INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Pose Estimation of Free-form Objects

(Theory and Experiments)

Bodo Rosenhahn, Gerald Sommer, Reinhard Klette

Bericht Nr. 0401
March 2004

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

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

Pose Estimation of Free-form Objects

(Theory and Experiments)

Bodo Rosenhahn, Gerald Sommer, Reinhard Klette

Bericht Nr. 0401
March 2004

e-mail: bros028@cs.auckland.ac.nz,
gs@ks.informatik.uni-kiel.de,
r.klette@auckland.ac.nz

Pose Estimation of Free-form Objects

(Theory and Experiments)
CITR-TR-137

Bodo Rosenhahn[1], **Gerald Sommer**[2], **Reinhard Klette**[1]

[1] Centre for Image Technology and Robotics (CITR)
University of Auckland
Tamaki Campus
Private Bag 92019
Auckland
http://www.citr.auckland.ac.nz

[2] Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel
http://www.ks.informatik.uni-kiel.de

email:
bros028@cs.auckland.ac.nz
gs@ks.informatik.uni-kiel.de
r.klette@auckland.ac.nz

March 8, 2004

ABSTRACT

In this report we present geometric foundations and an algorithmic approach to deal with the 2D-3D pose estimation problem for free-form surface models. This work is an extension to earlier studies presented in [29]. The discussion of 1D contour models in [29] is extended to 2D free-form surface models. We use a parametric representation of surfaces and apply Fourier transformations to gain low-pass descriptions of objects. We present an algorithm for pose estimation, which uses the silhouette of the object as pictorial information and recovers the 3D pose of the object even for changing aspects of the object during image sequences. We further present extensions to couple surface and contour information on objects and show the potential of our chosen approach for complex objects and scenes.

CONTENTS

1. INTRODUCTION

Pose estimation has been studied in computer vision since its beginning. It is crucial for many computer and robot vision tasks. Object grasping, manipulation and recognition or self-localization of mobile robots are typical examples for the use of pose estimation. For a definition of the pose problem, we quote Grimson [13]:

Definition 1.1 *By pose we mean the transformation needed to map an object model from its inherent coordinate system into agreement with the sensory data.*

In this report we are interested in the so-called 2D-3D pose estimation problem. With 2D-3D pose estimation we mean to fit 2D measurement data (an image of an object) with a 3D object model. We are interested in estimating a rigid motion (containing both 3D rotation and 3D translation) which *fits* the object data with the image data. This is visualized in figure 1.1: We first observe an object in an image (left). Furthermore the camera is calibrated, so that we can localize a camera coordinate system in the 3D world (visualized with the optical center and the shifted camera plane in the middle image). The object model is shown in the right image in its local coordinate system. Our aim is to compare the 3D object with the 2D image data to define a fit-value. We are interested in estimating that rigid motion which leads to the *best* fit between the object model and the image data as visualized in figure 1.2.

Though the problem sounds simple on a first glimpse, there are several important (and partially still open) questions which are discussed in the literature:

1. How will the involved mathematical spaces be dealt with?
2. How will a 3D object be compared with 2D image data?
3. Which kind of image information is to be used?
4. How is a rigid motion estimated?
5. How will 3D object models be represented?

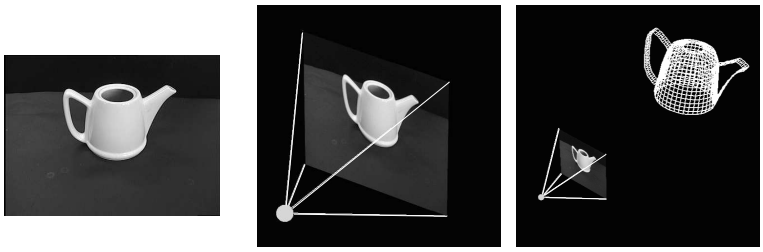

Fig. 1.1: Our assumptions for the 2D-3D pose estimation problem: (left) An image of an object is assumed. (middle) The camera is assumed to be calibrated with respect to a world coordinate system. (right) A 3D object model (in its own object coordinate system) is also given.

Indeed some of the questions can not be answered in a generalized manner. Instead they are dependent on the situation: For some scenes (e.g. industrial applications), a point based representation within an orthographic camera model and a simple corner extractor might be sufficient, but for more general scenes, this can be an inadequate representation. Therefore there is need to deal with the pose problem in a more general and flexible manner. In this work we do not give a long overview about the existing literature (e.g. [4, 9, 13, 19, 20, 23, 24, 35, 37]). Instead we refer to [11, 29] where different works are presented and discussed. This report can be seen as an extension to [29]. This earlier work investigates the pose problem and presented solution approaches to the above listed questions. But there are still remaining questions which we start to answer in this report. We first summarize our former results in [29]:

1. How will the involved mathematical spaces be dealt with?
   Three mathematical spaces can be found in the 2D-3D pose estimation problem. These are the Euclidean space (the space in which the object model and its size is given), the kinematic space (where the rigid motions are the unknown transformations) and the projective space (where the calibrated camera and the image data are given). To deal with the pose estimation problem, there is need to let interact these spaces in an efficient manner. We propose the use of the conformal geometric algebra (CGA) [21]. The CGA is build up on a conformal model of the Euclidean space which is coupled with a homogeneous model to deal with kinematics and projective geometry simultaneously. This enables us to deal with the Euclidean, kinematic and projective space in one framework and therefore to cope with the pose problem in an efficient manner. Furthermore, the unknown rigid motion is expressed as a screw-motion which is caused by an orthogonal operator called,

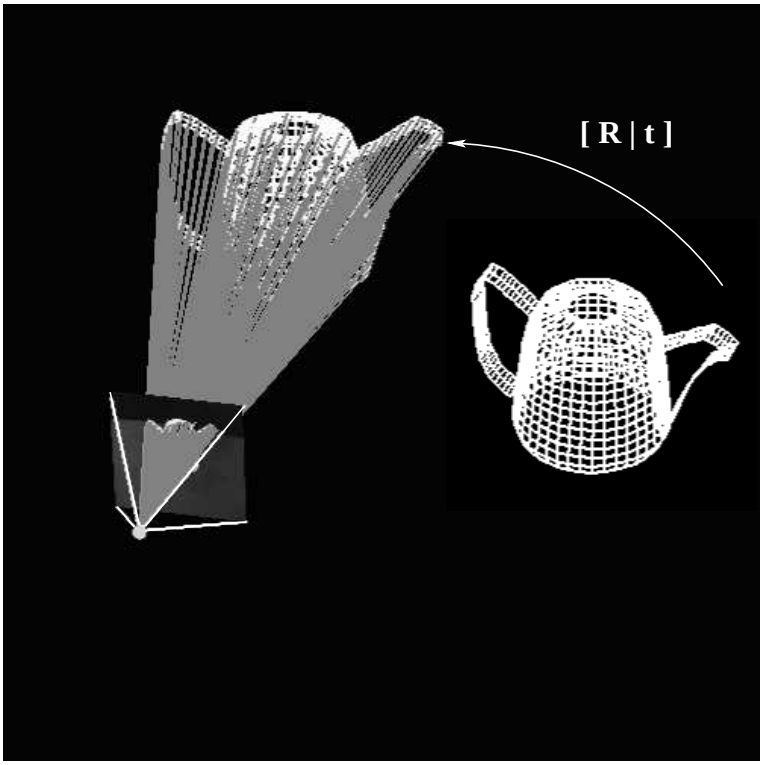

Fig. 1.2: Visualization of the pose estimation problem: The task is to find the rigid motion which leads to a best fit between image data and the object model.

motor. In contrast to the homogeneous Euclidean space, in conformal space a motor can be applied not only to points but to any higher-order geometric entity in a linear manner, just by computing the geometric product. This leads to compact and easily interpretable equations.

2. How will a 3D object be compared with 2D image data?
   They are compared by reconstructing 2D image features (e.g. points) to 3D entities (e.g. 3D projection rays). By using this strategy we are able to formalize the pose problem as a pure 3D problem and to model a distance measure between 2D image data and 3D object data in plain 3D.

3. Which kind of image information is to be used?
   There does not exist an unique answer to this question, since it is dependent on the scenario, the used object model and the task. [29] uses on the one hand corner or line features, but also 2D contour information which results in silhouette based pose estimation.

4. How to estimate a rigid motion?
   We compared three different algorithms for estimating pose parameters. The first is a simple SVD-approach, the second a Kalman filter and the third is a gradient descent method. The gradient descent method uses a twist representation of rigid motion: Since the rigid motions constitute a Lie group, there always exists a Lie algebra which generates the group actions by applying the exponential function. The elements of the Lie algebra forming the group of rigid motion are called twists. These model the motion in terms of a screw motion [25, 32, 4]. Therefore, we call the gradient descent method the *twist* approach. It proved as fast, stable and adaptive. Furthermore with this approach we are not only able to deal with 2D-3D point correspondences, but also with 2D line and 3D point correspondences or 2D line and 3D line correspondences simultaneously. For a comparison of pure 3D point based pose estimation algorithms, the reader should also consult [18].

5. How will 3D object models be represented?
   This question can also not be answered in general, since the *good* representation of an object is dependent on the object itself, its location in the environment and the task of the system. This is the reason why we introduced an object hierarchy and started with simple point and line features for object modeling. Then we extended them to kinematic chains (comparable to [4]) and used the concept of *coupled twists* to model cycloidal curves and surfaces [22, 7] as higher order features for object representation. Then we showed the direct connection between these kinematically generated curves to 3D contours represented by Fourier descriptors [1] and ended up in silhouette based pose estimation of objects modeled by free-form contours. Missing in [29] is the extension to pose estimation of free-form surface models. The aim of this report is to close this gap.

In this report we investigate an approach for pose estimation of free-form surface models. We quote Besl [3] for a definition of free-form surfaces:

Definition 1.2 *A free-form surface has a well defined surface that is continuous almost everywhere except at vertices, edges and cusps.*

For geometric and topological classifications of surfaces see [17]. We start with the description of objects as 2-parametric surfaces and introduce a Fourier representation which allows us to use a low-pass approximation of the object model. Then we introduce the pose estimation algorithm and start with a silhouette based approach. Finally, we continue to use additional internal features for stabilizing the estimation results.

Before we can introduce this extension of the pose estimation of free-form surface models, we have to summarize basic notations, algorithms and formalizations in the next chapter. This will also be necessary because the later introduced pose estimation algorithm for surface models goes back to a contour based algorithm.

2. PREVIOUS WORKS

In this section we summarize parts of our previous works [29]. We start with a brief introduction into geometric algebras (GAs), but will only introduce the main definitions and notations. A more detailed introduction can be found in [29, 27, 5, 28, 33, 15]. The second section introduces the point based pose estimation, starting with the basic constraint equation and its analysis. Then we consider the twist-approach for pose estimation. The third section summarizes the contour based free-form pose estimation algorithm. We start with the representation of 3D free-form contours within a Fourier model. Furthermore, we introduce the ICP-algorithm for tracking objects and present some extensions (e.g. outlier elimination).

2.1 Geometric algebras

What is currently called *geometric algebra* [15] can be seen as a Clifford[1] algebra with its main focus on a suited geometric interpretation. Clifford's contribution of inventing a geometric extension of the real number system to a complete algebraic representation of directed numbers is historically reviewed by M. Yaglom in [36]. In [36] there is also enlightened the relation to former works of Grassmann (1809-1877) or Hamilton (1805-1865). The term geometric algebra was introduced by David Hestenes in the 1960's, who further developed Clifford algebra in classical geometry and mechanics.

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

[1] William K. Clifford (1845-1879).

with associated splits.

In general a geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ $(p,q,r,\in \mathbb{N}_0)$ 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. A geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ is constructed from a vector space $\mathbb{R}^{p,q,r}$, endowed with the signature (p,q,r), by application of a *geometric product*. The notation $\boldsymbol{\mathcal{G}}_{p,q,r}(\mathbb{R}^{p,q,r})$ is sometimes used to stress the vector space and its signature the geometric algebra is built from. The product defining a geometric algebra is called *geometric product* and is denoted by juxtaposition, e.g. $\boldsymbol{AB}$ for two multivectors $\boldsymbol{A}$ and $\boldsymbol{B}$. The geometric product of vectors consists of an outer $(\wedge)$ product and an inner $(\cdot)$ product. Their effect is to increase or to decrease the grade of the algebraic entities, respectively.

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

$$\mathbf{e}_i\mathbf{e}_j \;:=\; \begin{cases} 1\in\mathbb{R} & \text{for} \quad i=j\in\{1,\ldots,p\} \\ -1\in\mathbb{R} & \text{for} \quad i=j\in\{p+1,\ldots,p+q\} \\ 0\in\mathbb{R} & \text{for} \quad i=j\in\{p+q+1,\ldots,n\} \\ \mathbf{e}_{ij} & \text{for} \quad i\neq j, \end{cases} \tag{2.1}$$

with $\mathbf{e}_{ij}=\mathbf{e}_i\wedge\mathbf{e}_j=-\mathbf{e}_j\wedge\mathbf{e}_i$. The geometric product of the same two basis vectors leads to a scalar, whereas the geometric product of two different basis vectors leads to a new entity, which is called a *bivector*. This bivector represents the subspace, spanned by these two vectors.

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

$$\boldsymbol{M} \;=\; \sum_{i=0}^{n}\langle\boldsymbol{M}\rangle_i, \tag{2.2}$$

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

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

$$\boldsymbol{u} \cdot \boldsymbol{v} := \frac{1}{2}(\boldsymbol{uv} + \boldsymbol{vu}), \tag{2.3}$$

$$\boldsymbol{u} \wedge \boldsymbol{v} := \frac{1}{2}(\boldsymbol{uv} - \boldsymbol{vu}). \tag{2.4}$$

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

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

$$(\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \begin{cases} ((\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot \boldsymbol{v}_1) \cdot (\boldsymbol{v}_2 \wedge \ldots \wedge \boldsymbol{v}_s) & \text{if } r \geq s \\ (\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_{r-1}) \cdot (\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s)) & \text{if } r < s, \end{cases} \tag{2.5}$$

with

$$(\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot \boldsymbol{v}_1 = \sum_{i=1}^{r} (-1)^{r-i} \boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_{i-1} \wedge (\boldsymbol{u}_i \cdot \boldsymbol{v}_1) \wedge \boldsymbol{u}_{i+1} \wedge \ldots \wedge \boldsymbol{u}_r, \tag{2.6}$$

$$\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \sum_{i=1}^{s} (-1)^{i-1} \boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_{i-1} \wedge (\boldsymbol{u}_r \cdot \boldsymbol{v}_i) \wedge \boldsymbol{v}_{i+1} \wedge \ldots \wedge \boldsymbol{v}_s. \tag{2.7}$$

The blades of highest grade are n-blades, called *pseudoscalars*. Pseudoscalars differ from each other by a nonzero scalar only. For non-degenerate geometric algebras there exist two unit n-blades, called the *unit pseudoscalars* $\pm\boldsymbol{I}$.

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

$$[\boldsymbol{P}] := \boldsymbol{P}\boldsymbol{I}^{-1}. \tag{2.8}$$

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

$$\begin{aligned} [\boldsymbol{v}_1 \ldots \boldsymbol{v}_n] &= [\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n] \\ &= (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_n)\boldsymbol{I}^{-1}. \end{aligned} \tag{2.9}$$

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

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

$$\boldsymbol{X}^\star \;:=\; \boldsymbol{X}\boldsymbol{I}^{-1}. \tag{2.10}$$

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

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

$$\begin{aligned}\widetilde{\boldsymbol{A}_{\langle s\rangle}} &= (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \ldots \wedge \boldsymbol{a}_{s-1} \wedge \boldsymbol{a}_s)^\sim \\ &= \boldsymbol{a}_s \wedge \boldsymbol{a}_{s-1} \wedge \ldots \wedge \boldsymbol{a}_2 \wedge \boldsymbol{a}_1.\end{aligned} \tag{2.11}$$

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

For blades $\boldsymbol{A}$ and $\boldsymbol{B}$ it is possible to use the join to express *meet* operations: Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two arbitrary blades and let $\boldsymbol{J} = \boldsymbol{A}\dot{\wedge}\boldsymbol{B}$, then

$$\boldsymbol{A} \vee \boldsymbol{B} \;:=\; \left(\boldsymbol{A}\boldsymbol{J}^{-1} \wedge \boldsymbol{B}\boldsymbol{J}^{-1}\right)\boldsymbol{J}, \tag{2.12}$$

defines the meet $\vee$, also called the shuffle product, which is a common factor of $\boldsymbol{A}$ and $\boldsymbol{B}$ with the highest grade.

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

$$\boldsymbol{A}\boldsymbol{B} \;=\; \frac{1}{2}(\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A}) + \frac{1}{2}(\boldsymbol{A}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}) =: \boldsymbol{A}\overline{\times}\boldsymbol{B} + \boldsymbol{A}\underline{\times}\boldsymbol{B}. \tag{2.13}$$

The reader should consult [27] 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.

The algebra we use to deal with the pose estimation problem is the conformal geometric algebra. Therefore we now give a brief introduction into this algebra.

2.2 Conformal geometric algebra

The basic idea behind this algebra are stereographic projections: Simply speaking, a stereographic projection is one way to generate a flat map of the earth. A stereographic projection has a clear geometric description and is visualized for the 1D case in figure 2.1: 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. The interception theorems can be applied to achieve the following

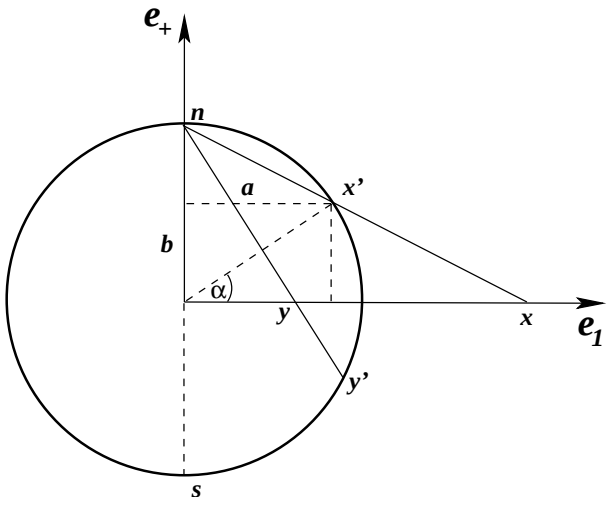

Fig. 2.1: Visualization of a stereographic projection for the 1D case: Points on the circle are projected onto the line or vice versa.

formulas for projecting a point on a unit circle and vice versa: A point $\boldsymbol{x}'$ on the circle is given by its angle α:

$$\begin{aligned} \boldsymbol{x}' &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \cos(\alpha)\mathbf{e}_1 + \sin(\alpha)\mathbf{e}_+. \end{aligned} \tag{2.14}$$

The point on the line then has the coordinates

$$\boldsymbol{x} = \left(\frac{\cos(\alpha)}{1-\sin(\alpha)}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{2.15}$$

To project a point $x\mathbf{e}_1$ ($x \in \mathbb{R}$) onto the circle the following equation holds [26],

$$\begin{aligned} \boldsymbol{x}' &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+. \end{aligned} \tag{2.16}$$

Using homogeneous coordinates (i.e. using the additional basis vector $\mathbf{e}$), the three basis vectors $\mathbf{e}_1$, $\mathbf{e}_+$ and $\mathbf{e}$ are now spanning the considered homogeneous model of the conformal space. This leads to a homogeneous representation of the point on the circle as

$$\boldsymbol{x}' = x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}. \tag{2.17}$$

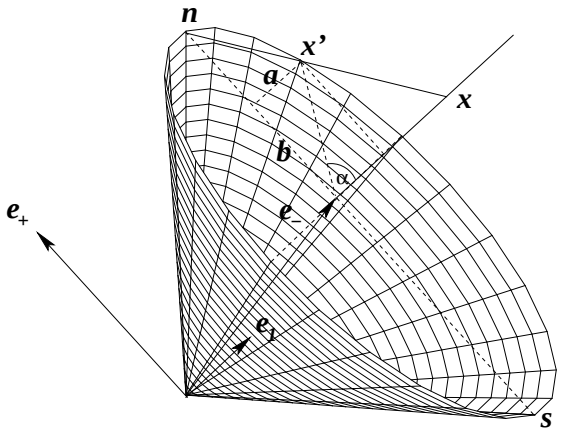

Fig. 2.2: Visualization of the homogeneous model for stereographic projections for the 1D case. All stereographically projected points lie on a cone, which is a null-cone in the Minkowski space. Note that in comparison to figure 2.1 the coordinate axes are rotated and drawn perspectively.

In [21] $\mathbf{e}$ is defined to have a negative signature, and therefore $\mathbf{e}$ is replaced with $\mathbf{e}_-$, whereby $\mathbf{e}_-^2 = -1$. This has the advantage that in addition to using a homogeneous representation of points, they are further embedded in a Minkowski space. Euclidean points, stereographically projected onto the unit circle in figure 2.2, are unit points and then represented by the set of null vectors in the new space. A Euclidean point is mapped to the conformal space by

$$\boldsymbol{x} \quad \Rightarrow \quad \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_-, \tag{2.18}$$

with

$$(\boldsymbol{x}')^2 = a^2 + b^2 - 1 = 0. \tag{2.19}$$

The coordinates (a, b) are the coordinates of a point on the unit circle. Note that each point in Euclidean space is in fact represented by a line of null vectors in the new space: the scaled versions of the null vector on the unit circle. This homogeneous representation of a point is used as *point* in the conformal geometric algebra. This is shown in figure 2.2.

To introduce CGA, we follow [21] and start with a *Minkowski plane*, $\boldsymbol{\mathcal{G}}_{1,1}$. Its vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the properties

$$\mathbf{e}_+^2 = 1, \quad \mathbf{e}_-^2 = -1, \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0. \tag{2.20}$$

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

Tab. 2.1: The entities and their dual representations in CGA.

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

$$\mathbf{e}_0 := \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} := \mathbf{e}_- + \mathbf{e}_+. \tag{2.21}$$

We use these two additional basis vectors to define the conformal geometric algebra, $\boldsymbol{\mathcal{G}}_{4,1}$, for the 3D case. The algebra $\boldsymbol{\mathcal{G}}_{4,1}$ contains $2^5 = 32$ elements. We further denote the conformal unit pseudoscalar as

$$\boldsymbol{I}_C \;=\; \mathbf{e}_{+-123} = \boldsymbol{E}\mathbf{e}_{123} = \boldsymbol{E}\boldsymbol{I}_E. \tag{2.22}$$

The points of the CGA, $\underline{\boldsymbol{x}}$, are related to those of the Euclidean space $\boldsymbol{x}$ by

$$\underline{\boldsymbol{x}} \;=\; \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0. \tag{2.23}$$

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

$$\begin{aligned} \underline{\boldsymbol{x}} &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0 \\ &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2(\mathbf{e}_+ + \mathbf{e}_-) + \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \\ &= \boldsymbol{x} + \left(\frac{1}{2}\boldsymbol{x}^2 - \frac{1}{2}\right)\mathbf{e}_+ + \left(\frac{1}{2}\boldsymbol{x}^2 + \frac{1}{2}\right)\mathbf{e}_-. \end{aligned} \tag{2.24}$$

This is exactly the homogeneous representation of a stereographically projected point onto the circle, given in equation (2.17) for the 1D case. A point is only more compactly written by using $\{\mathbf{e}, \mathbf{e}_0\}$, instead of $\{\mathbf{e}_+, \mathbf{e}_-\}$.

Since we now have the basic representation of a point in CGA we are able to define other entities based on this point representation. The entities and

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

Tab. 2.2: Table of conformal transformations, versors and scaling parameters.

their dual representation are summarized in table 2.1. This table is taken from [29, 21]. As can be seen, it is possible to express points, line, planes, circles, spheres and point pairs in this algebra. Lines are given in Plücker form (direction $\boldsymbol{r}$ and moment $\boldsymbol{m}$) and planes are given in Hessian form (normal $\boldsymbol{n}$ and distance d).

2.2.1 Conformal transformations

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

$$\sigma\underline{\boldsymbol{x}}' \;=\; G\underline{\boldsymbol{x}}G^{-1}, \tag{2.25}$$

where G is a versor and σ a scalar. Table 2.2, taken from [29, 21], summarizes the conformal transformations. The first column shows the type of operation performed with the versor product. The second column shows as example the result of a transformation acting on a point. The third column shows the versor which has to be applied and the last column shows the scaling parameter σ which is (sometimes) needed to result in a homogeneous point and to ensure the scaling $\underline{\boldsymbol{x}}' \cdot \mathbf{e} = \underline{\boldsymbol{x}} \cdot \mathbf{e} = -1$.

To express the pose estimation problem, we need a versor to express the rigid body motion. 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. The rotation of an entity can be performed just by multiplying the entity from the left with the rotor $\boldsymbol{R} \in \langle\boldsymbol{\mathcal{G}}_{4,1}\rangle_2$ and from the right with its reverse $\widetilde{\boldsymbol{R}}$. For example, a rotation of a point can be written as

$$\underline{\boldsymbol{x}}' \;=\; \boldsymbol{R}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{R}}. \tag{2.26}$$

The rotor $\boldsymbol{R}$ is given as

$$\begin{aligned} \boldsymbol{R} &= \exp\left(-\frac{\theta}{2}\boldsymbol{n}\right) \\ &= \cos\left(\frac{\theta}{2}\right) - \boldsymbol{n}\sin\left(\frac{\theta}{2}\right). \end{aligned} \tag{2.27}$$

Here $\boldsymbol{n}$ is a unit bivector representing the plane of the rotation (its dual $\boldsymbol{n}^\star$ corresponds to the rotation axis) and $\theta \in \mathbb{R}$ represents the amount of rotation. The negative value $-\theta$ in the rotor is used to gain a counter clockwise rotation and therewith a mathematically positive rotation.

To translate an entity with respect to a translation vector $\boldsymbol{t} \in \langle \boldsymbol{\mathcal{G}}_3 \rangle_1$, it is possible to use a so called *translator*, $\boldsymbol{T} \in \langle \boldsymbol{\mathcal{G}}_{4,1} \rangle_2$,

$$\boldsymbol{T} = (1 + \frac{\mathbf{e}\boldsymbol{t}}{2}) = \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right). \tag{2.28}$$

Similar to a rotation, an entity can be translated by multiplying the entity from the left with the translator $\boldsymbol{T}$ and its reverse $\widetilde{\boldsymbol{T}}$ from the right,

$$\underline{\boldsymbol{x}}' = \boldsymbol{T}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{T}}. \tag{2.29}$$

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

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}. \tag{2.30}$$

It is a special even grade multivector, called a motor.

Now follows a further definition of a motor in CGA based on the so-called *twists*. The idea is to interpret a motor as an element of the Lie group of rigid body motion and to construct it by exponentiation of its generator as a Lie algebra member. In fact, the mentioned twist is the generator of the motor. So every rigid body motion can be expressed as a twist or screw motion [25], which is a rotation around a line in space (in general not passing through the origin)[2] combined with a translation along this line. In CGA it is possible to use the rotors and translators to express screw motions in space. A screw motion is defined by an axis $\boldsymbol{l}^\star$, a pitch h and a magnitude θ. The *pitch* of the screw is the ratio of translation to rotation, $h := \frac{d}{\theta}$ $(d, \theta \in \mathbb{R},\ \theta \neq 0)$. If $h \to \infty$, then the corresponding screw motion consists of a pure translation along the axis of the screw. The resulting motor takes the form [29]

$$\boldsymbol{M} = \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}\boldsymbol{m}\right)\right). \tag{2.31}$$

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

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

2.3 Point based pose estimation

Now we have introduced the foundations of the used mathematical framework and we are able to express the 2D-3D pose estimation problem for point correspondences: Recalling figure 1.2, we have the following assumptions: We assume to have extracted 2D image points, e.g. corners of a calibrated camera and know their correspondence to 3D points, e.g. corners of the assumed object model. To compare these features, we reconstruct projection rays from the image points and claim that the transformed 3D points must be incident with the reconstructed rays. This can be expressed in CGA in the following way: Let $\underline{\boldsymbol{X}} \in \boldsymbol{\mathcal{G}}_{4,1}$ be an object point. The (unknown) transformed point can be written as

$$\underline{\boldsymbol{X}}' \;=\; \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}. \tag{2.32}$$

Let $\boldsymbol{x} \in \boldsymbol{\mathcal{G}}_{2,1}$ be an image point. The projective reconstruction of the ray based on this point can be written as

$$\boldsymbol{L}_x \;=\; \mathbf{e} \wedge \boldsymbol{O} \wedge \boldsymbol{x} \quad \in \boldsymbol{\mathcal{G}}_{4,1}. \tag{2.33}$$

The vector $\boldsymbol{O} \in \boldsymbol{\mathcal{G}}_{3,1}$ denotes the optical center of the camera. It leads to a Plücker representation of a line given with its direction and moment. By using the commutator product ($\underline{\times}$) we can now compare the transformed object point with the 3D projection ray. In [29] we have further shown, that this is a well suited constraint equation since it expresses a 3D distance measure between the line and the point in 3D, as shown in figure 2.3. The constraint equation for pose estimation from 2D-3D point correspondences can now be expressed in the following way:

$$\lambda\Big(\;\underbrace{\underbrace{(\boldsymbol{M}\;\underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\;\widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\;\underline{\times}\;\underbrace{\mathbf{e}\wedge(\;\underbrace{\boldsymbol{O}}_{\text{optical center}}\;\wedge\;\underbrace{\boldsymbol{x}}_{\text{image point}}\;)}_{\substack{\text{projection ray,}\\ \text{reconstructed from the image point}}}}_{\substack{\text{collinearity of the transformed object}\\ \text{point with the reconstructed line}}}\;\Big)\cdot\mathbf{e}_+ \;=\; 0. \tag{2.34}$$

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

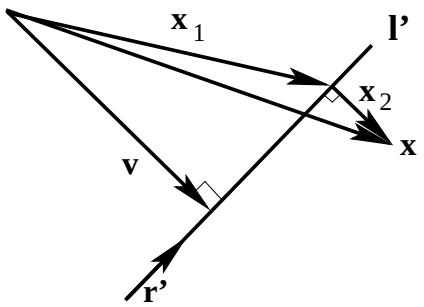

Fig. 2.3: The comparison of the 3D line with the 3D point leads to the 3D perpendicular error vector.

2.3.1 Numerical estimation of pose parameters

In this section we explain our approach for numerical pose estimation. Since the unknown rigid motion is given as exponential function it can not directly be solved in a fast way. Therefore, we linearize the equations (which means we go to the Lie algebra) and iterate the solutions. This results in a gradient descent method. The Euclidean transformation of a point $\underline{\boldsymbol{X}}$ caused by the motor $\boldsymbol{M}$ is approximated in the following way:

$$\begin{aligned} \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right)\underline{\boldsymbol{X}}\exp\left(\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right) \\ &\approx (1-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'}))\underline{\boldsymbol{X}}(1+\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})) \\ &\approx \boldsymbol{E}+\mathbf{e}(\boldsymbol{x}-\theta(\boldsymbol{l'}\cdot\boldsymbol{x})-\theta\boldsymbol{m'}). \end{aligned} \tag{2.35}$$

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

$$\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \approx \boldsymbol{E}+\mathbf{e}(\boldsymbol{x}-\boldsymbol{l}\cdot\boldsymbol{x}-\boldsymbol{m}). \tag{2.36}$$

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

$$\begin{aligned} 0 &= \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \times \underline{\boldsymbol{L}} \\ \Leftrightarrow 0 &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right)\underline{\boldsymbol{X}}\exp\left(\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right) \times \underline{\boldsymbol{L}} \\ \Leftarrow\approx\Rightarrow 0 &= (\boldsymbol{E}+\mathbf{e}(\boldsymbol{x}-\boldsymbol{l}\cdot\boldsymbol{x}-\boldsymbol{m}))\times \underline{\boldsymbol{L}} \\ \Leftrightarrow 0 &= \lambda(\boldsymbol{E}+\mathbf{e}(\boldsymbol{x}-\boldsymbol{l}\cdot\boldsymbol{x}-\boldsymbol{m}))\times \underline{\boldsymbol{L}}. \end{aligned} \tag{2.37}$$

Because of the approximation ($\Leftarrow\approx\Rightarrow$) the unknown motion parameters $\boldsymbol{l}$ and $\boldsymbol{m}$ are linear. This equation contains six unknown parameters of the

rigid body motion. These unknowns are the unknown twist parameters. The linear equations can be solved for a set of correspondences by applying e.g. the Householder method. From the solution of the system of equations, the motion parameters $\boldsymbol{R}, \boldsymbol{t}$ can easily be recovered by evaluating $\theta := \|\boldsymbol{l}\|$, $\boldsymbol{l'} := \frac{\boldsymbol{l}}{\theta}$ and $\boldsymbol{m'} := \frac{\boldsymbol{m}}{\theta}$: Separating $\boldsymbol{m'}$ into a part $\boldsymbol{t} \cdot \boldsymbol{l}$ perpendicular to $\boldsymbol{n} = \boldsymbol{l'}^{\star}$ and $\frac{d}{\theta}\boldsymbol{n}$ parallel to $\boldsymbol{n}$,

$$\boldsymbol{m'} = \boldsymbol{t} \cdot \boldsymbol{l'} - \frac{d}{\theta}\boldsymbol{n},$$

leads to

$$\begin{aligned}
\boldsymbol{M} &= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l'} + \mathbf{e}\boldsymbol{m'}\right)\right) \\
&= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l'} + \mathbf{e}(\underbrace{\boldsymbol{t} \cdot \boldsymbol{l'} - \frac{d}{\theta}\boldsymbol{n}}_{\boldsymbol{m'}})\right)\right) \\
&= \exp\left(\frac{\mathbf{e}d\boldsymbol{n}}{2} - \frac{\theta}{2}\left(\boldsymbol{l'} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l'})\right)\right) \\
&= \exp\left(\frac{\mathbf{e}d\boldsymbol{n}}{2}\right)\exp\left(-\frac{\theta}{2}\left(\boldsymbol{l'} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l'})\right)\right) \\
&= \boldsymbol{T}_{d\boldsymbol{n}}\boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}}.
\end{aligned}$$

Note, that $\exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t} \cdot \boldsymbol{l})\right)\right)$ gives a general rotation (a rotation around a line in space) which can be separated in $\boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}}$, see [29] for details. Since now, everything is expressed in terms of rotors and translators, there is no need to estimate the exponential function of a multivector, but just of cosine and sine functions on the real numbers. Therefore, the motor $\boldsymbol{M}$ can be recovered very fast from the parameters $\theta, \boldsymbol{l'}$ and $\boldsymbol{m'}$. In matrix calculus this results in a similar manner on the application of the Rodrigues' formula [25, 10].

Experiments and examples of this approach can be found in [29].

2.4 Pose estimation of free-form contours

So far we have introduced the basic pose estimation algorithm for 2D-3D point correspondences. Now we directly switch to our proposed model for 3D contour representation and present the algorithm for pose estimation of 3D free-form contours.

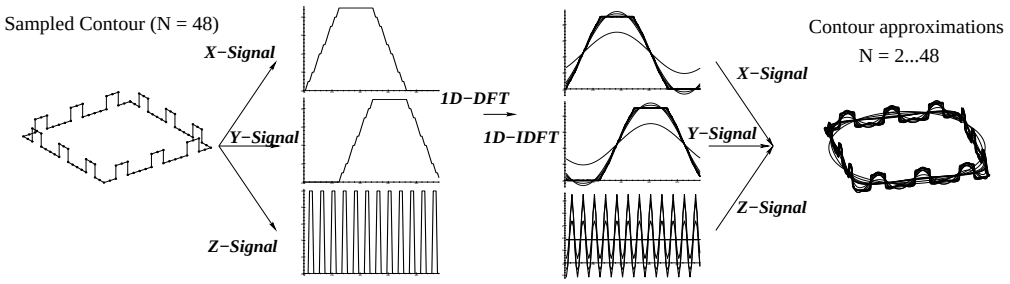

Fig. 2.4: Visualization of contour modeling and approximation by using three 1D Fourier transformations.

Fourier descriptors can be used for object recognition [12] and affine invariant pose estimation [2] of closed contours. They have the advantage of a low-pass object representation (as explained later) and they interpolate sample points along a contour as a continuously differentiable function. During our research we rediscovered the use of Fourier descriptors since they are the generalization of so-called *twist-generated* curves we used to model cycloidal curves (cardioids, nephroids etc.) within the pose problem [29]. We now deal with the representation of 3D free-form contours in order to combine these with our previously introduced point based pose estimation constraints. Since the later introduced pose estimation algorithm for surface models goes back to a contour based one, the recapitulation of our former works on contour based pose estimation is of importance.

The main idea is to interpret a one-parametric 3D closed discrete curve, represented by a finite set of contour points as three separate 1D signals which represent the projections of the curve along the x, y and z axis, respectively. Since the curve is assumed to be closed, the signals are periodic and can be analyzed by applying a 1D discrete Fourier transform (1D-DFT) and an inverse discrete Fourier transform (1D-IDFT) on each signal. Subject to the sampling theorem, this leads to the representation of the curve $C(\phi)$ as

$$C(\phi) = \sum_{m=1}^{3} \sum_{k=-N}^{N} \boldsymbol{p}_k^m \exp(\frac{2\pi k\phi}{2N+1} \boldsymbol{l}_m).$$

In this equation we have replaced the imaginary unit $i = \sqrt{-1}$ with three different rotation planes, represented by the bivectors $\boldsymbol{l_i}$, with $\boldsymbol{l_i}^2 = -1$. The vectors $\boldsymbol{p}_k$ are the phase vectors obtained from the 1D-DFT. Using only a low-index subset of the Fourier coefficients results in a low-pass approximation of the object model which can be used to regularize the pose estimation algorithm. The principle of modeling free-form contours is visualized in figure 2.4.

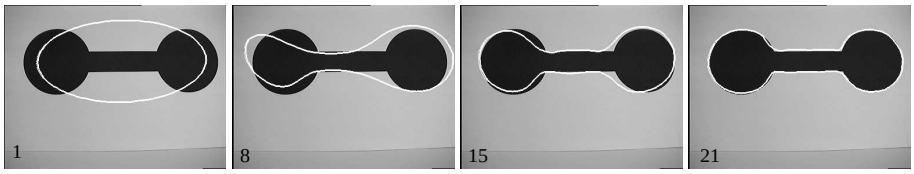

Fig. 2.5: Pose results of the low-pass filtered contour during the iteration.

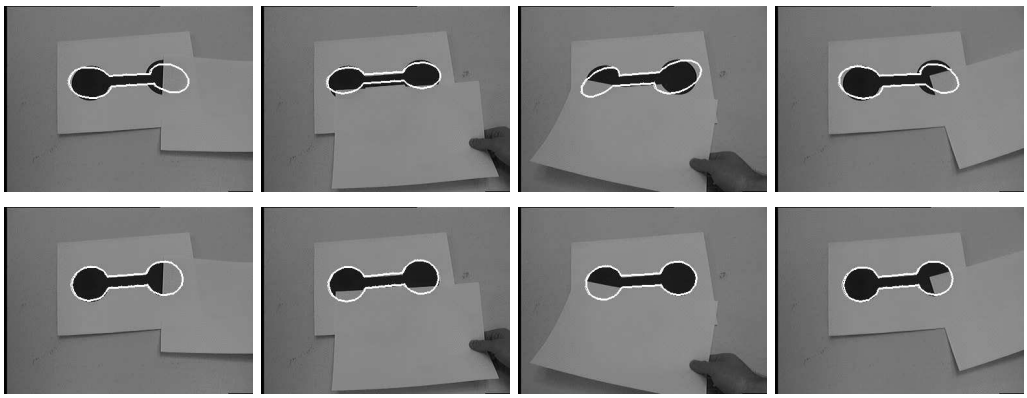

Fig. 2.6: Different pose results for distorted image data. The first row shows results obtained with the non-modified ICP algorithm. The second row shows pose results obtained with the outlier-elimination during the ICP algorithm.

For pose estimation this model is then combined with a version of an ICP-algorithm (iterative closest point) [38]. Using the approach for pose estimation of point-line correspondences, the algorithm for pose estimation of free-form contours consists of iterating the following steps:

```
(a) Reconstruct projection rays from the image points.
(b) Estimate the nearest point of each projection ray to a
    point on the 3D contour.
(c) Estimate the pose of the contour by using this
    correspondence set.
(d) goto (b).
```

The idea is, that all image contour points simultaneously pull on the 3D contour. But whereas ICP algorithms are mostly applied to sets of 2D or 3D points, here it is applied to a trigonometric interpolated function and to 3D projection rays, reconstructed from image points. Figure 2.5 shows an example. As can be seen, we refine the pose results by using a low-pass approximation for pose estimation and by adding successively higher

frequencies during the iteration. This is basically a multi-resolution method and helps to avoid local minima during the iteration.

The robustness of the algorithm with respect to distorted image data is shown in figure 2.6. In this image sequence the image contour is distorted by covering parts of the contour with a white paper. This leads to minor or major errors during the contour extraction in the image. The first row of figure 2.6 shows the results obtained with a non-modified ICP-algorithm. Since we have already clarified that the constraint equations express a geometric distance measure in the 3D space, it is easy to detect outliers and implement an algorithm which automatically detects outliers and eliminates their equations. Some results of the modified algorithm are shown in the second row of figure 2.6. We call this procedure the *outlier-elimination* method. As can be seen, the obtained results are much better. But indeed, these examples give just a guess about the stability of the proposed method. It is not possible to compensate totally wrong extracted contours or too much missing information. Further experiments of this approach can also be found in [30].

3. POSE ESTIMATION OF FREE-FORM SURFACE MODELS

In this section we present our extensions: We start with the representation of free-form surfaces and will then explain the basic pose estimation algorithm.

3.1 Surface representation

We are concerned with the formalization of free-form surfaces in the framework of the 2D Fourier transformation. This enables us to gain a low-pass object description and therefore to regularize the estimation. Then we can refine the object model during iteration steps. Hence the multi-scale object representation can be adapted to its inherent geometric complexity. We assume a two-parametric surface [6] of the form

$$F(\phi_1, \phi_2) \;=\; \sum_{i=1}^{3} f^i(\phi_1, \phi_2)\mathbf{e}_i.$$

This means, we have three 2D functions $f^i(\phi_1, \phi_2) : \mathbb{R}^2 \rightarrow \mathbb{R}$ acting on the different Euclidean base vectors $\mathbf{e}_i$ $(i = 1, \ldots, 3)$. The idea behind a two-parametric surface is to assume two independent parameters ϕ_1 and ϕ_2 to sample a 2D surface in 3D space. Projecting this function along $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, it leads to the three 2D functions $f^i(\phi_1, \phi_2)$. In fact, we are using a mesh model of the object [6]. For a discrete number of sampled points, $f^i_{n_1,n_2}$, $(n_1 \in [-N_1, N_1]; n_2 \in [-N_2, N_2];\ N_1, N_2 \in \mathbb{N},\ i = 1, \ldots, 3)$ on the surface, we can now interpolate the surface by using a 2D discrete Fourier transform (2D-DFT) and then apply an inverse 2D discrete Fourier transform (2D-IDFT) for each base vector separately. Subject to the sampling theorem, the surface can therefore be written as a series expansion which appears in geometric algebra as

$$\begin{aligned} F(\phi_1, \phi_2) &= \sum_{i=1}^{3} \sum_{k_1=-N_1}^{N_1} \sum_{k_2=-N_2}^{N_2} \boldsymbol{p}^i_{k_1,k_2} \exp\left(\frac{2\pi k_1 \phi_1}{2N_1+1}\,\boldsymbol{l_i}\right) \exp\left(\frac{2\pi k_2 \phi_2}{2N_2+1}\,\boldsymbol{l_i}\right) \\ &= \sum_{i=1}^{3} \sum_{k_1=-N_1}^{N_1} \sum_{k_2=-N_2}^{N_2} \boldsymbol{R}^{k_1,\phi_1}_{1,i}\boldsymbol{R}^{k_2,\phi_2}_{2,i}\,\boldsymbol{p}^i_{k_1,k_2}\,\widetilde{\boldsymbol{R}^{k_2,\phi_2}_{2,i}}\widetilde{\boldsymbol{R}^{k_1,\phi_1}_{1,i}}. \end{aligned}$$

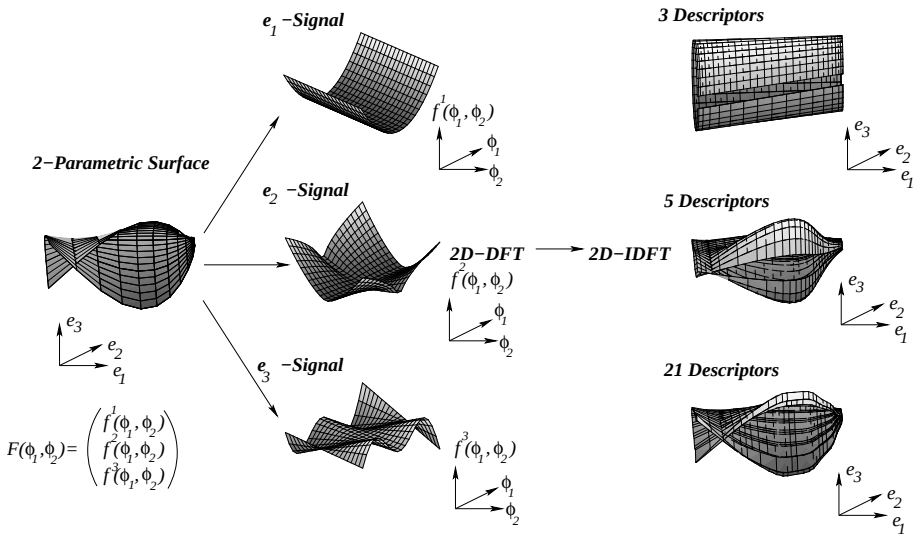

Fig. 3.1: Visualization of surface modeling and approximation by using three 2D Fourier transformations.

The complex Fourier coefficients are contained in the vectors $\boldsymbol{p}^i_{k_1,k_2}$ that lie in the plane spanned by $\boldsymbol{l_i}$. We call them phase vectors. These vectors can be obtained by a 2D-DFT of the sample points $f^i_{n_1,n_2}$ on the surface,

$$\boldsymbol{p}^i_{k_1,k_2} = \frac{1}{(2N_1+1)(2N_2+1)} \sum_{n_1=-N_1}^{N_1} \sum_{n_2=-N_2}^{N_2} f^i_{n_1,n_2} \exp\left(-\frac{2\pi k_1 n_1}{2N_1+1}\boldsymbol{l_i}\right) \exp\left(-\frac{2\pi k_2 n_2}{2N_2+1}\boldsymbol{l_i}\right)\mathbf{e}_i.$$

This is visualized in figure 3.1 as extension to the 1D case of figure 2.4: a two-parametric surface can be interpreted as three separate 2D signals interpolated and approximated by using three 2D-DFTs and 2D-IDFTs. Figure 3.2 shows approximation levels of a car model.

3.2 The algorithm for silhouette based pose estimation of free-form surfaces

We now continue with the algorithm for silhouette based pose estimation of surface models.

In our scenario, we assume to have extracted the silhouette of an object in an image. In the experiments this is simply done by using a thresholded color interval and by smoothing the resulting binary image with morphological operators.

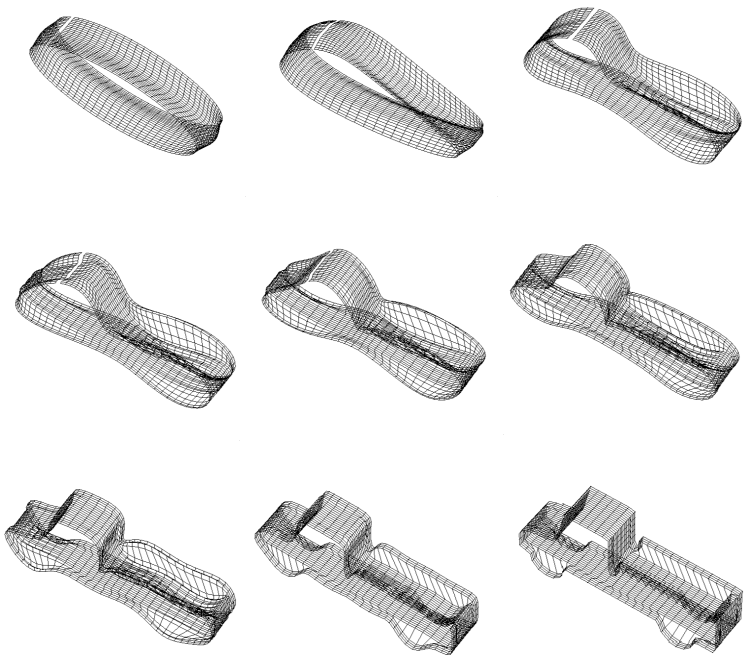

Fig. 3.2: Low-pass approximations of an example object model.

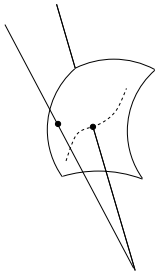

Fig. 3.3: A main problem during pose estimation of surface models: There is need to express tangentiality between the surface and the reconstructed projection rays. Pure intersection is not sufficient for pose estimation.

There is a problem to be considered: It is not useful to express an intersection constraint between the reconstructed projection rays and the surface model. This is visualized in figure 3.3: Postulating the intersection of rays with the surface leads to the effect, that the object is moved directly in front of the camera. Then every reconstructed ray intersects the surface and the constraint is trivially fulfilled. Therefore there is need to express tangential-

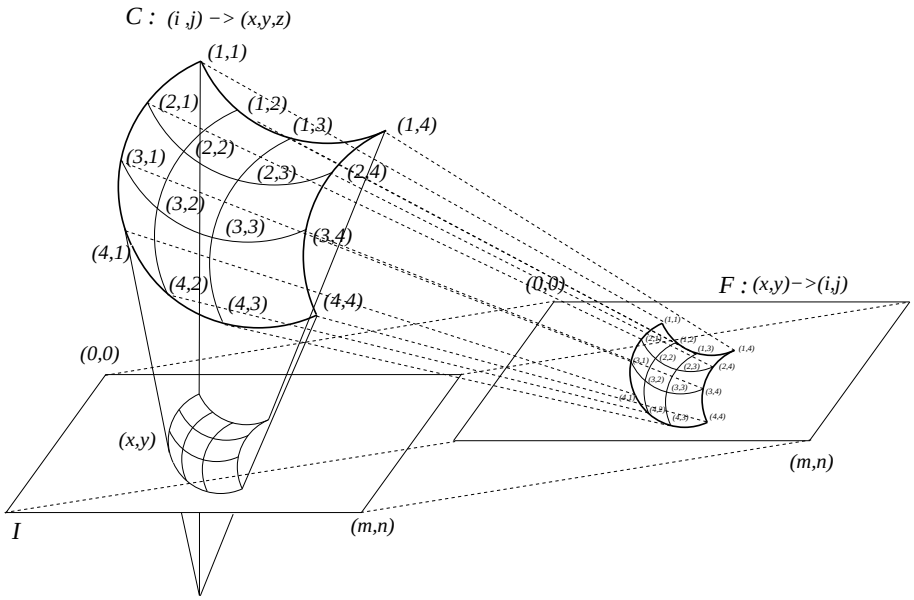

Fig. 3.4: To determine the 3D position of a 2D image node, a field F is used, which stores the relation of image points to the 3D mesh. $C(F(x, y))$ gives the 3D coordinates of a node (x,y) on the image.

ity between the surface and the reconstructed projection rays and there is need to express a distance measure within our description.

To solve this problem we propose to get a contour model from the surface model which is tangential with respect to the camera coordinate system. To compare points on the image silhouette with the surface model, the idea is to work with those points on the surface model which lie on the outline of a 2D projection of the object. This means we are considering the 3D silhouette of the surface model with respect to the camera. To obtain this, we project the 3D surface on a virtual image. Then the contour is calculated and from the image contour the 3D silhouette of the surface model is reconstructed. To obtain the 3D silhouette, there is need to get from the image of a node point to its 3D value. This is done with the help of an additional field F: The basic principle is visualized in figure 3.4: First we assume a contour $C(i, j) \rightarrow (x, y, z)$ which gives the 3D coordinates of the surface node for the two time parameters (i, j). This contour model is projected with the projection matrix in a virtual image, I. The model is projected with connecting line segments between the surface nodes and the nodes themselves are projected in another gray-scale value. This is shown in the close-ups of figure 3.5. Then we reserve a 2D field F with the size of the image (n, m) and save in the field for every node on its pixel position (x, y) the time parameters (i, j) of the surface. This

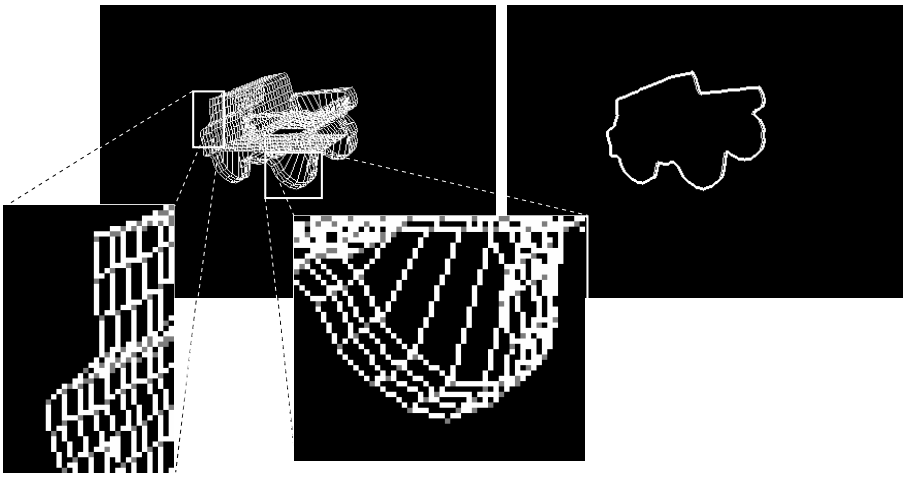

Fig. 3.5: Left: The surface model projected on a virtual image. Right: The estimated 3D silhouette of the surface model, back projected in an image.

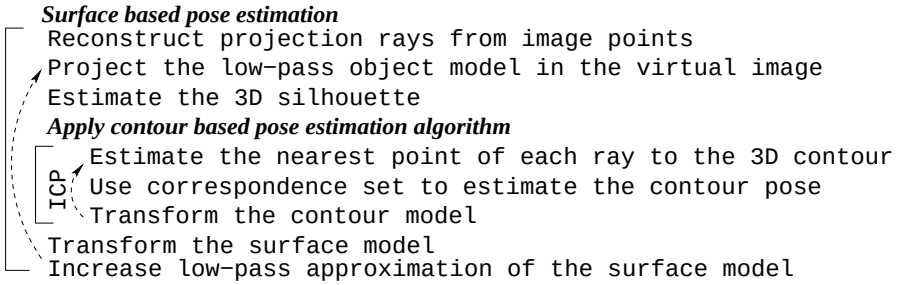

Fig. 3.6: The algorithm for pose estimation of surface models.

means, that we can detect the 3D surface point for a given surface node on the image with the help of the field F and the function C, since $C(F(x, y))$ gives the 3D coordinates of the node image point, (x, y). To gain the 3D silhouette points we use a contour algorithm which follows the image of the mesh model by a recursive procedure. Then the nodes of the mesh model are collected (this is easy, since they are projected in another gray-scale value than the connecting line segments) from which the corresponding 3D-outline is calculated with the help of F. This is visualized in figures 3.4 and 3.5. The contour model is then applied on our previously introduced contour based pose estimation algorithm. Since the aspects of the surface model are changing during the ICP-cycles, a new silhouette will be estimated after each cycle to deal with occlusions within the surface model. The algorithm for pose

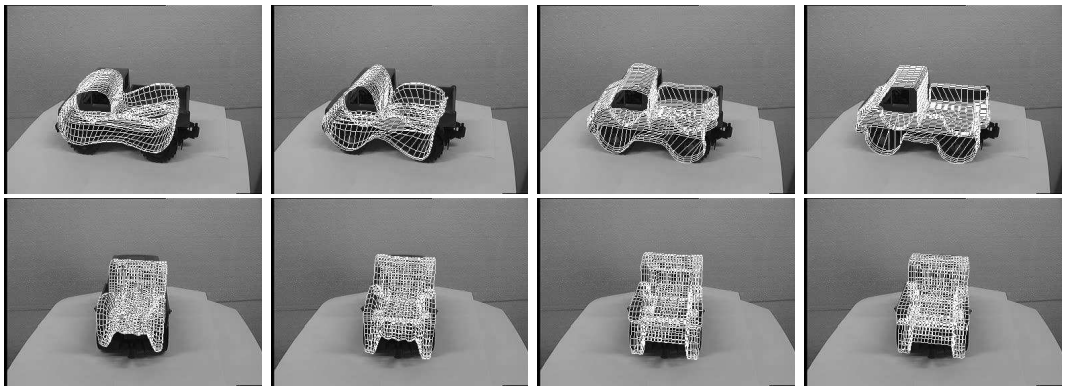

Fig. 3.7: Pose results of the low-pass contours during the ICP cycle.

estimation of surface models is summarized in figure 3.6. The convergence behavior of the silhouette based pose estimation algorithm is shown in figure 3.7. As can be seen, we refine the pose results by adding successively higher frequencies during the iteration. This is basically a multi-resolution method and helps to avoid getting stuck in local minima during the iteration.

3.3 Experiments on one-component silhouette based pose estimation

This section presents experiments to the previously introduced pose estimation algorithm for free-form surface models.

3.3.1 Turntable experiment of a car model

The first experiment deals with a quantitative error analysis of the surface based pose estimation algorithm. The aim is not only to visualize the pose results, but also to compare the pose results with a ground truth. The experiment is organized as follows:

We put a car model on a turntable and perform a 360 degrees rotation on the turntable. We further have a 3D surface model of the car and a calibrated camera system observing the turntable. The rotation on the turntable corresponds to a 360 degrees rotation along the y-axis in the calibrated camera system. During the image sequence we apply the surface based free-form pose estimation algorithm. Example images (and pose results) of this sequence with extracted image silhouettes are shown in figure 3.8. As can be

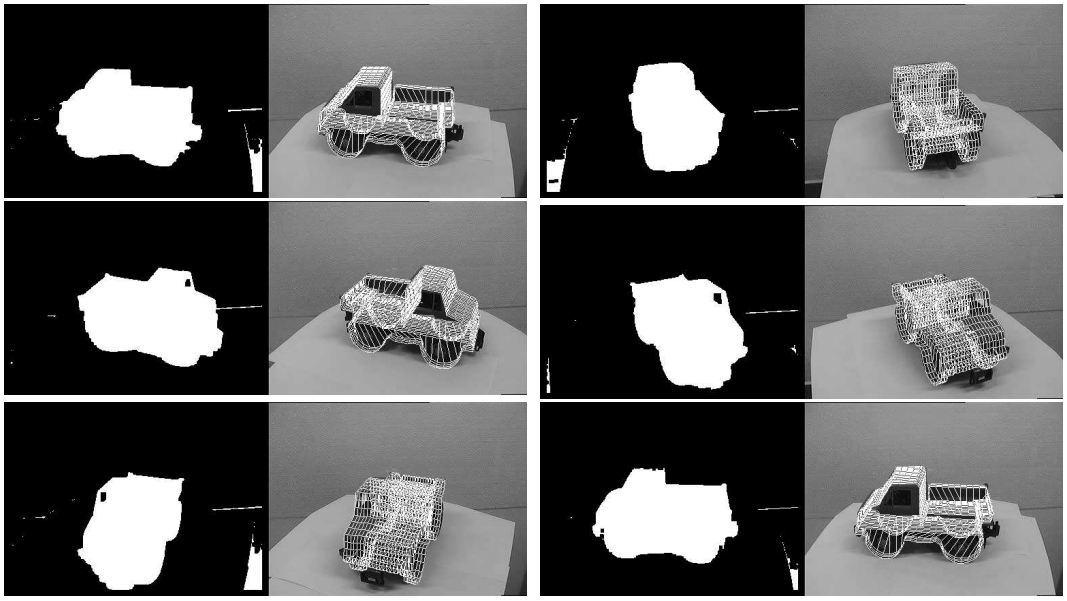

Fig. 3.8: Pose results of the image sequence and the used extracted image silhouettes. Note the extraction errors which occur because of shadows and other fragments.

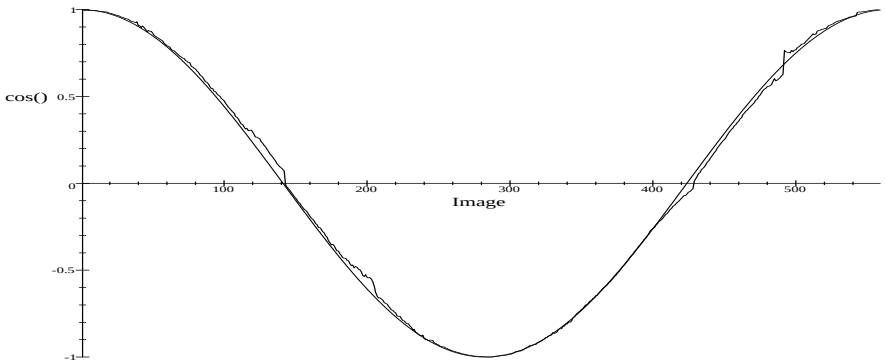

Fig. 3.9: Comparison of the ground truth of the movement with the estimated movement. Since a 360 degrees turn is performed, the cosine of the angles leads to exactly one cosine period.

seen, there exist shadows under the car, which lead to noisy segmented images. Also some parts of the car (e.g. the front bumper or the tow coupling) are not exactly modeled. This means, that there are errors in the image sequence, which are detected during pose estimation.

The aim is now to gain a quantitative error measure of the pose results. To visualize the estimated pose in a quantitative manner, we make use of

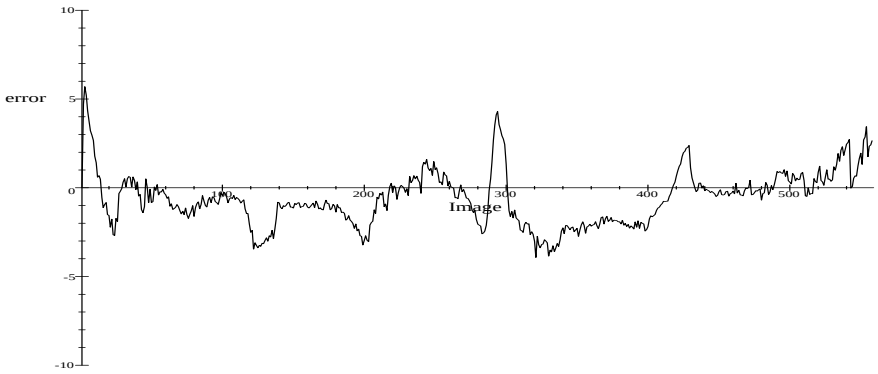

Fig. 3.10: The absolute error between the ground truth and the estimated angle in degrees. The maximum error is 5.73 degrees and average error is 1.29 degrees.

the a priori knowledge of the car movement and compare the angle of the turntable with the estimated y-angle of the car pose. The cosine of these angles is visualized in figure 3.9. As can be seen, the estimated angles are close to the real angles and the ground truth is exactly one cosine period.

Figure 3.10 shows the absolute error between the ground truth and the estimated angle in degrees during the whole image sequence. In the image sequence, the maximum error is 5.73 degrees (at the beginning of the sequence). The average absolute error of the image sequence is 1.29 degrees.

This result must be seen in relation to the noisy extracted image data (figure 3.8) and is therefore a good result. The errors are mainly dependent on the quality of image feature extraction, the calibration quality and the accuracy of the object model.

3.3.2 Stability with respect to image noise

The next experiment enlightens the stability of our algorithm with respect to noisy extracted image data.

The estimation of the silhouette is so far a very simple approach: Just a threshold is assumed for the gray-value boundary and a binary image is extracted from this threshold. In this experiment we analyze the quality of the pose result in relation to this image segmentation threshold. It is clear that too much missing or additional information leads to wrong tracking results, and we give an answer to the sensitivity of our pose algorithm.

The sequence itself contains 300 images and consists of a car on a turntable performing a 180 degrees rotation. We can compare the angle of the turntable as ground truth with the estimated angle of the car pose.

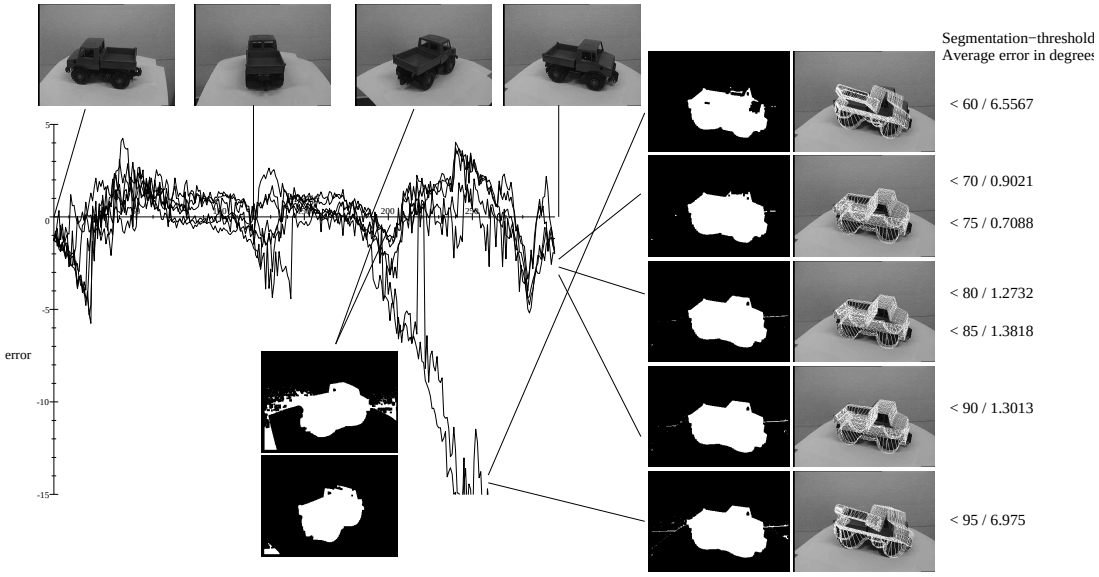

Fig. 3.11: Summary of the stability of the algorithm with respect to different image thresholds. The relative pose errors and example images are shown.

The result is summarized in figure 3.11 which will be explained in more detail:

The first four images show examples of the car during the 180 degrees rotation to visualize the test scenario. The diagram visualizes the error between the ground truth and the estimated angles in degrees during the sequence. The ten images on the right present the silhouette and the pose result of the last image, to show whether the tracking was successful or not. The last column shows the used segmentation threshold for the gray-scale image in the interval $[0, \ldots, 255]$ and the average error in degrees. The extracted silhouettes visualize the difference during image feature extraction.

As can bee seen from the diagram, for 200 images the object is tracked in a tolerable area with a pose quality of $[-5, \ldots, 5]$ degrees. But for the extreme threshold boundaries (< 60 or > 95) the tracking starts to fail. The reason is shown in the remaining two silhouettes: Too much additional and missing information occurs, which can not be compensated from the algorithm. The best result was achieved with a threshold parameter of 75 leading to an average error of 0.7088 degrees. Note, that the experiments are performed with an image resolution of 384×288 pixels and the object is located approximately one meter in front of the camera. Furthermore a full object model description is used with changing aspects of the object during the image sequence. This is a higher complexity compared to often used

constant aspects during pose estimation.

The threshold-value is an additional parameter within the algorithm which must be chosen dependently on the scenario. In this experiment we analyzed the stability of the algorithm with respect to this parameter. It is clear, that too much missing and additional information leads to instable behavior, but the main point is to gain a stable behavior of the algorithm within these boundary values. Since this is achieved (the tracking was always successful within the boundary values), the algorithm can be called robust with respect to this parameter.

The result of the experiment is that the tracking is successful for threshold boundaries within the gray-scale values $]65, 90[$. The algorithm can be called robust with respect to this parameter since small deviations of the threshold parameter do not lead to instable behavior of the algorithm. The calculated average and minimum errors show the potential of the algorithm.

It is clear, that there exist far better algorithm for boundary extraction [8]. The reason why we choose this image pre-processing algorithm is twofold: On the one hand it is very fast (the image preprocessing takes 12ms) and therefore it is well suited within our real-time system. On the other hand, it is a simple approach which leads to non-optimized results. The challenge is to check the potential of the pose algorithm with such noisy and badly extracted image data.

3.3.3 Motion boundaries for tracking objects

The next experiment deals with motion boundary conditions for successful tracking. This means we are interested in the rotations and translations between two frames which still leads to a stable tracking. For this experiment we assume well extracted image data, see section 3.3.2.

The experiment is organized as follows: We take the scenario with the car model, an extracted silhouette and estimate the pose. In the scenario the camera is located approximately 1 m in front of the object. The object itself has the length and height of 20×12 cm. The images have a resolution of 384×288 pixel. The pose result is taken as ground truth, see e.g. the upper left image in figures 3.12 or 3.13. Then we translate or rotate the object along/around the x, y and z-axes with values between $[-70, \ldots, 70]$ mm and $[-40, \ldots, 40]$ degrees, respectively. This means we perform a controlled motion in space for the virtual object model. After this we estimate the pose again and test whether the pose estimation was successful or not. As result we gain point clouds which are shown in figures 3.12 and 3.13. The images show example motions which still lead to the correct pose.

It is clear, that the success of the tracking is dependent on the object

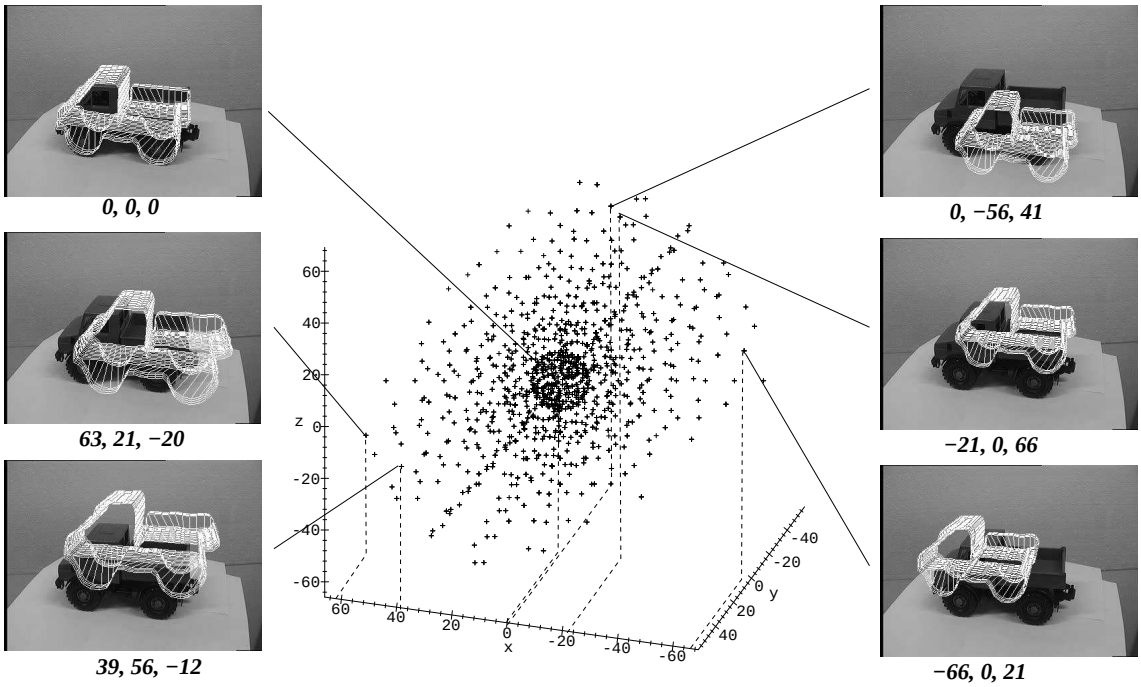

Fig. 3.12: The point cloud shows possible translations in space which still lead to a correct pose. The images on the border visualize the effect of a couple of 3D translations in the image.

size and the camera location and parameters. For this scene the result is the following: We are able to track the object model with a motion of up to 6 cm and $20 - 40$ degrees in space. This corresponds to a deviation of approximately 30 pixel in the image plane. These are the extreme values. To gain a stable tracking we recommend a motion of max. 3 cm and 15 degrees in space, since in this interval the point clouds are more dense. Furthermore can be seen, that the point cloud for successful tracking is indeed no sphere in space but a deformed one along the cone of sight. This is easily understandable since only monocular images are used for pose estimation.

More generally we recommend as rough rule for successful tracking of objects, that the object should move less then one third of the object size to gain stable results.

3.3.4 Convergence behavior

In this part we enlighten the convergence behavior for a few test cases. In this experiment we take the three translational and rotational movements given by the right images of figures 3.12 and 3.13. This means we use the trans-

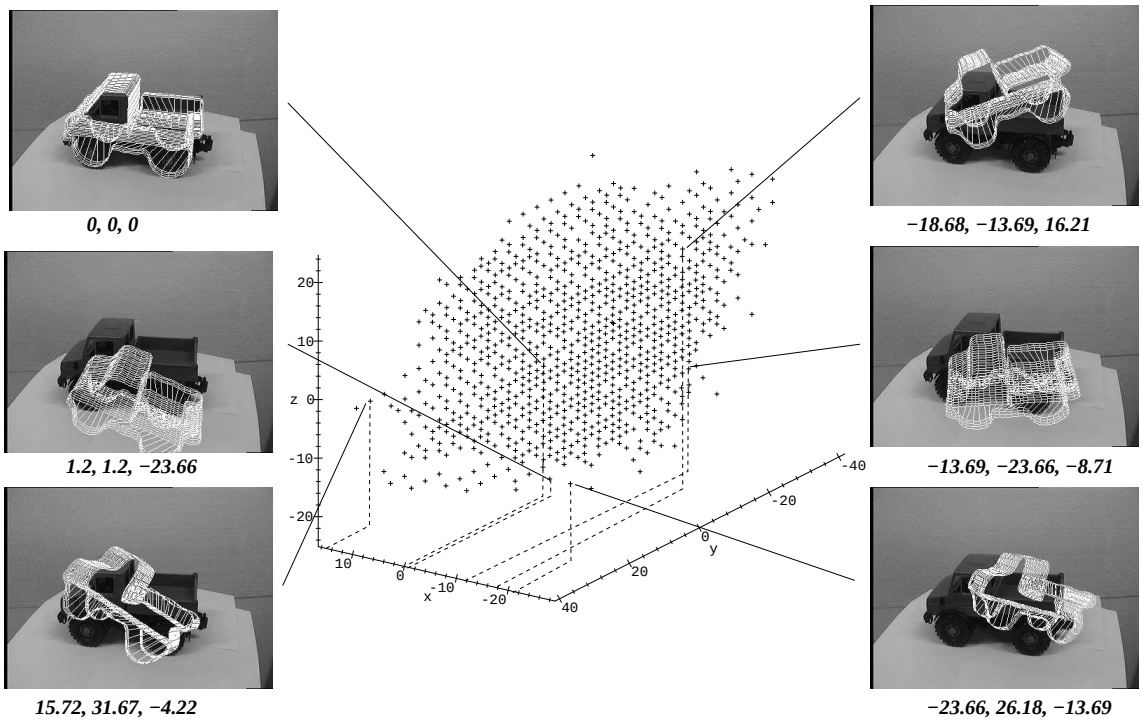

Fig. 3.13: The point cloud shows possible rotations in space which still lead to a correct pose. The images on the border visualize the effect of a couple of 3D rotations in the image.

lational movements $(0, -56, 41)$, $(-21, 0, 66)$ and $(-66, 0, 21)$ (mm) in space and the rotational movements $(-18.68, -13.69, 16.21)$, $(-13.69, -23.66, -8.71)$ and $(-23.66, 26.18, -13.69)$ (degrees) in space. Then we start the pose estimation algorithm and compare the error with respect to the ground truth after each pose estimation. Since we have three nested loops (the pose estimation, the silhouette based pose estimation and the surface based pose estimation) we estimate between 80 and 200 poses till the result converges. For comparison purposes, the nested loops are set to constant number of iterations. In this case we use 5 iterations for each pose, 6 iterations for each ICP-cycle and 10 iterations for each contour. The algorithm is stopped for an error below 1 mm in space. The convergence behavior is shown in figure 3.14.

Three results can be seen: Firstly, the diagrams describe step-functions. The size of the steps is 5 which shows, that the number of iterations during pose estimation (the most inner loop) can be reduced significantly. This is in consistency with our experiments of the convergence behavior of the pure point-based pose estimation algorithm, where we showed that there are not

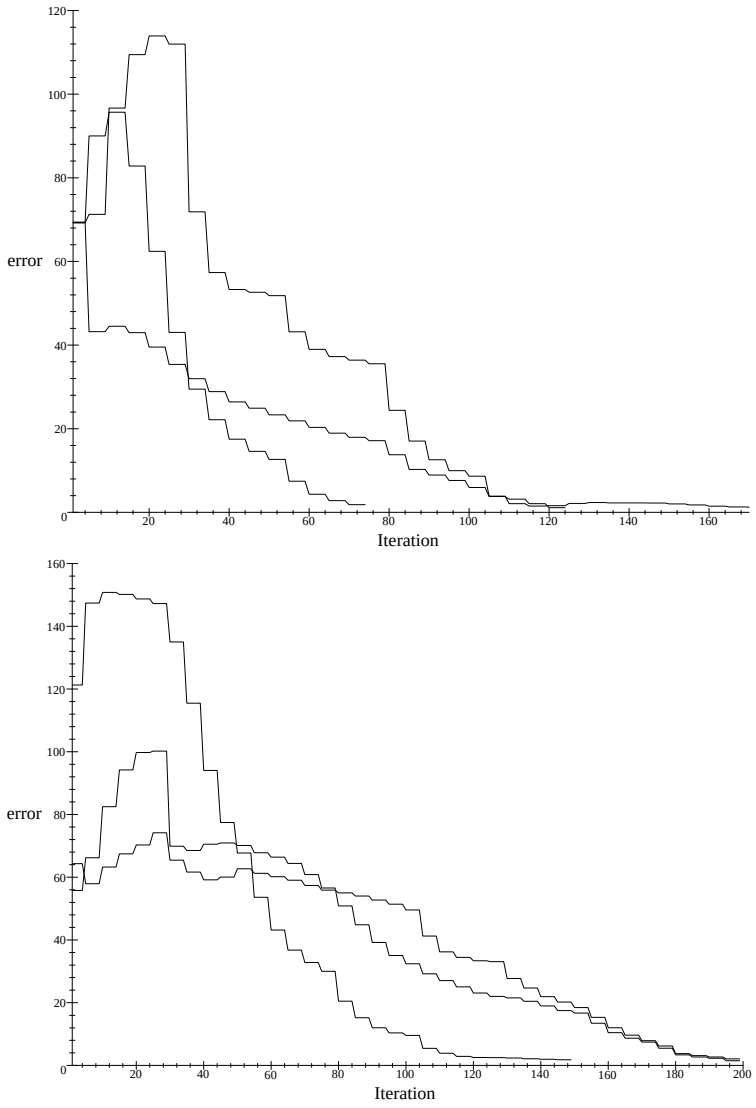

Fig. 3.14: Convergence behavior of three translational (top) and three rotational (bottom) movements.

many iterations needed to converge even to large motions [29].

Secondly can be seen that the translational movements converge faster than the rotational movements. The reason is, that the aspects of the object are changing more during rotations than during translations. Therefore the repeated silhouette estimations are of importance for rotational movements which are on the outer loop of the iteration algorithm. This is the reason, why the convergence is slower for rotational movements than for translational movements.

Thirdly can be seen that the convergence behavior is not monotonous. The reason is twofold: firstly does the use of the low-pass information change the size of the object model since higher frequency parts of the car are neglected. Furthermore is the object moved in space to fit the contour. This results in poses where the object is moved closer to the camera till it is fitted to the object by moving it after-wards away from the camera.

4. EXTENSIONS ON POSE ESTIMATION OF FREE-FORM OBJECTS

This section deals with extensions of our previously introduced approach for one-component silhouette based pose estimation of free-form surface models: We start with multiple component silhouette based pose estimation of free-form surface models. This means we introduce an algorithm for pose estimation of objects which are represented by including free-form surface patches. The second extension concerns the combined use of surface and contour information as object representation. This means, we explain how to couple entities of different dimension during pose estimation to stabilize the pose results. We call the approach *multiple component mixed-mode pose estimation*.

4.1 Dealing with multiple free-form surface patches

This part presents an extension of our approach for surface based free-form pose estimation to multiple surface patches. The reason is that several objects can be represented by their included free-form parts more easily. Assume for example a tea pot. A tea pot (see e.g. figure 4.1) consists of a handle, a container and a spout.

To deal with these multiple surface patches simultaneously as well as with occlusions or partial occlusions of the patches requires a modification of the present free-form pose estimation algorithm.

The main point is that we still work with one virtual 3D silhouette of the object which is now generated from the including free-form patches. This requires a modification of the 3D reconstruction algorithm from the virtual image, but the basic pose estimation algorithm is the same. It is still a contour based pose estimation algorithm including our outlier elimination method within the ICP-algorithm:

We assume an extracted image silhouette and start with the reconstruction of the image contour points to 3D projection rays. This reconstruction

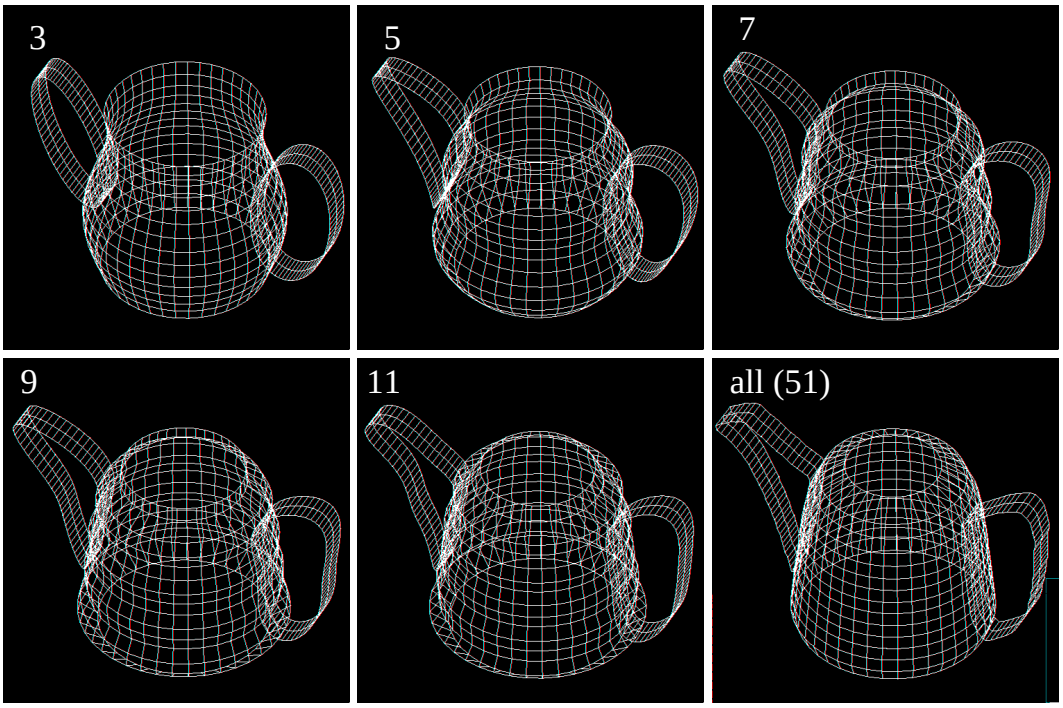

Fig. 4.1: Low pass approximations of an object model containing multiple surface patches.

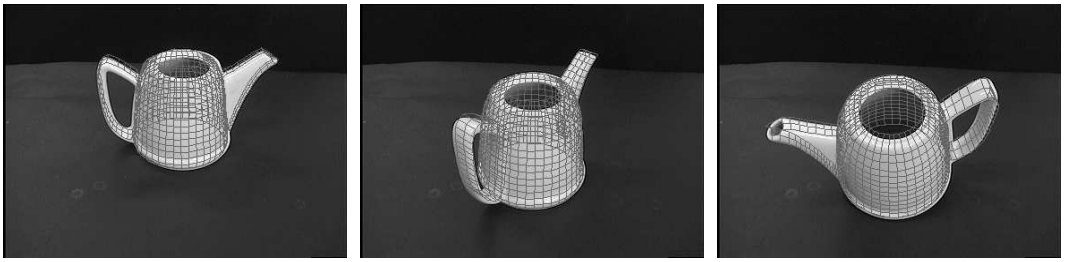

Fig. 4.2: First pose results of the tea pot. The projected object model visualizes the quality of the 3D pose.

is only estimated once for each image. Then the parts of the object model are projected in a virtual image. Since we assume the surface parts as rigidly coupled, we extract and reconstruct one 3D silhouette of the surface model. To obtain this, we use in principle the approach presented in section 3.2, but the function F, which provides us with the reconstruction information, is now extended from $F(x,y) -> (i,j)$ to $F(x,y) -> (k,i,j)$, where k gives the index number of the corresponding contour, see figure 3.4.

Then we apply the 3D contour on our contour based pose estimation

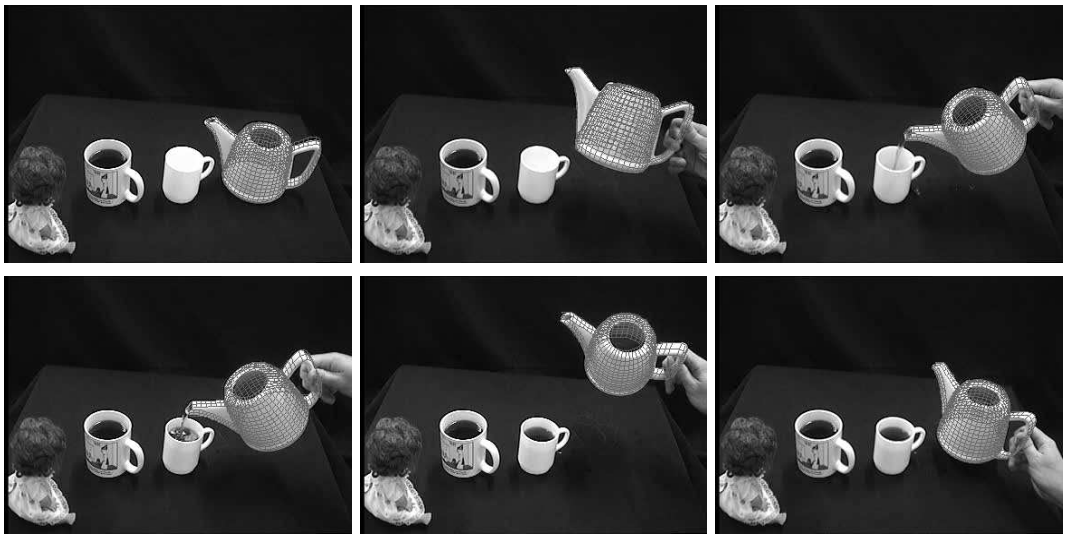

Fig. 4.3: Example images of the tracked tea pot. The hand grasping the tea pot leads to outliers during the image silhouette extraction which are detected and eliminated during the pose estimation.

algorithm, which contains an ICP-algorithm and our gradient descent method for pose estimation. We then transform the surface model with the pose calculated from the contour based pose estimation algorithm and increase the low-pass approximation of each surface patch. Since the aspect of the object model can change after the iterated rigid transformations, we generate a new 3D silhouette: The algorithm continues with a new projection of the object model in a virtual image and the loop repeats till the algorithm converges.

Figure 4.1 shows low-pass examples of our used example object model, a tea pot. Figure 4.2 shows first pose results of the object model. It can be seen that partially occluded surface parts are no problem for the algorithm.

Figure 4.3 shows further example images during an other image sequence containing 350 images. The main aspect of this image sequence is to show that our algorithm is even able to deal with outliers during image processing which are caused by the human hand grasping the tea pot. This is done by our outlier elimination method as described in section 2.4.

4.2 Combining contour and surface patches

So far only the observed contour of the object model is used for pose estimation. Though the quality of the results is already convincing, there is the problem of loosing the *internal* information during the pose estimation.

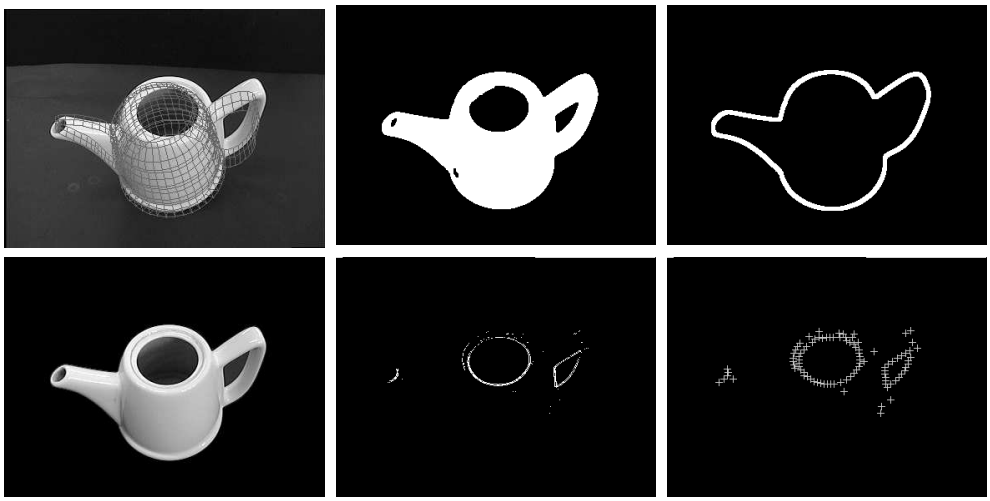

Fig. 4.4: Image processing steps for gaining internal object features: Background subtraction and internal edge detection.

We now present a mixed-mode approach which applies additional edge information on the silhouette based pose estimation. We call additional edges, which are not on the outline of the surface contour with respect to the camera, 'internal' edges, since they are inside the boundary contour in the image. Depending on the object, they can be easily obtainable features which we want to use as additional information to stabilize the pose result. This means to extend the assumed model from one 3D component to multiple components of different dimension.

Roughly speaking this means that we are again applying our silhouette based pose estimation algorithm but extend the generated system of equations with such equations that are gained through the use of edge information within the surface model. This requires an extension of our image processing steps and the used object representation:

4.2.1 Extended image processing

Sofia we use a color-threshold to gain a color blob and use a contour search algorithm to gain the silhouette of the observed object in the image. To gain additional internal edge information we proceed as follows:

```
1. Subtract back-ground from the object
2. Apply a Laplace filter
3. Subtract the contour from the filtered image
```

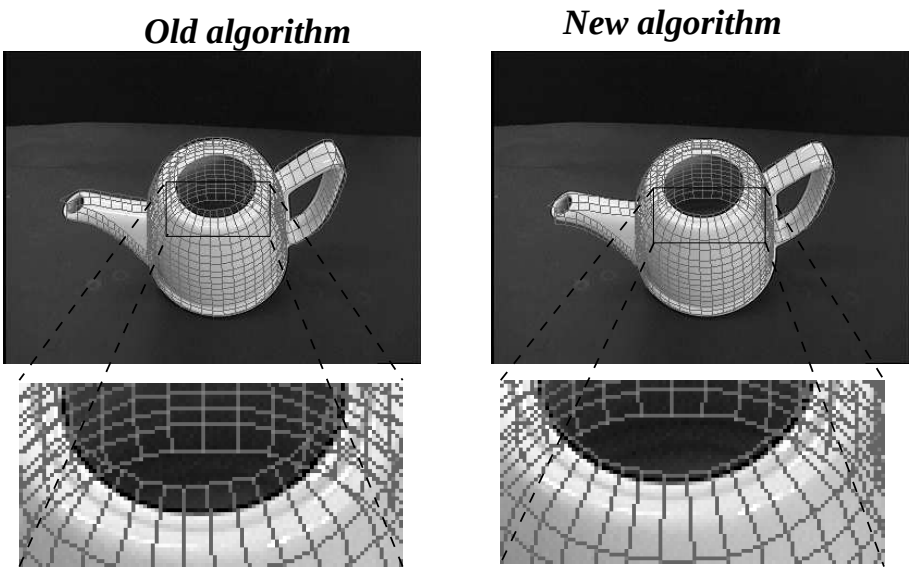

Fig. 4.5: Comparison of pose results of the pure silhouette based pose estimation algorithm (left) and the modified one (right) which uses additional internal edge information.

This is visualized in figure 4.4: The upper left image shows the initial situation and the non-matched object model overlaid in the image. From the extracted image color blob (upper middle image) the contour is extracted (upper right image). Then the background is subtracted (lower left image). After this we filter the image. We choose an eight-neighborhood Laplace operator. Then we subtract the contour and gain an edge image as shown in figure 4.4 lower middle. Using a sub-sampling we gain a number of *internal* edges we use for pose estimation as additional information.

4.2.2 Multiple component mixed-mode pose estimation

It is useless to claim incidence of these extracted points with the surface model since the constraints are trivially fulfilled. Therefore we have to add additional edge information to the surface model by combining the surface representation with internal contours representing the object edges (e.g. the latch edges of the tea pot or the opening of the container).

We then apply the pose estimation algorithm. The generated set of equations can now be separated in two parts, those obtained from the silhouette

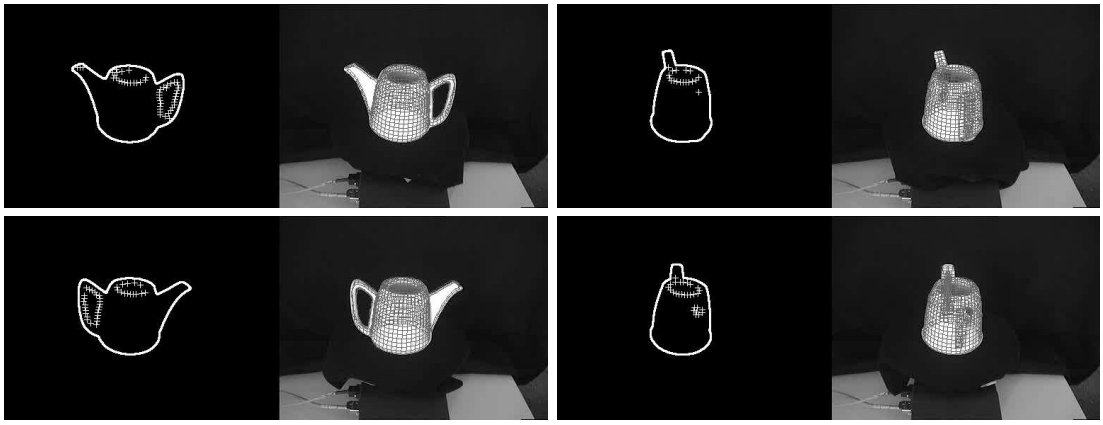

Fig. 4.6: Example images of the tea pot on the turntable. One image shows the result after the image processing, the extracted contour and the used internal feature points. The other image shows the pose result.

and those obtained from the internal feature points:

$$\left(\begin{array}{c}\text{Equations obtained from the silhouette}\\ \text{to the outline of the object model}\\ \hline\hline \text{Equations from internal edges and the}\\ \text{internal edge contour points}\end{array}\right)$$

The unknowns are the six parameters of the rigid motion gained through linearization with respect to its equivalent screw motion. Since both parts can contain larger mismatches or wrong correspondences (see e.g. the falsely extracted edges in figure 4.4), we apply again our outlier elimination method to reduce senseless correspondences.

The effect of the use of such additional information is shown in figure 4.5. As can be seen, the opening contour of the tea pot is forced to the opening in the image and is therefore stabilizing the result.

According to our experiments of the car on the turntable we accomplish a similar one with the tea pot. But now estimate the absolute error in degrees between the ground truth and the real pose with and without using internal image information. The result is shown in figures 4.6 and 4.7: Figure 4.6 shows example images during the image sequence and the image processing results, the used contour and the used internal image features. Figure 4.7 shows the comparison of the estimated pose with the angle of the turntable. The average error of the pure silhouette based pose estimation algorithm is 1.319 degrees and the average error by using additional internal features is 0.949 degrees. In this sequence we use an image resolution of 384×288

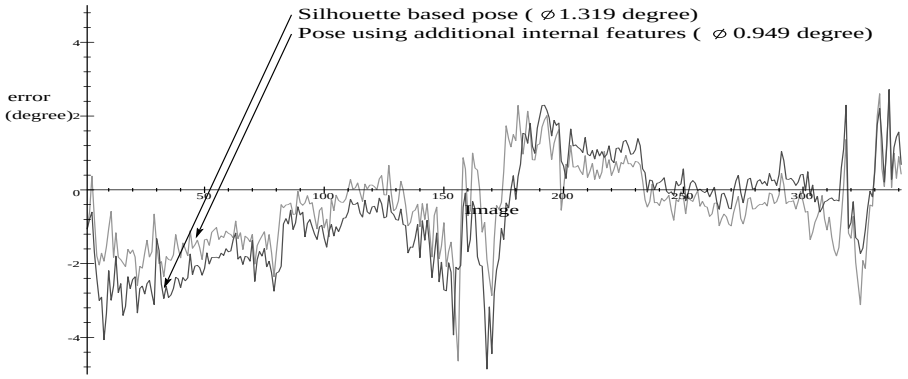

Fig. 4.7: The error of the pure silhouette based pose estimation in comparison with the modified algorithm which uses additional internal object features. The error measure is the absolute angle error in degrees during the turntable image sequence.

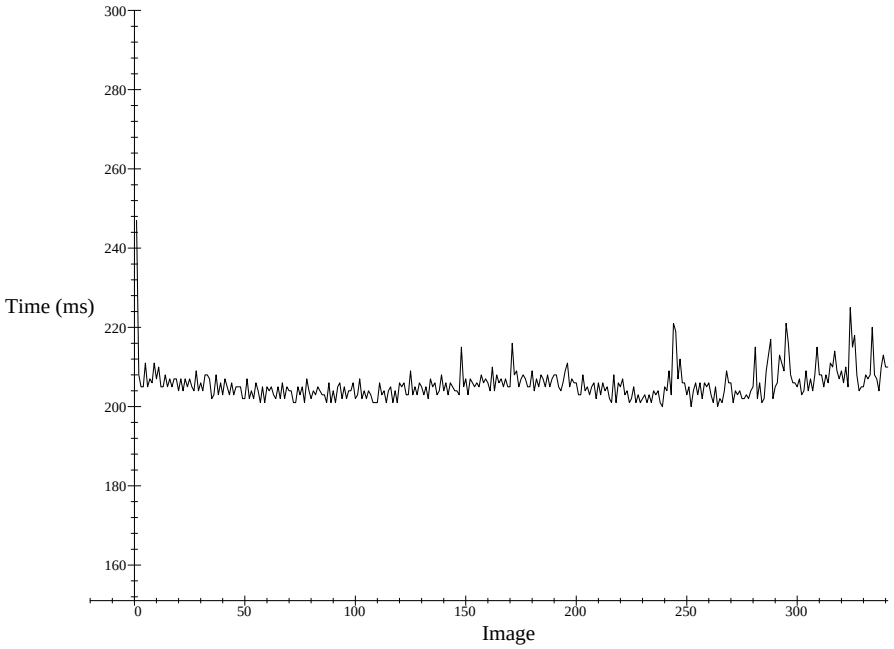

Fig. 4.8: Computing time during tracking the tea pot.

pixels and the object is located approximately 1m in front of the camera. The calibration is performed with a calibration pattern containing 16 manually tracked reference points leading to a calibration error of 0.9 pixels for the reference points. This means that we only work with roughly calibrated cameras and low image resolution which explains the error.

Indeed the comparison holds for just this scenario. For other objects the use of additional internal information might be useless or much more important than the extracted image silhouette. The aim of the experiment is to show that it is possible to extend our silhouette based pose estimation algorithm to scenarios which also use internal edge information of the surface model. To achieve this, we perform an extended image processing to gain internal edge information and we extend our surface model to a combination of free-form surface patches and free-form contour parts.

Figure 4.8 shows the computing time of each image during tracking the tea pot. The average computing time is 205.6 ms and the first image has a computing time of 247 ms to start the tracking.

5. DISCUSSION

In this report we start with a brief introduction to the 2D-3D pose estimation problem. This work is an extension to [29] and therefore many foundations are just briefly explained and scratched. But the basics of Clifford algebras, point based pose estimation and contour based free-form pose estimation are completely given. The main point of this contribution is to present a pose estimation algorithm for free-form surface models which extends our former works. Therefore, we start with an introduction to our surface representation by the use of 2D Fourier descriptors. Then we present the basic pose estimation algorithm which goes back to a contour based one by dealing with just the outline of the object as used object information. This leads to a natural distance measure of tangentiality between the surface and the reconstructed projection rays. In the experiments we present results from different image sequences and analyze the stability with respect to image noise, the used image processing algorithm and the boundaries of the tracking assumption. Furthermore we present extensions to the simultaneous use of multiple surface patches. Furthermore, we show how it is possible to use *internal* contour information to stabilize the pose result. This basically links contour based pose estimation with surface based pose estimation and shows the adaptivity of our chosen approach to arbitrary scenes. Our experiments show the applicability of our chosen approach to different scenes. The algorithms are stable with respect to image or object noise. Furthermore is the present report a natural extension to our former works [29] with all previously developed advantages (e.g. a low-pass representation of objects). Further works will deal with coupling this model to kinematic chains. This means that we are interested in pose estimation of human bodies modeled with free-form surface patches which contain additional joints. There are lots of interesting works to follow and we have not utilized all possibilities and extensions of our chosen approach

BIBLIOGRAPHY

[1] Arbter K. Affininvariante Fourierdeskriptoren ebener Kurven *Technische Universität Hamburg-Harburg*, PhD Thesis, 1990.

[2] Arbter K. and Burkhardt H. Ein Fourier-Verfahren zur Bestimmung von Merkmalen und Schätzung der Lageparameter ebener Raumkurven. *Informationstechnik*, Vol. 33, No. 1, pp.19-26, 1991.

[3] Besl P.J. The free-form surface matching problem. *Machine Vision for Three-Dimensional Scenes, Freemann H. (Ed.)*, pp. 25-71, Academic Press, San Diego, 1990.

[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] CLU Library, A C++ Library for Clifford Algebra. available at http://www.perwass.de/CLU *Lehrstuhl für kognitive Systeme, University Kiel,* 2001.

[6] Campbell R.J. and Flynn P.J. A survey of free-form object representation and recognition techniques. *CVIU: Computer Vision and Image Understanding*, No. 81, pp.166-210, 2001.

[7] O'Connor J.J. and Robertson E.F. Famous Curves Index *http://www-history.mcs.st-andrews.ac.uk/history/Curves/Curves.html*

[8] Cremers D. A variational framework for image segmentation combining motion estimation and shape regularization, *IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, Dyer C. and Perona P. (Eds.), Madison, Wisconsin, Vol. 1, pp. 53-58, 2003.

[9] Drummond T. and Cipolla R. Real-time tracking of multiple articulated structures in multiple views. In *6th European Conference on Computer Vision, ECCV 2000, Dubline, Ireland*, Part II, pp.20-36, 2000.

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

[11] Goddard J.S. Pose and Motion Estimation From Vision Using Dual Quaternion-Based Extended Kalman Filtering. *University of Tennessee, Knoxville*, Ph.D. Thesis, 1997.

[12] Granlund G. Fourier preprocessing for hand print character recognition. *IEEE Transactions on Computers*, Vol. 21, pp. 195-201, 1972.

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

[14] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [33], pp. 3-23, 2001.

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

[16] Hestenes D. and Ziegler R. Projective geometrie with Clifford algebra. *Acta Applicandae Mathematicae*, No.23, pp.25–63, 1991.

[17] Klette R. and Rosenfeld A. Digital Geometry - Geometric Methods for Digital Picture Analysis. *Morgan Kaufmann*, San Francisco, 2004.

[18] 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.

[19] Horaud R., Phong T.Q. and Tao P.D. Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision (IJCV)*, Vol. 15, pp.225–243, 1995.

[20] Kriegman D.J., Vijayakumar B., and Ponce, J. Constraints for recognizing and locating curved 3D objects from monocular image features. In *Proceedings of Computer Vision (ECCV '92), G. Sandini, (Ed.)*, LNCS 588, Springer-Verlag, pp. 829–833, 1992

[21] Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [33], pp. 27-52, 2001.

[22] Lee X. A Visual Dictionary of Special Plane Curves. *http://xahlee.org/SpecialPlaneCurves_dir/specialPlaneCurves.html*

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

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

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

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

[27] Perwass C. Applications of Geometric Algebra in Computer Vision. *Cambridge University*, Ph.D. Thesis, 2000. Available at *http://www.perwass.de*

[28] Perwass C. and Hildenbrand D. Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space. An Introductory Tutorial. *Technical Report 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003.

[29] Rosenhahn B. Pose Estimation Revisited *Technical Report 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003. Available at `http://www.ks.informatik.uni-kiel.de`

[30] Rosenhahn B., Perwass Ch. and Sommer G. Free-form pose estimation by using twist representations *Algorithmica*, Vol. 38, Nr.1, January 2004, in print.

[31] Rosenhahn B., Perwass C. and Sommer G. Pose estimation of free-form surface models. In *Pattern Recognition, 25th DAGM Symposium*, B. Michaelis and G. Krell (Eds.), Springer-Verlag, Berling Heidelberg, LNCS 2781, pp. 574-581.

[32] Selig J.M. Geometric Foundations of Robotics. *World Scientific Publishing*, 2000.

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

[34] Tello R. Fourier descriptors for computer graphics. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 5, pp. 861-865, 1995.

[35] Walker M.W. and Shao L. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, Vol. 54, No. 3, pp.358–367, 1991.

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

[37] Zerroug, M. and Nevatia, R. Pose estimation of multi-part curved objects. *Image Understanding Workshop (IUW)*, pp. 831-835, 1996

[38] Zang Z. Iterative point matching for registration of free-form curves and surfaces. *IJCV: International Journal of Computer Vision*, Vol. 13, No. 2, pp. 119-152, 1999.