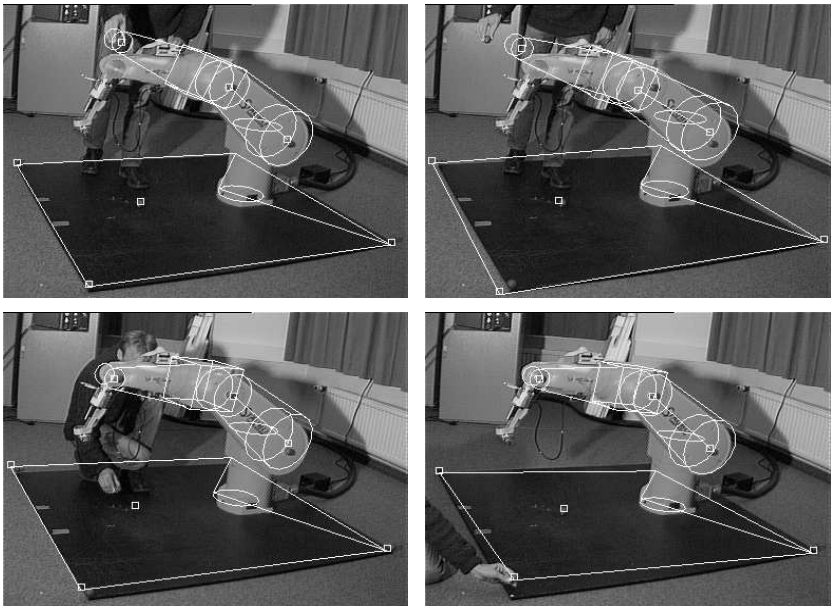

Fig. 4.10: Stability example for disturbed color markers and visualization of the geometry of the robot which is modeled within the constraints.

points, lines, kinematic chains and circles are presented.

In the last experiment, we use a model which contains a prismatic and a revolute joint. In addition to the model features of figure 4.11, here we also use a 3D sphere. The model is depicted in figure 4.12. Figure 4.13 shows some pose estimation results of the object model. Though we measured the size of the model by hand, the pose is accurate and also the joint parameters are good approximated. All informations is arranged in one linear system of equations, which leads to simultaneous solving of the pose parameters by using all different features.

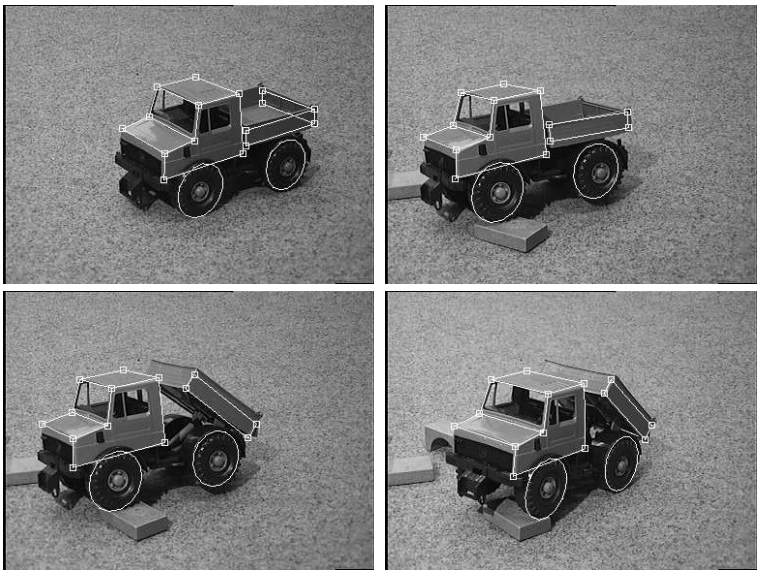

Fig. 4.11: Pose estimation of an object, consisting of 3D points, lines, circles and kinematic chain segments.

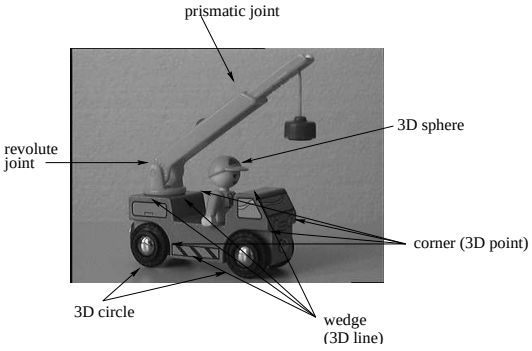

Fig. 4.12: Object model, consisting of different entities.

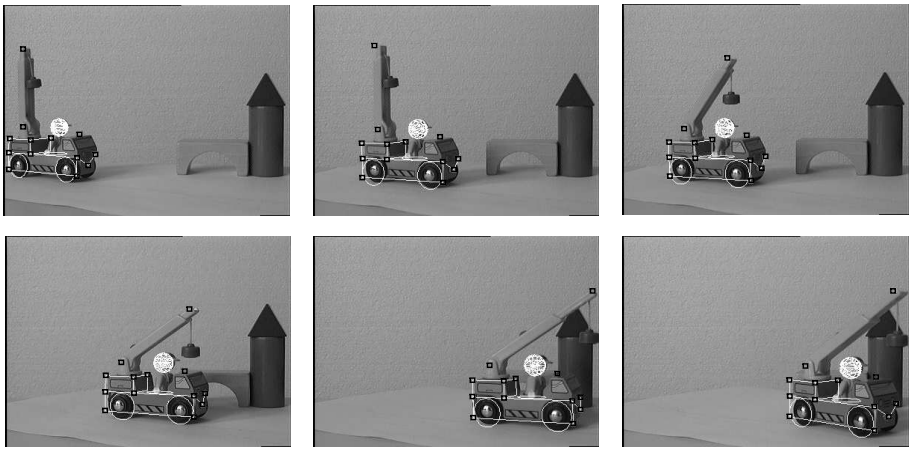

Fig. 4.13: Pose estimation by using all types of model features.

5. SUMMARY AND DISCUSSION

In this part several important topics for computer vision and robotics were discussed. First of all, we use the foundations of part I to deal with the pose estimation problem. The framework of conformal geometric algebra enables us to handle Euclidean, projective and conformal geometry by using suitable subalgebras. We present an extended framework for pose estimation of object models, which consist of different types of entities, including points, lines, planes, circles, spheres and kinematic chains. We present efficient approaches to solve the pose estimation problem numerically by using all information simultaneously. Our experiments with monocular pose estimation of kinematic chains show that this is a first step to advantageously cope with robot vision problems [26] in an advanced algebraic way. The algorithm for pose estimation of kinematic chains, compared with e.g. [4] is more efficient since we do not build constraints in the image plane, but constraints in the 3D space. Furthermore, we are able to use a full perspective camera model in this context and not only an orthographic one. The equations of different constraints are put into one system of equations for estimating the parameters of the rigid body motion. Since each point of a kinematic chain contributes with two linearly independent equations, also the higher order points of kinematic chains influence the result of the whole pose. This is an advantage over classical hierarchical approaches as recommended in [12, 40].

Instead of using invariances as an explicit formulation of ideal geometry, we are using implicit formulations of geometry as geometric constraints. Since in our constraints spatial distance measures have to be minimized, we can quite easily deal with noisy image features, inexact calibrated cameras and noisy object model features. Because the optimization is performed with respect to the spatial distance in Euclidean space, the task of pose estimation is more simple in comparison to the minimization of distances on the manifold of rigid body motions as performed in [38, 5].

In particular the noise adaptive use of the constraints in this context is very interesting with respect to the design of behavior based [34] or learning robot systems. Several articles concerning the fusion of noisy data, e.g. [13], can be compared with our approach in that respect. But we are also interested to apply different kinds of entities with different reliabilities of

measurements so that a system is able to adapt and to pick up the needed information by itself. Only few works exist so far which deal with this important topic for stable running systems, e.g. [16].

We implemented the problem in C++ [19] and are able to estimate the motion (and kinematic chain) parameters in real-time with 20 frames per second on a SUN Ultra 10 and gain 80 fps on a Linux 2GHz machine.

Our very recent work concerns extensions of this feature based pose approach and is presented in [32]. There we use virtually coupled twists to yield a special family of curves (so-called 3D *cycloidal curves*) and extend this approach to general free-form contours.

Acknowledgments

We would like to thank Oliver Granert, Daniel Grest, Norbert Krüger and Christian Perwass for fruitful discussions and hints for performing this work. This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).

BIBLIOGRAPHY

[1] Bayro-Corrochano E. and Kähler D. Kinematics of robot manipulators in the motor algebra. In [35], pp. 473-490, 2001.

[2] Beveridge J.R. Local search algorithms for geometric object recognition: Optimal correspondence and pose. *Technical Report CS 93–5, University of Massachusetts*, 1993.

[3] Blaschke W. Kinematik und Quaternionen, Mathematische Monographien 4. *Deutscher Verlag der Wissenschaften*, 1960.

[4] Bregler C. and Malik J. Tracking people with twists and exponential maps. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Santa Barbara, California, pp.8-15 1998.

[5] 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.

[6] Daniilidis K. Hand-eye calibration using dual quaternions. *Int. Journ. Robotics Res*, Vol. 18, pp. 286–298, 1999.

[7] 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.

[8] 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.

[9] 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*, Springer-Verlag, LNCS 1888, pp. 175-185, 2000.

[10] Gallier J. Geometric Methods and Applications for Computer Science and Engineering. *Springer Verlag*, New York, 2001.

[11] Grimson W. E. L. Object Recognition by Computer. *The MIT Press, Cambridge, MA*, 1990.

[12] Hauck A. Lanser S and Zierl C. Hierarchical recognition of articulated objects from single perspective views. *IEEE: Computer Vision and Pattern Recognition*, Puerto Rico, pp.870-876, 1997.

[13] Hel-Or Y. and Werman M. Pose estimation by fusing noisy data of different dimensions. *IEEE Transactions on Pattern Analysis and Machine Inteligence (PAMI)*, Vol. 17, No.2, pp. 195-201, February 1995.

[14] Hestenes D. and Ziegler R. Projective geometrie with Clifford algebra. *Acta Applicandae Mathematicae*, Vol. 23, pp.25–63, 1991.

[15] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [35], pp. 3-23, 2001.

[16] 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.7:2, pp. 126–136, 1996.

[17] Homer H.H. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13:6, pp.530–541, 1991.

[18] 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*, Vol. 15, pp. 225–243, 1995.

[19] KiViGraP (Homepage of the Kieler Vision and Grasping Project). http://www.ks.informatik.uni-kiel.de/~kivi/kivi.html.

[20] Lowe D.G. Solving for the parameters of object models from image descriptions. in *Proc. ARPA Image Understanding Workshop*, 1980, pp. 121-127.

[21] Lowe D.G. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, Vol. 31, No. 3, 1987, pp. 355-395.

[22] 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.

[23] Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [35], pp. 27-52, 2001.

[24] Murray R.M., Li Z. and Sastry S.S. A Mathematical Introduction to Robotic Manipulation. *CRC Press*, 1994.

[25] Needham T. Visual Complex Analysis. *Oxford University Press*, 1997

[26] Pauli J. Development of Camera-Equipped Robot Systems *Technical Report 9904,Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2000.

[27] Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T. Numerical Recipes in C. Cambridge University Press, 1993.

[28] Perwass C. and Lasenby J. A novel axiomatic derivation of geometric algebra. Technical Report CUED/F - INFENG/TR.347, Cambridge University Engineering Department, 1999.

[29] Rosenhahn B., Granert O., Sommer G. Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering, Birkhäuser Verlag, Dorst L., Doran C. and Lasenby J. (Eds.)*, pp. 373-383, 2001.

[30] 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. (Eds.), Springer-Verlag Heidelberg, LNCS 1998, pp. 9-19, 2001.

[31] 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.

[32] Rosenhahn B., Perwass Ch. and Sommer G. Pose Estimation of 3D Freeform Contours. *Technical Report 0207,Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2002.

[33] Shevlin F. Analysis of orientation problems using Plücker lines. *International Conference on Pattern Recognition, Brisbane*, Vol. 1, pp.685–689, 1998.

[34] Sommer G. Algebraic aspects of designing behavior based systems *Algebraic Frames for the Perception-Action Cycle, AFPAC'97*, Eds. Sommer G. and Koenderink J.J., Springer-Verlag Heidelberg, LNCS 1315, pp.1–28, 1997.

[35] Sommer G., editor. Geometric Computing with Clifford Algebra. *Springer Verlag Heidelberg*, 2001.

[36] Sommer G., Rosenhahn B., and Zhang Y. Pose estimation using geometric constraints. In: *Multi-Image Search and Analysis, R.Klette, Th. Huang, G.Gimmel'farb (Eds.)*, LNCS 2032, Springer-Verlag, Heidelberg, pp. 153–170, 2001.

[37] Stäubli Roboter RX90-CS7 Instruction Manual *D280.190.12.C, November*, 1992.

[38] 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.

[39] Walker M.W. and Shao L. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, Vol. 54:3, pp.358–367, 1991.

[40] 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. (Eds.)*, Springer-Verlag Heidelberg, LNCS 1998, pp. 27-34, 2001.