INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Pose Estimation of 3D Free-form Contours

Bodo Rosenhahn, Christian Perwass, Gerald Sommer

Bericht Nr. 0207

August 2002

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

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

Pose Estimation of 3D Free-form Contours

Bodo Rosenhahn, Christian Perwass, Gerald Sommer

Bericht Nr. 0207
August 2002

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

ABSTRACT

In this report we discuss the 2D-3D pose estimation problem of 3D free-form contours. In our scenario we observe objects of any 3D shape in an image of a calibrated camera. Pose estimation means to estimate the relative position and orientation (containing a rotation $\mathcal{R}$ and translation $\mathcal{T}$) of the 3D object to the reference camera system. The fusion of modeling free-form contours within the pose estimation problem is achieved by using the conformal geometric algebra. The conformal geometric algebra is a geometric algebra which models entities as stereographic projected entities in an homogeneous model. This leads to a linear description of kinematics on the one hand and projective geometry on the other hand. To model free-form contours in the conformal framework we use twists to model cycloidal curves as twist-depending functions and interpret n-times nested cycloidal curves as functions generated by 3D Fourier descriptors. This means, we use the twist concept to apply a spectral domain representation of 3D contours within the pose estimation problem. We will show that twist representations of objects can numerically efficient and easily be applied to the pose estimation problem. The pose problem itself is formalized as implicit problem and we gain constraint equations, which have to be fulfilled with respect to the unknown rigid body motion. Several experiments visualize the robustness and real-time performance of our algorithms.

CONTENTS

1. INTRODUCTION

This contribution concerns the 2D-3D pose estimation problem of 3D free-form contours. 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 [30, 31], Grimson [18] and others. In their work point correspondences are used. More abstract entities can be found in [25, 50, 26, 7]. Discussed entities are circles, cylinders, kinematic chains or other multi-part curved objects. Works concerning free-form curves can be found in [13, 46]. In their work contour point sets, affine snakes, or active contours are used for visual servoing. An overview of free-form object representations is given in [10]. In this work several mathematical forms are discussed, e.g. parametric forms, algebraic implicit surfaces, superquadrics, generalized cylinders or polygonal meshes.

There exist two main strategies to deal with object models: Firstly, the object can be represented by characteristic object features (like edges or corners, etc.) and then applied to different problems (e.g. pose estimation, object recognition). Secondly, the object can be modeled as itself, e.g. in form of an implicit or parametric surface. The main properties of these strategies are clear: If we assume scenarios containing *easy* objects with easy extractable corner or edge features (e.g. buildings or artificial objects), there is no need to complicate the situation by using full parameterized models. But especially in natural environments with curved shapes and surfaces, feature extraction and matching is a problem. Then there is need to deal with an object as a whole, or as one single entity, respectively. We want to deal with objects of general shape and call them free-form objects as a general class of entities. For a definition of free-form objects, we want to quote Besl [6]:

Definition 1.1 *A free-form surface has a well defined surface that is continuous almost everywhere except at vertices, edges and cusps.*

Sculptures, car bodies, ship hulls, air planes, human faces or organs are typical examples for free-form objects.

The main problem we concern is the algebraic coupling of free-form contours with the pose estimation problem. Therefore we use as link between these different topics the *twist* representation. Twists are representing rigid body motions in the framework of Lie algebras [16]. In this work we will use twists twofold, on the one hand within our pose estimation problem as representations of rigid body motions and on the other hand to model the object contours as orbits generated from a set of parameterized operators for general rotations. This unification of representations enables a compact description of the pose estimation problem for free-form contours in an implicit manner by using constraint equations, which have to be fulfilled.

The ICP (Iterative Closest Point) algorithms are well known for aligning 3D object models. Originally ICP starts with two data sets (of points) and an initial guess for their rigid body motion. Then the transformation is refined by repeatedly generating pairs of corresponding points of the sets and minimizing an error metric. The ICP algorithms are mostly applied on 2D or 3D point sets. Instead, we will later use it for comparison of a trigonometric interpolated function with reconstructed projection rays. Different works concerning ICP algorithms can be found in [43, 12, 24, 51].

To solve the pose estimation problem of free-form contours we will start with the pose estimation problem for entities like points, lines and planes. Then we will continue with cycloidal curves as a special case of algebraic curves. In general a cycloidal curve is generated by a circle rolling on a circle or a line without slipping [28]. We will use a twist representation to model these curves (they are later called 3D 2twist cycloidal curves) and generalize them to 3D $n$twist cycloidal curves. This representation can be used to model a 3D trigonometric interpolated curve of a 3D contour. The representation of an object shape by using twists is compact and transformations of the object can be estimated just by transforming the generators of the entity. Furthermore, instead of estimating the pose for a whole 3D contour, we are able to use a low-pass version of the contour as an approximation, leading to a speed up of the algorithm. As later shown, 3D cycloidal curves are strongly connected to Fourier descriptors [2, 1, 17] as spectral representation of a contour. Fourier descriptors are often used for object recognition but are hard to connect with the 2D-3D pose estimation problem for a full perspective camera model. In this work we overcome this problem by applying Fourier descriptors in a kinematic formalization of the pose problem.

The paper is organized as follows: We will start with our preliminary works in which we generated a set of basis entities which can be used to model objects for pose estimation. Then we continue with cycloidal curves and end up in free-form contours as trigonometric interpolated functions. Aiming a trigonometric interpolation can be interpreted as constraint superposition of

orbits generated by a set of coupled twists we use for our pose estimation scenario. The contribution ends with experiments on pose estimation of free-form contours.

1.1 Preliminary work

Our recent work [39, 40, 41] can be summarized in the scenario of figure 1.1:

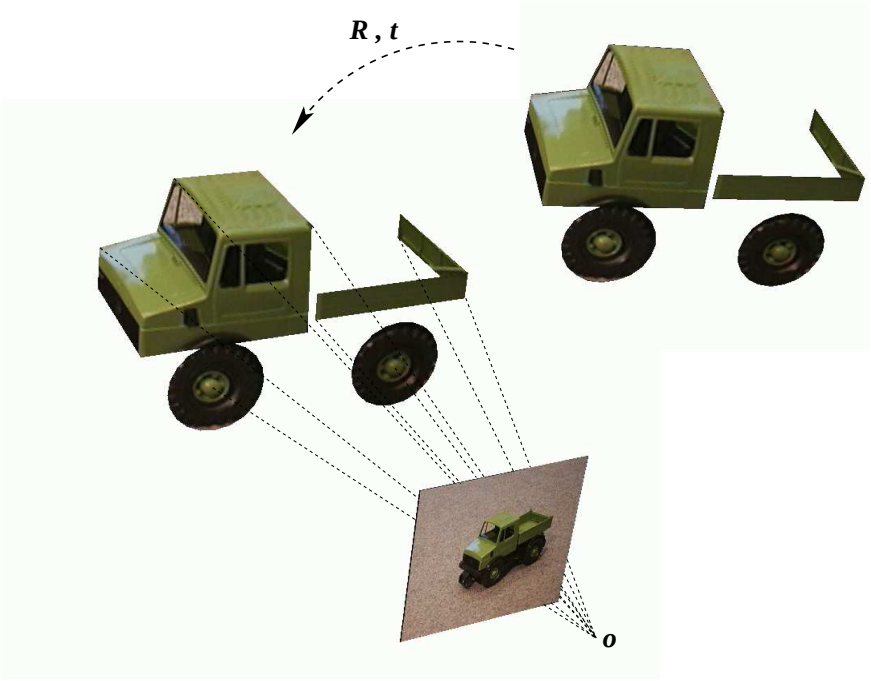

Fig. 1.1: The scenario. The assumptions are the projective camera model, the model of the object (consisting of points, lines, circles and kinematic chains) and corresponding extracted entities on the image plane. The aim is to find the pose ($\boldsymbol{R}$, $\boldsymbol{t}$) of the model, which leads to the best fit of the object with the actually extracted entities.

In these preliminary works, we assume as object features 3D points, 3D lines, 3D spheres, 3D circles or kinematic chain segments of a reference model (see figure 1.2 for an example). Further, we assume extracted corresponding features 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 actually extracted entities. To relate 2D image information to 3D entities we interpret an extracted image entity, resulting from the

perspective projection, as a one dimension higher entity, gained by projective reconstruction from the image entity. This idea will be used to formulate the

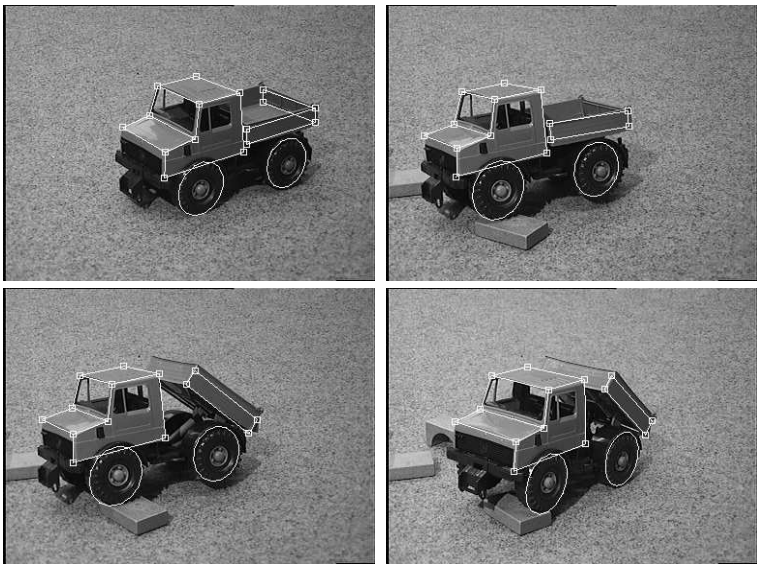

Fig. 1.2: Pose estimation by using different types of correspondences

scenario as a pure kinematic problem. As mentioned before, there exist many scenarios (e.g. in natural environments) in which it is not possible to extract point-like features as corners or wedges. Then there is need to deal e.g. with the silhouette of the object as a whole entity instead of sparse local features on the silhouette. Besides, there exist 3D objects which cannot adequately be represented by primitive object features as points, lines or circles. These are the scenarios we address in this contribution. Additionally we argue that from a statistical point of view, pose estimations of global object descriptions are more accurate and robust than those from a sparse set of local features.

1.2 Algebraic curves

This section gives a brief summary of algebraic curves [28]. There exist many ways to define algebraic curves. They can be defined as parametric or algebraic implicit forms or polynomial equations [10]. For example a conic can be defined as the set of intersection points of two projectively related pencils of lines [21]. It is also possible to define a conic as intersection of a cone with a plane. Parametric, cartesian or polar equations of a curve lead to quite different representations. E.g. the parametric equations of a plane

cardioid is

$$(x, y) \quad = \quad (a(2\cos(t) - \cos(2t)), a(\sin(t) - \sin(2t)), \tag{1.1}$$

a cartesian equation is

$$\left(x^2 + y^2 - 2ax\right)^2 \quad = \quad 4a^2(x^2 + y^2) \tag{1.2}$$

and the polar equation is

$$r \quad = \quad 2a\,(1 + cos(\theta))\,. \tag{1.3}$$

The resulting question is: Which representation of an algebraic curve is well suited within the pose estimation problem? As mentioned before, we want to use concepts, which are already element of the kinematic framework we use for the pose problem. Therefore we prefer to describe algebraic curves as orbits of a twist generated function. The second argument for using twists consists in their compact representation within the pose problem to gain small and easily interpretable equations. More detailed information about algebraic curves can also be found in [11].

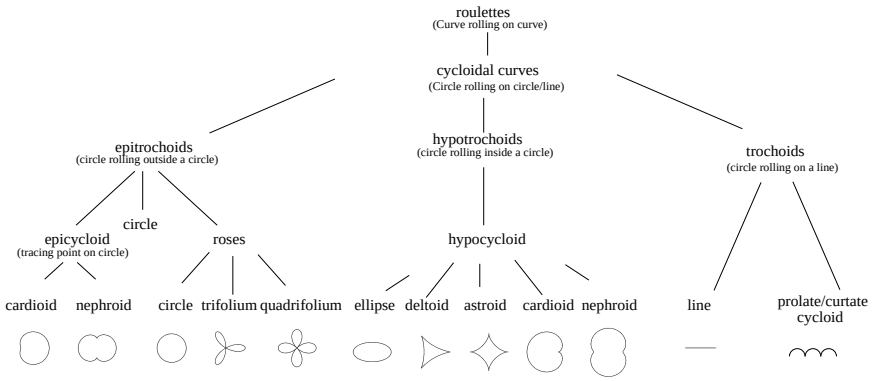

Fig. 1.3: Tree of algebraic curves

Here we will concentrate on a subclass of the *roulettes*, the *cycloidal curves*, which are circles rolling on circles or lines. Figure 1.3 shows a subtree of the family of algebraic curves. Cycloidal curves can be distinguished between epitrochoids, hypotrochoids and trochoids, which split to other subclasses. Figure 1.3 shows examples of these curves.

Since a circle can be interpreted as a point rotating around a line in the space, a circle rolling on a circle is nothing more than a point rotating around two twists in a fixed and dependent manner.

The curves are distinguished by the relative position of the twists with respect to the starting point on the curve and the frequency of the twists. We will now continue to explain some curves more detailed:

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

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

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

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

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

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

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

Indeed it is possible to generalize cycloidal curves to e.g. circles rolling on circles/line, which are again rolling on circles/lines. This generalization of n nested rolling circles is later called $n$twist cycloidal curve. Since these circles can be interpreted as the sum of n *phase* vectors, we will later show the connection of $n$twist cycloidal curves to Fourier descriptors of closed curves, well known from signal theory. Since Fourier descriptors can be used to trigonometrically interpolate functions, we have then the direct link to use cycloidal curves for any free-form contour.

These curves are mostly defined in the 2D plane. For our scenario of pose estimation we will extend these curves to 2D or 3D curves in the 3D space.

2. CURVES IN CONFORMAL GEOMETRIC ALGEBRA

This section concerns the formalization of cycloidal curves in conformal geometric algebra. Geometric algebras are the language we use for our pose problem and the main arguments for using this language in that context are its dense symbolic representation and its coupling of projective and kinematic geometry. One main problem for 2D-3D pose estimation are the involved mathematical spaces which are elements of the stratification hierarchy proposed by Faugeras [14]. On the one hand we are interested to estimate an affine transformation (a rigid body motion) and on the other hand we observe objects in an image and therefore have to deal with projective geometry. To overcome this problem we use a conformal embedding and so we are able to deal with Euclidean, projective and kinematic geometry in one language. We will first introduce the basic notation of conformal geometric algebra and the modeling of entities and their kinematic transformations. We make use of it to model cycloidal curves in the 3D space. Modeling cycloidal curves is also possible in affine geometry, but in the next section we then combine this formalization within our pose estimation problem and then there is need for using conformal geometry: The image entities are projectively reconstructed to, e.g., projection rays and then transformed to conformal lines (Plücker lines [38]). So the projective aspect of the pose problem is transformed to a kinematic one and then combined with the kinematic description of the cycloidal curves.

This section gives only a brief introduction to geometric algebras. The reader should consult [42] for a more detailed introduction.

2.1 Introduction to conformal geometric algebra

In this section we introduce the main properties of the conformal geometric algebra (CGA) [27]. The aim is to clarify the notations. A more detailed introduction concerning geometric algebras can be found in [45].

In general, a geometric algebra $\mathcal{G}_{p,q}$ is a linear space of dimension 2^n,

$n = p + q$, with a subspace structure, called blades, to represent so-called multivectors as higher order algebraic entities in comparison to vectors of a vector space as first order entities. A geometric algebra $\boldsymbol{\mathcal{G}}_{p,q}$ results in a constructive way from a vector space $\mathbb{R}^{p,q}$, endowed with the signature (p, q), $n = p + q$, 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 $\boldsymbol{AB}$ contains an outer ($\wedge$) and an inner ($\cdot$) product, whose roles are to increase or to decrease the order of the algebraic entities, respectively. We will demonstrate this affect in case of two vectors $\boldsymbol{a}$, $\boldsymbol{b} \in \boldsymbol{\mathcal{G}}_{p,q}$:

$$\begin{aligned} \boldsymbol{ab} &= \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b} \\ &= \frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) + \frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}), \end{aligned} \tag{2.1}$$

where $\alpha = \boldsymbol{a} \cdot \boldsymbol{b}$ is a scalar and $\boldsymbol{A} = \boldsymbol{a} \wedge \boldsymbol{b}$ is a bivector, thus,

$$\boldsymbol{ab} = \alpha + \boldsymbol{A} \tag{2.2}$$

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

$$\boldsymbol{M} = \sum_{i=0}^{n} \langle \boldsymbol{M} \rangle_i. \tag{2.3}$$

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 is generated by the outer product of i vectors. This means in general that every i-blade is a multivector of grade i but not vice versa.

For later use we introduce the commutator $\underline{\times}$ and anti-commutator $\overline{\times}$ products, respectively, for any two multivectors,

$$\boldsymbol{AB} = \frac{1}{2}(\boldsymbol{AB} + \boldsymbol{BA}) + \frac{1}{2}(\boldsymbol{AB} - \boldsymbol{BA}) =: \boldsymbol{A} \overline{\times} \boldsymbol{B} + \boldsymbol{A} \underline{\times} \boldsymbol{B}. \tag{2.4}$$

We use the *conformal geometric algebra* [27] to model the geometry of our scenario for free-form pose estimation. 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.

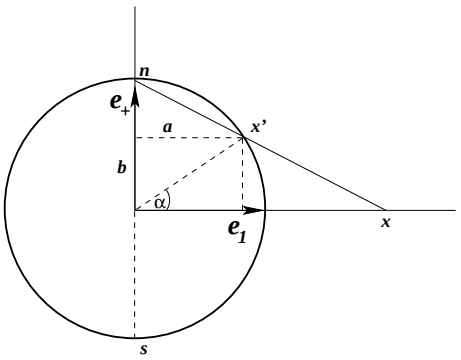

Fig. 2.1: Visualization of a stereographic projection for the 1D case: Points on the lines are projected on the circle. Note that the point at infinity projects to the north pole $\boldsymbol{n}$, and the origin projects to the south pole $\boldsymbol{s}$.

The rule for a stereographic projection has a nice 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. 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 [34]. To simplify the calculations, we will restrict ourselves to the 1-D case, as shown in figure 2.1. 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 $\mathbf{e}_+^2 = \mathbf{e}_1^2 = 1$.

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

$$\boldsymbol{x} \;=\; \left(\frac{a}{1-b}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{2.5}$$

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

$$\begin{aligned} \boldsymbol{x'} &= a\mathbf{e}_1 + b\mathbf{e}_+ \\ &= \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+, \end{aligned} \tag{2.6}$$

and using homogeneous coordinates this leads to a homogeneous representa-

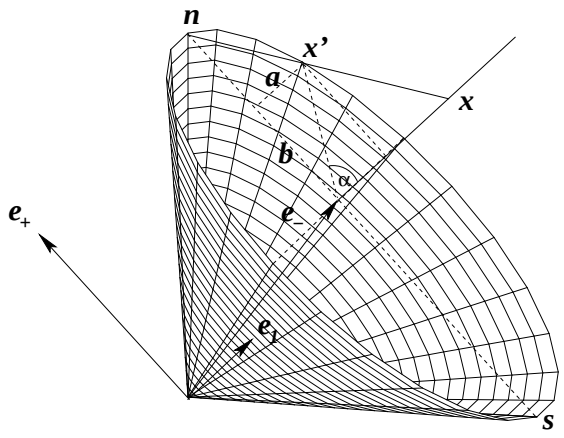

Fig. 2.2: Visualization of the homogeneous model for stereographic projections for the 1D case. All stereographic projected points are on a cone, which is a null-cone in the Minkowski space. Note that in comparison to figure 2.1, the coordinate axes are rotated and perspective drawn.

tion of the point on the circle as

$$\boldsymbol{x}' \;=\; x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}_3. \tag{2.7}$$

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

$$\boldsymbol{x} \;\Rightarrow\; \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_3. \tag{2.8}$$

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 2.1, are then represented by the set of null vectors in our new space. That is, we have the mapping

$$\boldsymbol{x} \;\Rightarrow\; \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_-, \tag{2.9}$$

with

$$(\boldsymbol{x}')^2 = a^2 + b^2 - 1 = 0 \tag{2.10}$$

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 [27] 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 $\mathcal{G}_{n+1,1}$ 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 $\mathcal{G}_{n+1,1}$, the conformal geometric algebra. Figure 2.2 visualizes the homogeneous model for stereographic projections for the 1D case.

Substituting the expressions for a and b from equation (2.6) into equation (2.9), we get

$$\boldsymbol{x}' \;=\; x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}_-. \tag{2.11}$$

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.

To introduce the conformal geometric algebra (CGA) we follow [27] and start with the *Minkowski plane* $\mathbb{R}^{1,1}$, which has an orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the properties

$$\mathbf{e}_+^2 = 1 \quad \mathbf{e}_-^2 = -1 \quad \text{and} \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0. \tag{2.12}$$

A *null basis* can now be introduced by the vectors

$$\mathbf{e}_0 := \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} := \mathbf{e}_- + \mathbf{e}_+, \tag{2.13}$$

with $\mathbf{e}_0^2 = \mathbf{e}^2 = 0$. The vector $\mathbf{e}_0$ can be interpreted as the origin, and the vector $\mathbf{e}$ as a point at infinity. Note that this is in consistency with figure 2.2: $\mathbf{e}_0$ corresponds to the south pole, $\boldsymbol{s}$, and $\mathbf{e}$ corresponds to the north pole, $\boldsymbol{n}$, in homogeneous coordinates. Furthermore we define $\boldsymbol{E} := \mathbf{e} \wedge \mathbf{e}_0$.

In the case of working in an n-dimensional vector space $\mathbb{R}^n$ we couple an additional vector space $\mathbb{R}^{1,1}$, which defines a null space to gain $\mathbb{R}^n \oplus \mathbb{R}^{1,1} = \mathbb{R}^{n+1,1}$. From that vector space we can derive the conformal geometric algebra (CGA) $\mathcal{G}_{n+1,1}$ as linear space of dimension 2^{n+2}. For the 3-dimensional vector space $\mathbb{R}^3$ we gain $\mathcal{G}_{4,1}$, which contains $2^5 = 32$ elements of different structure.

The algebras $\mathcal{G}_{3,1}$ and $\mathcal{G}_{3,0}$ are suited to represent the projective and Euclidean space, respectively [19, 21]. Since

$$\mathcal{G}_{4,1} \supseteq \mathcal{G}_{3,1} \supseteq \mathcal{G}_{3,0}, \tag{2.14}$$

both algebras for the projective and Euclidean space constitute subspaces of the linear space of the CGA. It is possible to use operators to relate the different algebras and to guarantee the mapping between the algebraic properties. This relation is also interesting since it builds another stratification hierarchy, containing the Euclidean, projective and conformal space, in contrast to Faugeras' stratification hierarchy [14], containing the Euclidean, affine and projective space.

The basis entities of the 3D conformal space are spheres $\underline{\boldsymbol{s}}$, defined by the center $\boldsymbol{p}$ and the radius ρ, $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0$. A point $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0$ is nothing else but a degenerate sphere with radius $\rho = 0$, which can easily be seen from the representation of a sphere. Evaluating $\underline{\boldsymbol{x}}$ leads to

$$\begin{aligned} \underline{\boldsymbol{x}} &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0 \\ &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2(\mathbf{e}_+ + \mathbf{e}_-) + \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \\ &= \boldsymbol{x} + \left(\frac{1}{2}\boldsymbol{x}^2 - \frac{1}{2}\right)\mathbf{e}_+ + \left(\frac{1}{2}\boldsymbol{x}^2 + \frac{1}{2}\right)\mathbf{e}_- \end{aligned} \tag{2.15}$$

This is exactly the homogeneous representation of a stereographic projected point, given in (2.11). The basis vectors $\{\mathbf{e}, \mathbf{e}_0\}$ only allow for a more compact representation of vectors than when using $\{\mathbf{e}_+, \mathbf{e}_-\}$.

A point $\underline{\boldsymbol{x}}$ is on a sphere $\underline{\boldsymbol{s}}$ iff $\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} = 0$. As shown in [42], dual points, lines and planes can be expressed as $\underline{\boldsymbol{X}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{x}}$, $\underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$. But since we only work with the entities in their dual representation, we neglect the $\star$-sign in the further formulas.

In this work we do not use all properties which are offered by the conformal geometric algebra. There is no need for us to estimate e.g. inversions or other conformal mappings, which can be estimated in conformal geometric algebra. The properties we need are the intrinsic relation of projective and conformal geometry and the possibility to express rigid motions in a linear manner.

2.2 Twists as generators of rigid body motion in CGA

This section concerns the estimation of rigid body motion in conformal geometric algebra. It is well known, that a rigid motion of an object is a

continuous movement of the particles in the object such that the distance between any two particles remains fixed at all times [33]. A rigid motion is constituted by a rotation $\mathcal{R}$ and a translation $\mathcal{T}$. In CGA both operations can be expressed in a linear manner and they also can be applied to different entities (e.g. points, lines, circles, spheres) in the same manner. Rotations in $\mathcal{G}_{4,1}$ are represented by rotors,

$$\begin{aligned} \boldsymbol{R} &= \exp\left(-\frac{\theta}{2}\boldsymbol{l}\right) = \sum_{k=0}^{\infty} \frac{\left(-\frac{\theta}{2}\boldsymbol{l}\right)^k}{k!} \\ &= \cos\left(\frac{\theta}{2}\right) - \boldsymbol{l}\sin\left(\frac{\theta}{2}\right). \end{aligned} \tag{2.16}$$

The components of the rotor $\boldsymbol{R}$ are the unit bivector $\boldsymbol{l}$, which represents the dual of the rotation axis, and the angle θ, which represents the amount of rotation. If we want to translate an entity with respect to a translation vector $\boldsymbol{t} \in \mathcal{G}_{3,0}$, we can use a so-called *translator*,

$$\boldsymbol{T} = \exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right) = (1 + \frac{\mathbf{e}\boldsymbol{t}}{2}). \tag{2.17}$$

This translator is a special rotor, similar to a translator in the dual quaternion algebra. The main difference of CGA to the dual quaternion algebra [38] or motor algebra [5] is with respect to encode geometric entities. This leads to remarkable differences in estimating rigid motions for different entities. Let be $\underline{\boldsymbol{X}}$ a point in CGA. Then rotations and translations can be expressed by applying rotors and translators as versor products [20], e.g. $\underline{\boldsymbol{X}}' = \boldsymbol{R}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{R}}$, or $\underline{\boldsymbol{X}}'' = \boldsymbol{T}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{T}}$ [1]. To express a rigid body motion we concatenate multiplicatively a rotor and a translator. Such an operator (it is a special even-grade multivector) will be denoted as a motor $\boldsymbol{M}$, which is an abbreviation of "moment and vector". The rigid body motion of e.g. a point $\underline{\boldsymbol{X}}$ can be written as

$$\begin{aligned} \underline{\boldsymbol{X}}' &= \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \\ &= \boldsymbol{T}\boldsymbol{R}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}, \end{aligned} \tag{2.18}$$

see also [5]. But as mentioned before, this does not only hold for point concepts. Other entities like lines, plane, circles and spheres can be transformed in exactly the same manner. Note that this is in contrast to the motor algebra: The CGA is a universal algebra since it is build from 1-vectors whereas the motor algebra is build from 2-vectors. That is the reason why estimating

[1] $\widetilde{\boldsymbol{A}}$ denotes the algebra conjugate of $\boldsymbol{A}$ and $\boldsymbol{A}^\star$ denotes the dual representation of $\boldsymbol{A}$.

transformations in the motor algebra can only be done with taking care of different signs within the motors acting on the different entities, see [5] for more details.

Following e.g. [33], a rigid body motion of points can be expressed by a rotation around a line in space, which is also called a *general rotation*. This means, that the group of general rotations is isomorphic to the group of rigid body motion and therefore, they have the same generating Lie algebra. To gain the Lie algebra of a 1-parameter group action, the group action must be derivated and evaluated at zero. This leads to the generating Lie algebra, whose (scaled) exponential corresponds to a group action. General rotations are also called *twist* transformations. The Lie algebra element $\xi \in se(3)$ is a twist, and its Lie group element, the exponential $g = \exp(\xi\theta) \in SE(3)$ describes a rigid body motion [16]. This representation can also be applied to the motors and, thus, expresses a rotation of any of the above mentioned entities around a line in the space. This is in contrast to Euclidean geometry, where Lie algebras and Lie groups are only applied on point concepts.

The general idea of the twist representation of a motor is to translate both, the entity and the line to the origin, to perform a rotation and to translate back the transformed entity. The motor $\boldsymbol{M}$, interpreted as the exponential of a twist, may be written as

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}} \\ &= \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right) \\ &= \exp\left(-\frac{\theta}{2}\Psi\right). \end{aligned} \tag{2.19}$$

The rigid body motion of a point can then be written as

$$\begin{aligned} \underline{\boldsymbol{X}}' &= \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \\ &= (\boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}})\underline{\boldsymbol{X}}(\boldsymbol{T}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}). \end{aligned} \tag{2.20}$$

Note that we use ξ for a twist in the affine space and Ψ for a twist in the conformal space. For points these two structures are equivalent. But Ψ is more general since it can also be applied on other entities.

2.3 Operational definition of cycloidal curves

While in the last section we stated that twists can be considered as generators of the Lie group of rigid body motion for a certain set of entities, we will consider here curves as generalized geometric entities which result from twists

as orbits of a certain Lie group. That is, here we restrict ourselves to model curves by the algebraic constrained motion of points in space. As previously explained, cycloidal curves are circles rolling on circles or lines. In this section we will explain how to generate such curves as twist depending functions in conformal geometric algebra. For example conics are no entities which can

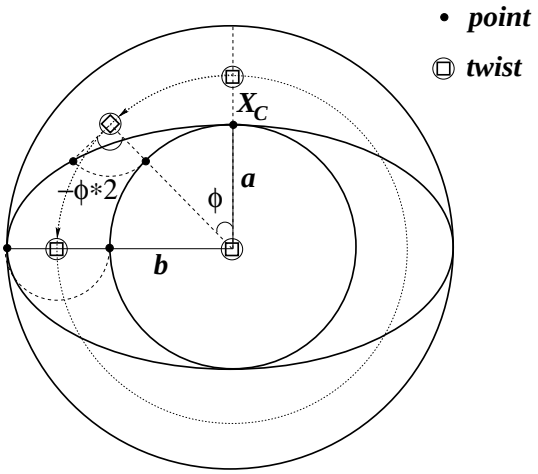

Fig. 2.3: A conic generated by two coupled twists

be directly described in conformal geometric algebra. The idea for modeling conics is visualized in figure 2.3: We assume two parallel twists in 3D space and a 3D point on the conic, and we transform the point around the two twists in a fixed and dependent manner. In this case we use two coupled parallel (not collinear) twists, rotate the point by -2ϕ around the first twist and by ϕ around the second one. The set of all points for $\phi \in [0, \ldots, 2\pi]$ generates a conic as the orbit of the corresponding Lie group.

In general, every cycloidal curve is generated by a set of twists ξ_i with *frequencies* λ_i acting on one point $\underline{\boldsymbol{X}}$ on the curve. Since m twists can be used to describe general rotations in the 2D plane or 3D space, we call the generated curves nD-$m$twist curves. With nD-$m$twist curves we mean n dimensional curves, generated by m twists with $n, m \in \mathbb{N}$. In the context of the 2D-3D pose estimation problem we use the cycloidal curves as 3D object entities. So we mean 3D-$m$twist curves, if we speak of just $m$twist curves.

We will start with very simple curves. The simplest one consists of one point (a point on the curve) and one twist. Rotating the point around the twist leads to the parameterized generation of a circle: The transformation can be expressed with a suitable motor $\boldsymbol{M}_\phi$ and an arbitrary 3D point, $\underline{\boldsymbol{X}}_Z$, on the circle. The 3D orbit of all locations on the circle the point can take

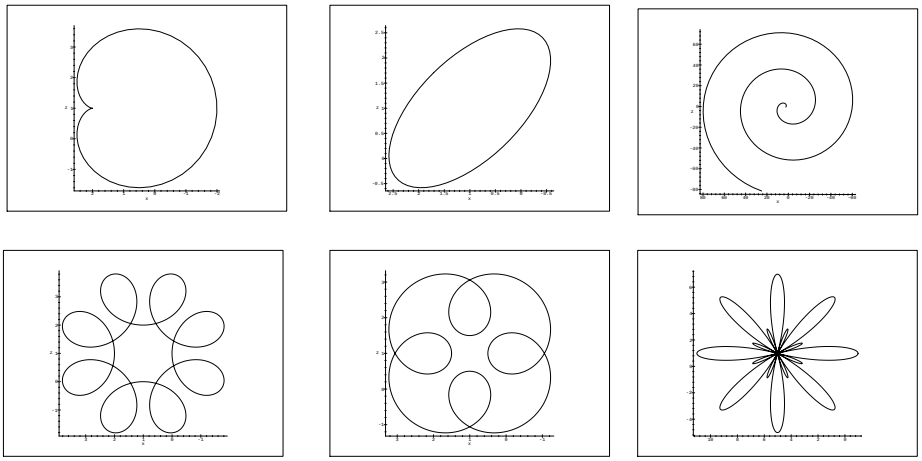

Fig. 2.4: Curves generated from 3D-2twists with parallel axes.

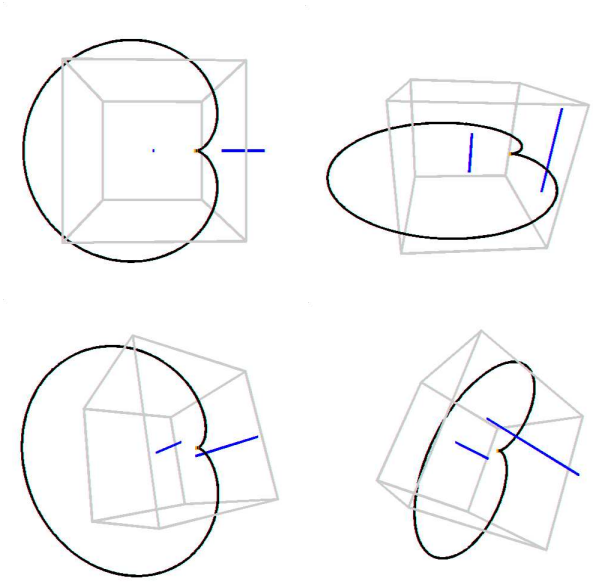

Fig. 2.5: Perspective views of a 3D-2twist generated curve. The 2twist curve, and the twists axes are visualized.

on is simply given by

$$\underline{\boldsymbol{X}}_Z^{\phi} \;=\; \boldsymbol{M}_{\phi}\underline{\boldsymbol{X}}_Z\widetilde{\boldsymbol{M}}_{\phi} \quad : \quad \phi \in [0,\ldots,2\pi]. \tag{2.21}$$

We call a circle also a *1twist* generated curve. The points on the orbit are constraint by the motor $\boldsymbol{M}_{\phi}$ as element of a Lie group. This is in contrast to classical subspace concepts in vector spaces.

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

Tab. 2.1: Well known 3D entities as $m$twist curves or surfaces

Now we can continue, and wrap a second twist around the first one. If we make the amount of rotation of each twist dependent on each other, we gain a 3D curve in general. This curve is firstly dependent on the relative positions and orientation of the twists with respect to each other, the (starting) point on the curve, and the ratio of angular frequencies. For parallel twist axes we gain a 2D curve in 3D space, whereas we get a 3D curve in 3D space for non-parallel twist axes.

The general form of a *2twist* generated curve is

$$\begin{aligned} \underline{\boldsymbol{X}}_C^{\phi} &= \boldsymbol{M}^2_{\lambda_2\phi}\boldsymbol{M}^1_{\lambda_1\phi}\underline{\boldsymbol{X}}_C\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi} \\ &= \exp\left(-\frac{\lambda_2\phi}{2}\Psi_2\right)\exp\left(-\frac{\lambda_1\phi}{2}\Psi_1\right)\underline{\boldsymbol{X}}_C\exp\left(\frac{\lambda_1\phi}{2}\Psi_1\right)\exp\left(\frac{\lambda_2\phi}{2}\Psi_2\right) \\ &\quad : \quad \lambda_1,\lambda_2 \in \mathbb{R},\ \ \phi \in [\alpha_1,\ldots,\alpha_2]. \end{aligned} \tag{2.22}$$

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

Figure 2.4 shows further examples of curves, which can be very easily generated by two coupled twists. Note that also the archimedic spiral is a 2twist generated curve. To gain an archimedic spiral, one twist has to be a translator. All these curves are given in the 3D space. In figure 2.4 only projections are shown. Figure 2.5 shows different projective views of a 3D twist generated curve. Table 2.1 gives an overview of some well known entities, interpreted as twist generated curves as well as twist generated surfaces.

The rigid body motion of these entities can easily be estimated, just by transforming the generating twists. The transformation of an $m$twist generated curve can be performed by transforming the m twists (which are just lines in the space) and the point on the curve. The description of these curves is compact, and rigid transformations can be estimated very fast. We

will focus on curves only in this paper.

2.4 Estimating twists from a given closed curve

So far we have discussed how a set of multiplicatively coupled twists can be used to generate a curve. Similarly, we can ask how a given closed curve may be parameterized with respect to a set of additively coupled twists. This problem is in fact closely related to Fourier descriptors, which are used for object recognition [17, 52, 3, 22] and affine pose estimation [4, 37] of closed contours. We will show here that a set of coupled twists acting on a vector is equivalent to a sum over a set of rotors, which each act on a different phase vector. The latter can be regarded as a Fourier series expansion, whose coefficients are also called Fourier descriptors.

The equivalence of coupled twists and a Fourier expansion is most easily shown in Euclidean space. Let

$$\boldsymbol{R}_i^\phi \;:=\; \exp(-\frac{\pi u_i \phi}{T}\boldsymbol{l}), \tag{2.23}$$

where $T \in \mathbb{R}$ is the length of the closed curve, $u_i \in \mathbb{Z}$ is a frequency number and $\boldsymbol{l}$ is a unit bivector which defines the rotation plane. Furthermore, $\widetilde{\boldsymbol{R}}_i^\phi = \exp(\pi u_i \phi / T\, \boldsymbol{l})$. Recall that $\boldsymbol{l}^2 = -1$ and, as noted in equation (2.16), we can therefore write the exponential function as

$$\exp(\phi\, \boldsymbol{l}) = \cos(\phi) + \sin(\phi)\, \boldsymbol{l}. \tag{2.24}$$

A 2twist generated curve may then be written in Euclidean space as follows,

$$\begin{aligned}
\underline{\boldsymbol{X}}_C^\phi &= \boldsymbol{M}^2_{\lambda_2\phi}\boldsymbol{M}^1_{\lambda_1\phi}\underline{\boldsymbol{X}}_C\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi} \\
\Leftrightarrow \boldsymbol{x}_C^\phi &= \boldsymbol{R}_1^\phi\left(\left(\boldsymbol{R}_1^\phi\,(\boldsymbol{x}_C - \boldsymbol{t}_1)\,\widetilde{\boldsymbol{R}}_1^\phi + \boldsymbol{t}_1\right) - \boldsymbol{t}_2\right)\widetilde{\boldsymbol{R}}_1^\phi + \boldsymbol{t}_2 \\
&= \boldsymbol{R}_2^\phi\boldsymbol{R}_1^\phi(\boldsymbol{x}_C - \boldsymbol{t}_1)\widetilde{\boldsymbol{R}}_1^\phi\widetilde{\boldsymbol{R}}_2^\phi + \boldsymbol{R}_2^\phi(\boldsymbol{t}_1 - \boldsymbol{t}_2)\widetilde{\boldsymbol{R}}_2^\phi + \boldsymbol{t}_2 \\
&= \boldsymbol{p}_0 + \boldsymbol{V}_1^\phi\,\boldsymbol{p}_1\,\widetilde{\boldsymbol{V}}_1^\phi + \boldsymbol{V}_2^\phi\,\boldsymbol{p}_2\,\widetilde{\boldsymbol{V}}_2^\phi,
\end{aligned} \tag{2.25}$$

where $\boldsymbol{p}_0 \equiv \boldsymbol{t}_2$, $\boldsymbol{p}_1 \equiv \boldsymbol{t}_1 - \boldsymbol{t}_2$, $\boldsymbol{p}_2 \equiv \boldsymbol{x}_C - \boldsymbol{t}_1$, $\boldsymbol{V}_1^\phi \equiv \boldsymbol{R}_2^\phi$, $\boldsymbol{V}_2^\phi \equiv \boldsymbol{R}_2^\phi\boldsymbol{R}_1^\phi$ and $\lambda_i = 2\pi u_i / T$. Note that for planar curves the rotors $\boldsymbol{R}_1^\phi$ and $\boldsymbol{R}_2^\phi$ act in the same plane and the vectors $\boldsymbol{x}_C$, $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ lie in the rotation plane. Hence, the $\{\boldsymbol{p}_i\}$ lie in the rotation plane.

It can be shown that if a vector $\boldsymbol{x}$ lies in the rotation plane of some rotor $\boldsymbol{R}$, then $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{x}\widetilde{\boldsymbol{R}}$. The previous equation can therefore be written as

$$\boldsymbol{x}_C^\phi \;=\; \boldsymbol{p}_0 + \boldsymbol{p}_1\,\widetilde{\boldsymbol{V}}_1^{2\phi} + \boldsymbol{p}_2\,\widetilde{\boldsymbol{V}}_2^{2\phi}. \tag{2.26}$$

Note that the square of a rotor is equal to a rotor of twice the angle in the same rotation plane. Therefore, $\widetilde{\boldsymbol{V}}_i^{\phi}\widetilde{\boldsymbol{V}}_i^{\phi} = \widetilde{\boldsymbol{V}}_i^{2\phi}$. Using the exponential form of rotors, we get

$$\boldsymbol{x}_C^{\phi} \;=\; \boldsymbol{p}_0 + \boldsymbol{p}_1 \exp\left(\frac{2\pi u_1 \phi}{T}\boldsymbol{l}\right) + \boldsymbol{p}_2 \exp\left(\frac{2\pi u_2 \phi}{T}\boldsymbol{l}\right). \tag{2.27}$$

This is equivalent to a Fourier series expansion where we have replaced the imaginary unit $i = \sqrt{-1}$ with $\boldsymbol{l}$ and the complex Fourier series coefficients with vectors that lie in the plane spanned by $\boldsymbol{l}$. The latter vectors are the phase vectors. In general it may be shown that any closed, planar curve $C(\phi)$ can be expressed as a series expansion

$$C(\phi) = \lim_{N\to\infty} \sum_{k=-N}^{N} \boldsymbol{p}_k \exp\left(\frac{2\pi k \phi}{T}\boldsymbol{l}\right) = \lim_{N\to\infty} \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi}. \tag{2.28}$$

For every closed curve there is a unique set of phase vectors $\{\boldsymbol{p}_k\}$ that parameterizes the curve. However, such a set corresponds to infinitely many different combinations of coupled twists. That is, given a set of coupled twists, we can obtain the corresponding phase vectors $\{\boldsymbol{p}_k\}$ but not vice versa. The spectral representation of a curve transforms the translational parts of its generating twists into a set of different phase vectors and therefore results in a pure rotor description. This additive representation is unique, whereas the multiplicative coupled twist representation is not. Therefore, we use the additive description for our pose estimation scenario later on.

The expansion in equation (2.28) is again closely related to the standard Fourier series expansion of a real, scalar valued function. In figure 2.6 a closed curve created by two coupled twists is shown in the yz-plane. Suppose that instead of $C(\phi)$ we consider $C_S(\phi) := C(\phi) + 2\pi\phi/T\, \boldsymbol{e}_1$, where $\boldsymbol{e}_1$ is the unit vector along the x-axis. If we project $C_S(\phi)$ onto the xy-plane and xz-plane, we obtain the two other curves shown. This visualizes the well known fact that we can regard any periodic function in a space of dimension n as the projection of a closed curve in a space of dimension $n+1$.

The phase vectors $\{\boldsymbol{p}_k\}$ are also called *Fourier descriptors*. It has long been known that one can also construct affine invariant Fourier descriptors [17, 1], that is, entities that describe a closed curve and stay invariant under affine transformations of the curve. This is particularly useful for object recognition and has been used in many applications [3, 15, 47]. The same relations that allow one to construct affine invariant Fourier descriptors also allow for affine pose estimation. This works in the following way. Consider a closed curve that lies on a plane which is tilted with respect to an observer. This curve is projected with an affine camera onto an image plane. The pose

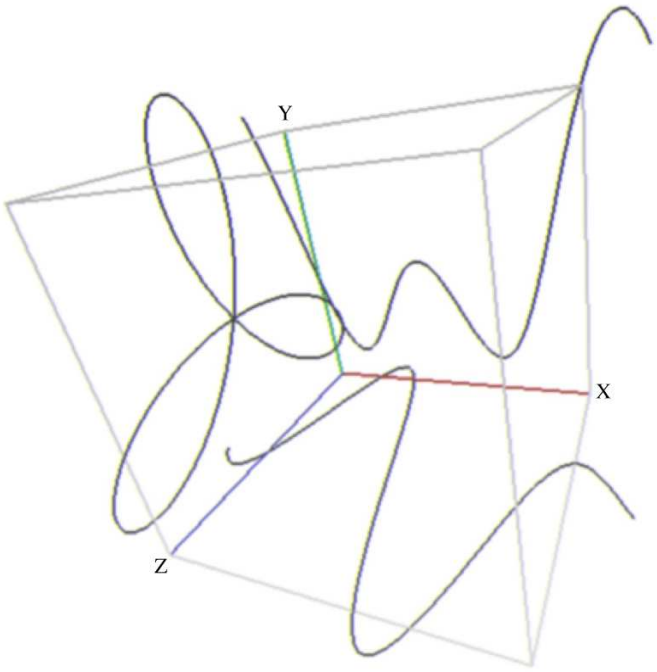

Fig. 2.6: Projections of a curve created by coupled twists.

of the plane in space can then be estimated given the Fourier descriptors of the projected curve as well as the Fourier descriptors of the original curve. See [2] for more details.

We attempted to perform a projective pose estimation via Fourier descriptors. Unfortunately, there are two major problems. First of all, if a closed curve is projected projectively, then the projected curve will not be sampled in the same way as the original curve. This already distorts the Fourier descriptors. Secondly, going through the equations we found that in order to solve the projective pose estimation problem via Fourier descriptors, one has to find analytic solutions to n^{th} degree polynomials. Since this is not possible in general, we cannot follow this approach. See appendix A for more details. We therefore investigated a different approach for the pose estimation of projected closed curves, which will be discussed in the following.

3. POSE ESTIMATION IN CGA

This section concerns the pose estimation problem. So far we have just formalized free-form entities and their twist representation. Now we will continue to formalize the 2D-3D pose estimation problem.

3.1 Pose estimation in stratified spaces

To define the pose problem we want to quote Grimson [18]:

Definition 3.1 *By pose, we mean the transformation needed to map an object model from its own inherent coordinate system into agreement with the sensory data.*

Thus, pose estimation is to relate several coordinate frames of measurement data and model data by finding out the transformation between. 2D-3D pose estimation means to estimate the relative position and orientation of a 3D object to a reference camera system. We already formalized our entities in the conformal algebra because we want to formalize the pose estimation problem in the conformal space. That is, *a kinematic transformed object entity has to lie on a projective reconstructed image entity.* Let $\underline{\boldsymbol{X}}$ be an object point given in CGA. The (unknown) transformation of the point can be written as $\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$. Let $\boldsymbol{x}$ be an image point on a projective plane. The projective reconstruction from an image point in CGA can be written as $\underline{\boldsymbol{L}}_x = \mathbf{e} \wedge \boldsymbol{o} \wedge \boldsymbol{x}$. This leads to a reconstructed projection ray, containing the optical center $\boldsymbol{o}$ of the camera, see e.g. figure 1.1, the image point $\boldsymbol{x}$ and the vector $\mathbf{e}$ as the point at infinity. Note that $\boldsymbol{o} \wedge \boldsymbol{x}$ formalizes the reconstructed ray in projective geometry. The expression $\mathbf{e} \wedge \boldsymbol{o} \wedge \boldsymbol{x}$ represents the reconstructed ray in conformal geometry and is therefore given in the same language as we use for our $m$twist generated curves.

To express the incidence of a transformed point with a reconstructed ray we can apply the commutator product, which expresses collinearity and directly transforms the constraint equation in an equation given in the Euclidean space (see e.g. [40] for the proofs). Thus, the constraint equation of

pose estimation from image points reads

$$\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\boldsymbol{o}\wedge\ \underbrace{\boldsymbol{x}}_{\text{image point}}\ )}_{\text{projection ray, reconstructed from the image point}}}_{\text{collinearity of the transformed object point with the reconstructed line}}\ =\ 0, \tag{3.1}$$

Constraint equations to relate 2D image lines to 3D object points, or 2D image lines to 3D object lines, can also be expressed in a similar manner. Note that the constraint equations implicitly represent an Euclidean distance measure which has to be zero. Such compact equations subsume the pose estimation problem at hand: find the best motor $\boldsymbol{M}$ which satisfies the constraint. But in contrast to other approaches, where the minimization of errors has to be computed directly on the manifold of the geometric transformations [8, 48], in our approach a distance in the Euclidean space constitutes the error measure. To change our constraint equation from the conformal to the Euclidean space, the equations are rescaled without loosing linearity within our unknowns.

The theoretical foundations concerning the mathematical spaces involved in the pose estimation problem and their algebraic coupling within geometric algebras is more detailed explained in [42].

3.2 Pose estimation of cycloidal curves

Now we can continue to combine the cycloidal curves with the pose estimation problem: We consider a 3D cycloidal curve, like

$$\underline{\boldsymbol{X}}_Z^{\phi}\ =\ \boldsymbol{M}^2_{\lambda_1\phi}\boldsymbol{M}^1_{\lambda_2\phi}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_2\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_1\phi}\ :\ \lambda_1,\lambda_2\in\mathbb{R},\ \phi\in[0,\ldots,2\pi]. \tag{3.2}$$

By substituting this expression within our constraint equation for pose estimation, we gain

$$\left(\boldsymbol{M}(\boldsymbol{M}^2_{\lambda_1\phi}\boldsymbol{M}^1_{\lambda_2\phi}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_2\phi}\widetilde{\boldsymbol{M}}^2_{\lambda_1\phi})\widetilde{\boldsymbol{M}}\right)\ \underline{\times}\ (\mathbf{e}\wedge(\boldsymbol{o}\wedge\boldsymbol{x}))\ =\ 0. \tag{3.3}$$

Since every aspect of the 2D-3D pose estimation problem of cycloidal curves is formalized in CGA, the constraint equation describing the pose problem is compact and easy to interpret: The inner parenthesis on the left contains the

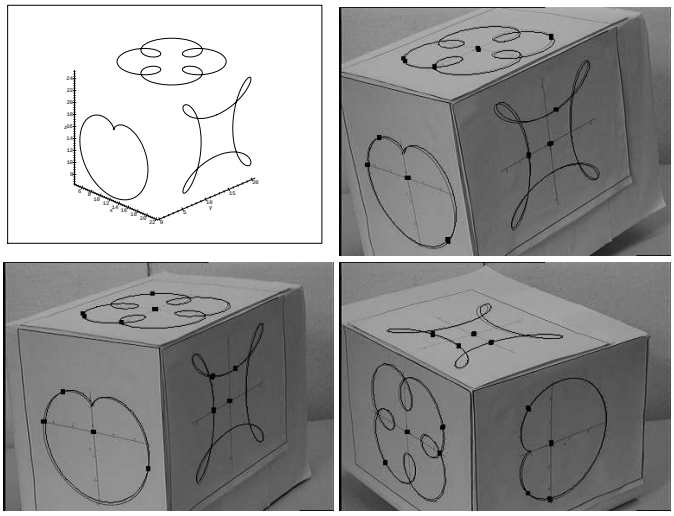

Fig. 3.1: Pose estimation of an object containing a cardioid and two cycloids.

operational definition of the cycloidal curve. The outer parenthesis contains the unknown motor $\boldsymbol{M}$, describing the rigid body motion of the 3D cycloidal curve. This is the pose we are interested in. The expression is then combined via the commutator product with the reconstructed projection ray and has to be zero. This describes the co-tangentiality of the transformed curve to a projection ray. The point $\boldsymbol{x}$ is a member of a 2D contour in the image plane.

The unknowns are the six parameters of the rigid motion $\boldsymbol{M}$ (three for the location of the line, two for its orientation and one rotation angle) and the angle ϕ for each point correspondence. An example for pose estimation of cycloidal curves is shown in figure 3.1. The upper left image shows the 3D object model. The other images show pose results of the model. To visualize the quality, the transformed and projected object model is overlaid in the images.

3.3 Pose estimation of free-form contours

So far we considered continuous 3D curves as representing objects. Now we assume a given closed, discretizied 3D curve, that is a 3D contour C with $2N$ sampled points in both the spatial and spectral domain with phase vectors $\boldsymbol{p}_k$ of the contour. We now replace a Fourier series development by the discrete Fourier transform. Then the interpolated contour can be expressed in the

Euclidean space as

$$C(\phi) \;=\; \sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi}. \tag{3.4}$$

For each ϕ does $C(\phi)$ lead to a point in the Euclidean space. We first have to transform this expression in the conformal space. Then we can, similar to the previous section, substitute this expression in the constraint equations for pose estimation. The transformation of the Fourier descriptors in the conformal space can be expressed as

$$\mathbf{e} \wedge (C(\phi) + \mathbf{e}_-) \;=\; \mathbf{e} \wedge \left(\left(\sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \right) + \mathbf{e}_- \right). \tag{3.5}$$

The innermost parenthesis contains the Fourier descriptors in the Euclidean space. The next parentheses transform this expression in the homogeneous space and then it is transformed to the conformal space. Substituting this expression in the pose constraint equation leads to

$$\begin{aligned} \left(\boldsymbol{M}(\mathbf{e} \wedge (C(\phi) + \mathbf{e}_-))\widetilde{\boldsymbol{M}}\right) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{o} \wedge \boldsymbol{x})) \;&=\; 0 \Leftrightarrow \\ \left(\boldsymbol{M} \left(\mathbf{e} \wedge \left(\left(\sum_{k=-N}^{N} \boldsymbol{R}_k^{\phi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\phi} \right) + \mathbf{e}_- \right) \right) \widetilde{\boldsymbol{M}} \right) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{o} \wedge \boldsymbol{x})) \;&=\; 0. \end{aligned} \tag{3.6}$$

The interpretation of this equation is also simple: The innermost part contains the substituted Fourier descriptors in the conformal space of equation (3.5). This is then coupled with the unknown rigid body motion (the motor $\boldsymbol{M}$) and compared with a reconstructed projection ray, also given in the conformal space.

Note that cycloidal curves are in that respect more general than contours as we assume contours as closed curves, whereas cycloidal curves (see e.g. a spiral) are in general not closed. This means, for closed curves can Fourier descriptors be interpreted as generator parameters of special cycloidal curves, but not vice versa. The main point is the coupling of a spectral representation of contours within the pose estimation problem. This is achieved in the previous equation by using a conformal embedding.

3.4 Estimation of pose parameters

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 may be represented as a polynomial of infinite degree (see, e.g., (2.16)

for the series expression of the exponential function), this is a non-trivial task, especially in the case of real-time estimations. The idea is to gain linear equations with respect to the generators of the motor. We use the exponential representation of motors and apply the Taylor series expression of first order for approximation. This corresponds 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 [16], but for the parameters which generate their Lie algebra $se(3)$ [33]. From this result linear equations in the generators of the unknown 3D rigid body motion. For the sake of simplicity we will show the linearization for 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:

$$\begin{aligned} \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))\right)\underline{\boldsymbol{X}}\exp\left(\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))\right) \\ &\approx (1-\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})))\underline{\boldsymbol{X}}(1+\frac{\theta}{2}(\boldsymbol{l}+\mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l}))) \\ &\approx \boldsymbol{E}+\mathbf{e}(\boldsymbol{x}+\theta(\boldsymbol{l}\cdot\boldsymbol{x})-\theta(\boldsymbol{t}\cdot\boldsymbol{l})) \end{aligned} \tag{3.7}$$

Setting $\boldsymbol{v} := \theta(\boldsymbol{l}\cdot\boldsymbol{x})$ and $\boldsymbol{m} := \theta(\boldsymbol{t}\cdot\boldsymbol{l})$, we get

$$\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \approx \boldsymbol{E}+\mathbf{e}(\boldsymbol{x}+\boldsymbol{v}-\boldsymbol{m}). \tag{3.8}$$

The combination of this approximation of the motion with the previously derived constraints for pose estimation results in

$$\begin{aligned} \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} \underline{\times} (\mathbf{e}\wedge(\boldsymbol{o}\wedge\boldsymbol{x})) &= 0 \\ \Leftarrow\approx\Rightarrow (\boldsymbol{E}+\mathbf{e}(\boldsymbol{x}+\boldsymbol{v}-\boldsymbol{m}))\underline{\times}(\mathbf{e}\wedge(\boldsymbol{o}\wedge\boldsymbol{x})) &= 0. \\ \Leftrightarrow \lambda(\boldsymbol{E}+\mathbf{e}(\boldsymbol{x}+\boldsymbol{v}-\boldsymbol{m}))\underline{\times}(\mathbf{e}\wedge(\boldsymbol{o}\wedge\boldsymbol{x})) &= 0 \end{aligned} \tag{3.9}$$

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 twist parameters for the general rotation (five unknowns for the location of the twist and one unknown angle, represented by the unknowns $\boldsymbol{v}$ and $\boldsymbol{m}$). Accumulating a set of such linear equations (for different features of the object model), leads to a linear system of equations. The system of linear equations can be solved for a set of correspondences by applying e.g. the well known Householder method. From the solution of the system of equations, the motion parameters $\boldsymbol{R}, \boldsymbol{t}$ can easily be recovered by evaluating $\theta := \|\boldsymbol{v}\|$, $\boldsymbol{l} := \frac{\boldsymbol{v}}{\theta}$ and $(\boldsymbol{t}\cdot\boldsymbol{l}) := \frac{-\boldsymbol{m}}{\theta}$.

Solving these equations, we get a first approximation of the rigid body motion. Iterating this process leads to a monotonous convergence to the actual pose and only a few iterations (mostly $5-8$) are sufficient to get a good approximated pose result. The algorithm itself corresponds to a gradient descend method applied in the 3D space. Note that the monotonous convergence does sometime leads to local minima. The aim is to avoid such local minima. Therefore we use low-pass information for contour approximation.

4. EXPERIMENTS

In this section we present experimental results of free-form contour pose estimation. Therefore we will start with an introduction to the main algorithm for pose estimation of free-form contours. Though the numerical estimation of the pose parameters is already clarified in the last section, the main problem is to determine suited correspondences between 2D image features and points on the 3D model curve. Therefore a version of an ICP-algorithm is presented and called the *increasing degree* method. Then we will continue with experiments on the convergence behavior of our algorithm and time performance versus accuracy. There also stability examples for distorted image data are shown. The algorithm proofs as stable and fast (real-time capable) for our scenarios. To deal with 3D objects and partial occluded aspects of objects during tracking, we then present a modified version of our increasing degree method. There we are able to deal with occlusion problems by using sets of Fourier descriptors to model aspects of the object within our scenario.

4.1 The algorithm of pose estimation for free-form contours

The aim is to formulate a 2D-3D pose estimation algorithm for any kind of free-form contour. The assumptions we make are the following:

1. The object model is given as a set of $2N$ 3D points f_j^3, spanning the 3D contour. Further we assume to know their phase coefficients $\boldsymbol{p}_k$.

2. In an image of a calibrated camera we observe the object in the image plane and extract a set of n 2D points x_j^2, spanning the 2D contour.

Since the number of contour points in the image is often too high (e.g. 800 points in our experimental scenario), we use just every 10th point and get an equal sampled set of contour image points.

Note that we have no knowledge which 2D image point corresponds to which 3D point of the interpolated model contour. Furthermore, a direct correspondence does not generally exist.

Using our approach for pose estimation of point-line correspondences, the algorithm for free-form contours consists of iterating the following steps:

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

The idea is, that all image contour points simultaneously pull on the 3D contour. The algorithm itself corresponds to the well-known ICP algorithm, e.g. discussed in [43, 51]. But whereas it is mostly applied on sets of 2D or 3D points we apply it on a trigonometric interpolated function and from image points reconstructed projection rays.

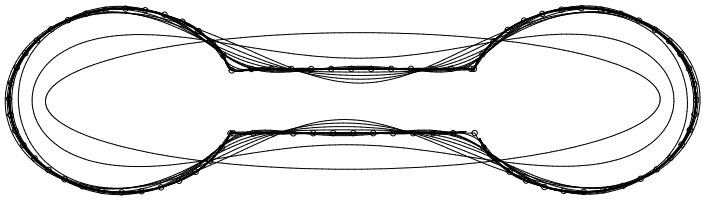

Fig. 4.1: The different approximation levels of the 3D object contour.

Note that this algorithm only works if we assume a scenario where the observations in the image plane are not too different. Thus, it is useful for tracking tasks. For our experiments we use up to 25 pixel deviation between two images of a sequence. A projection of the used object model for our first experiments is shown in figure 4.1. The discrete points and the different approximation levels are shown. The model itself consists of 90 contour points, is planar and has the width and height of 24×8 cm. Pose estimation results at different iterations are shown in figure 4.2. The white 2D contour is the transformed and projected 3D object model overlaid with the image.

Using the Fourier coefficients for contour interpolation works fine but the algorithm can be made faster by using a low-pass approximation for pose estimation and by adding successively higher frequencies during the iteration. This is basically a multi-resolution method. We call this technique the *increasing degree* method. Therefore we start the pose estimation procedure with just a few Fourier coefficients of the 3D contour and estimate

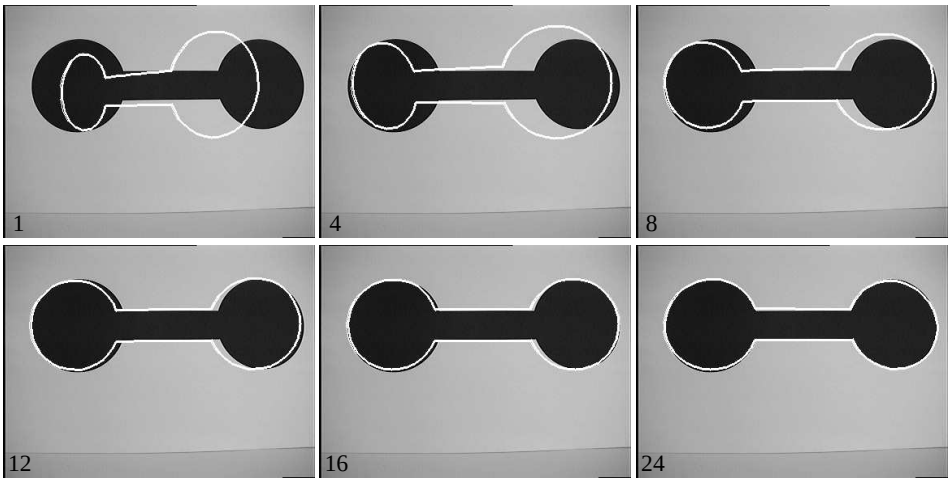

Fig. 4.2: Pose results during the iteration.

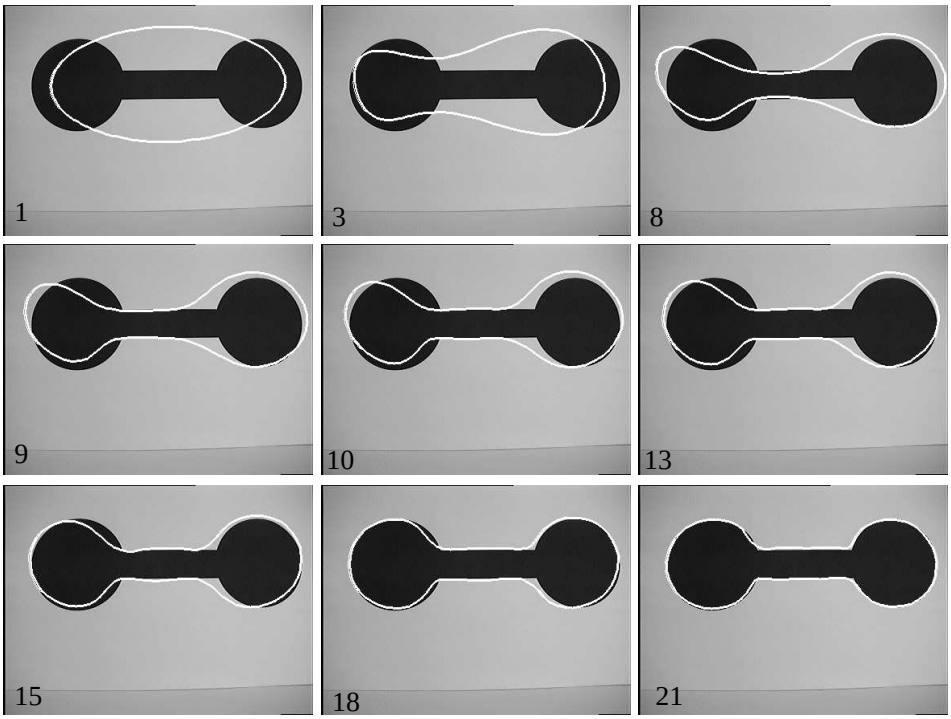

Fig. 4.3: Pose results of the low-pass filtered contour during the iteration.

the pose to a certain degree of accuracy. Then we increase the order of used Fourier coefficients and proceed to estimate the pose with the refined object description. This is shown in figure 4.3. In this experiment, the indicated iteration number corresponds directly to the number of used Fourier coef-

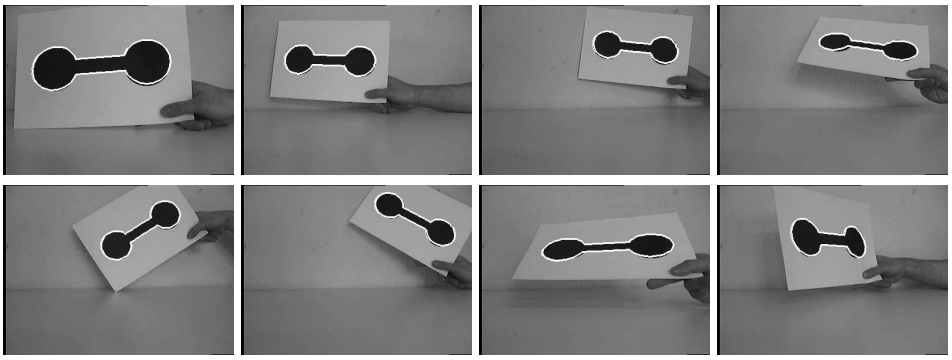

Fig. 4.4: Different pose results of the free-form contour.

ficients *minus one*. This means that we use two Fourier coefficients in the first iteration, four Fourier coefficients in the third iteration, etc. Iteration 21 uses 22 Fourier coefficients and figure 4.3 shows that the result is nearly perfect. Figure 4.4 shows pose results during an image sequence containing 530 images. As can be seen also perspective views of the free-form contour can be estimated.

4.2 The performance of the pose estimation algorithm

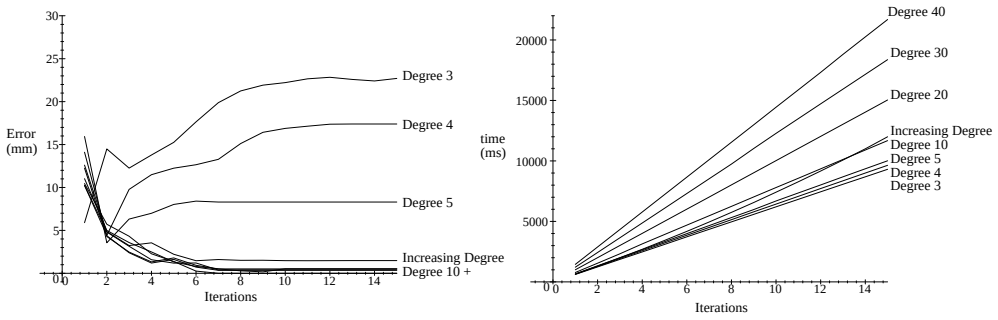

Fig. 4.5: Accuracy and computing time of the algorithm for constant number of image points (80) and different approximation levels of the contour.

The accuracy and time performance of the algorithm is dependent on the number of object and image points spanning the contours in 2D and 3D, respectively. Furthermore, we can use the low-pass approximation of the 3D

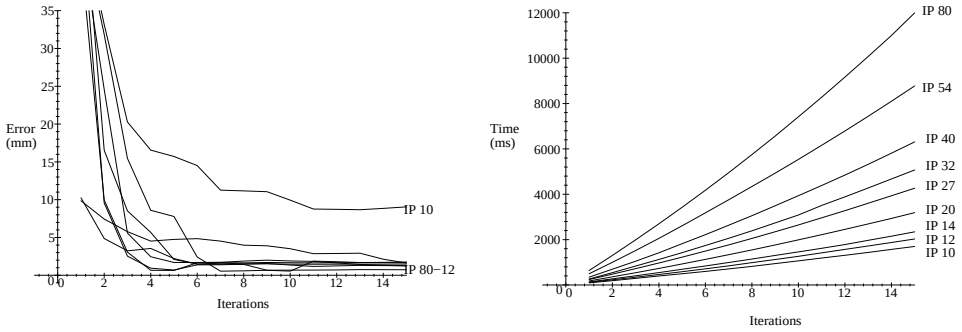

Fig. 4.6: Accuracy and computing time of the algorithm for changing number of used image points and the increasing degree algorithm.

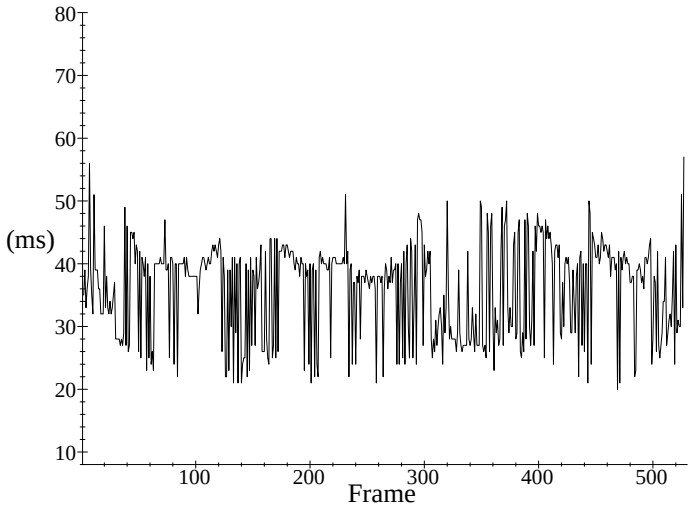

Fig. 4.7: Computing times for an image sequence containing 500 images.

contour. We made several experiments with changing approximation levels and changing number of image points.

In the first experiment we compare the results of our algorithm for different degrees of contour approximation. We use constant degrees $(3, 4, 5, 10, 20, 30, 40)$ of contour approximation over the iteration and compare the results with the method of increasing degrees, similar to figure 4.3. To test the accuracy of our algorithm we simply compare the translational

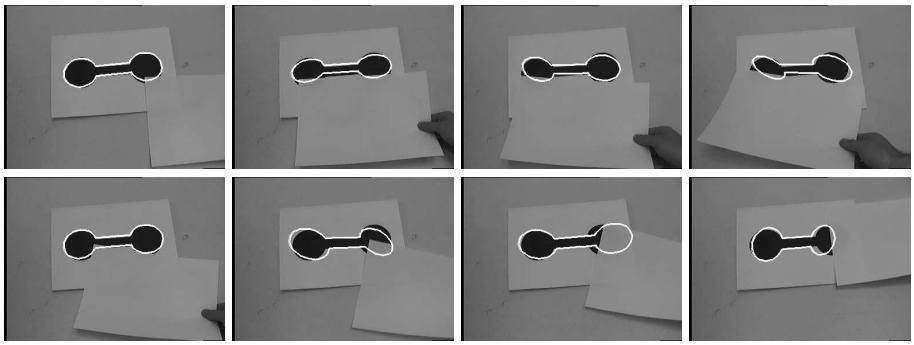

Fig. 4.8: Different pose results for distorted image data.

Computer	pose	min. search	total (25 It.)
Pentium IV 2 Ght	5ms	up to 20 ms	405 ms
Pentium III 850 Mht	15 ms	up to 25 ms	783 ms
Sparc Ultra 10	325 ms	up to 80 ms	10295 ms ($\sim$ 10 sec)
Sparc Ultra 1	8921 ms	up to 667 ms	232265 ms ($\sim$ 3.5 min)
Sparc 4	13322 ms	up to 1505 ms	356811 ms ($\sim$ 6min)
Sparc 10	23509 ms	up to 1743 ms	622975 ms ($\sim$ 10min)

Tab. 4.1: Computing time of the increasing degree method for the same scenario on different machines for 90 model points, 80 image points and 25 iterations.

error vector with the ground truth. The result is shown in figure 4.5. It can be seen that the algorithm converges after less than 10 iterations. Then the error vectors do not change any more. It is clear, that the use of less number of Fourier coefficients leads to fast but more inaccurate results. Our *increasing degree* method finds a good optimum between computing time and accuracy of the result. The algorithm converges after 7 iterations.

In a second experiment we use the *increasing degree* algorithm and change the number of extracted contour points. In this experiment we use $10, 12, 14, 20, 27, 32, 40, 54$ and 80 regularly sampled image points and compare the accuracy and computing time. The result is presented in figure 4.6. It can be seen, that the number of image points used affects both the computing time and the accuracy. But in comparison with the previous experiment there exists a critical break point with regard to the accuracy of the algorithm. While the use of $14 - 80$ image points does not affect the quality of the pose too much, the use of 10 points or less leads to wrong poses.

These results hold for just this scenario and will change for other scenarios. The main result is that it is indeed possible to use the whole image and

object information available to estimate the pose of the free-form contour. But we have to pay with computing time. Instead, the use of low-pass filtered or of sub-sampled contours fastens computing and leads to good results. In this scenario the computing time can be reduced from 35 seconds to less than 1 second without introducing non-tolerable errors.

Indeed, the computing time is very dependent on the machine itself. Therefore, we also tested the same algorithm on different machines in our group. The result is shown in table 4.1. As can be seen, e.g. the computing time for the increasing degree method, with 80 image points is 783 ms on a standard Linux 850 Mht machine. The column *pose* shows the computing time for each pose. The column *min. search* shows the computing time for estimating the minimum distance between projection rays and the model curve. Since we use the increasing degree method, the computing time for estimating the distances varies and increases with increasing number of Fourier coefficients. Therefore we just show the maximum computing time. The column *total* shows the total computing time for 25 Iterations. But since we are working with image sequences, 25 iterations are seldom needed.

Figure 4.7 shows the computing times for an image sequence containing 500 images. The computing time for each image varies between 20ms and 55ms. The average computing time is 34ms, which is equivalent to 29 fps. These results were achieved with a 2Ght Pentium IV computer.

Many ideas to fasten the algorithm can also be found in [43]. We did not improve the algorithm yet. This is part of future work. The main result is that the algorithm can also be used for real-time applications on standard Linux machines.

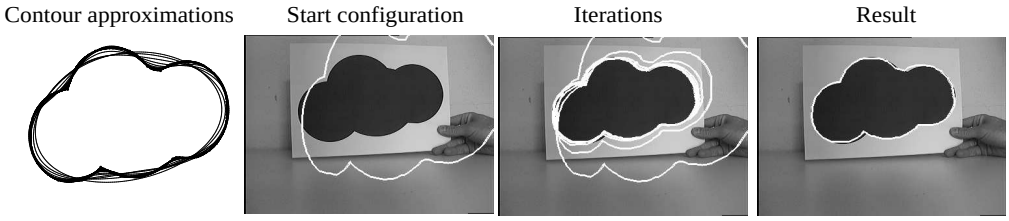

Fig. 4.9: Contour approximations of another planar object model and its convergence behavior.

The robustness of our algorithm with respect to distorted image data is shown in figure 4.8. In this image sequence (containing 450 images) we distort the image contour by covering parts of the contour with a white paper. This leads to slight or more extreme errors during the contour extraction in the image. Nevertheless, the behavior of the matching and the pose results are acceptable. These examples give just a guess about the stability of

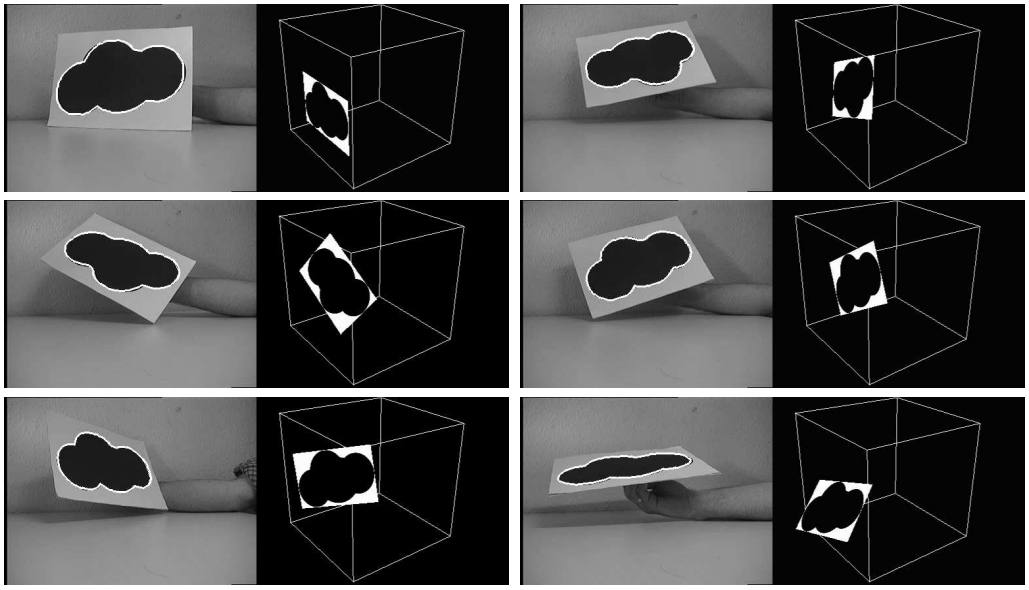

Fig. 4.10: Example images and 3D poses taken from an image sequence containing 700 images.

the proposed method. Total wrong extracted contours or too much missing informations are not possible to be compensated.

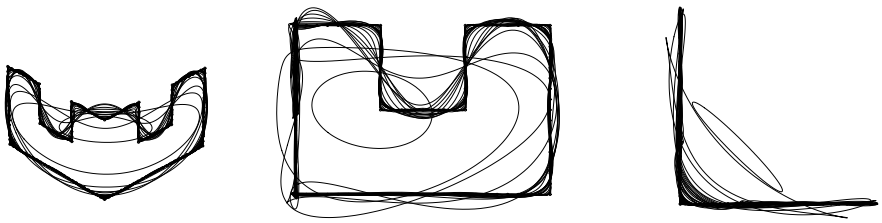

Fig. 4.11: Three perspective views of a non-planar object model and its approximations.

Figures 4.9-4.12 present results of other object models: We call the first object model the *cloud* and the second object model the *edge*. Figure 4.9 shows the 3D contour approximations in the left image and a convergence example in the other images. Figure 4.10 presents results of a sequence containing 700 images. As can be seen also strong perspective views, as in the lower right image, can be estimated. To compare the visual observable

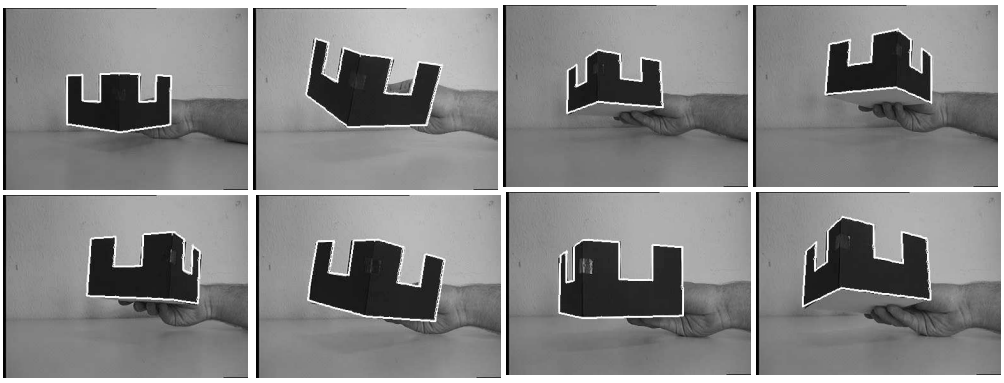

Fig. 4.12: Example images from an image sequence containing 500 images.

error (as a drawn contour in the image) with its real 3D pose we visualize in figure 4.10 the relative pose in a virtual environment. The real 3D pose matches with the observations in the image.

Figure 4.11 shows approximation levels of a non-planar object model in three different perspective views. This is an extreme example since the object model contains edges. Interpolation of a contour with Fourier descriptors leads to a trigonometric interpolated function. So the edges are always smoothed and several descriptors (we use 40) are required to achieve an acceptable result. Figure 4.12 presents different results from an image sequence containing 500 images.

Object contours which contain concavities are in danger to get trapped in local minima during using the ICP-algorithm with the gradient descend method for pose estimation. Though it is not always possible to find the global minimum (and therefore the best pose), using contour approximations helps to avoid local minima. This effect is achieved by using firstly a low-pass contour for pose estimation and then over the iterations a more refined contour.

The last two object models (the *cloud* and the *edge*) contain more local minima than our first one. Therefore we need more Fourier descriptors to gain acceptable results. This indeed decreases the computing time. While for the first object model the average computing time is 34ms, the average computing time of the cloud and edge model are 100ms and 200ms, respectively.

4.3 Simultaneous pose estimation of multiple contours

In the last experiments our object model is assumed as one (closed) contour. But many 3D objects can more easily be represented by a set of 3D contours

expressing the different aspects of the object. In this section we will extend our object model to a set of 3D contours. The main problem here is, how to deal with occluded or partially occluded contour parts of the object. For our first experiment we will use the object model which we already used in figures 4.11 and 4.12. We will interpret the model as an object containing two sides and one ground plate. This means we get a set of three planar contours to model the object. The three contours are merged to one object and perspective views are shown in figure 4.13. The three contours are assumed as rigidly coupled to each other. This means that the pose of one contour automatically defines the pose of the other contours.

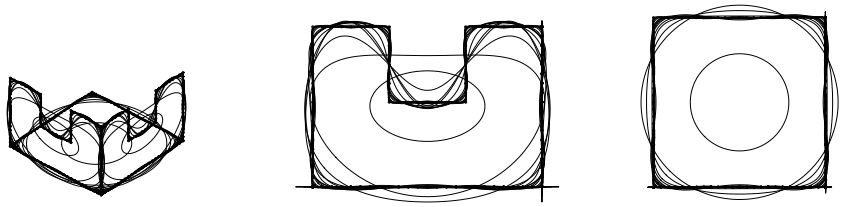

Fig. 4.13: Three perspective views of an object which is interpreted by a set of contours. The different approximations of the contours are also drawn.

Our algorithm to deal with partially occluded object parts is simple and effective:

```
Assumptions: n 3D contours and one boundary contour in the image
             dist(P,R) a distance function between a 3D point P and a
             3D ray R.
Aim: Estimate correspondences and pose.

(a) Reconstruct projection rays from the image points.
(b) For each projection ray R:
(c)   For each 3D contour:
     (c1) Estimate the nearest point P1 of ray R to a point on
     the contour.
     (c2) if (n==1) choose P1 as actual P for the
               point-line correspondence
     (c3) else compare P1 with P:
             if dist(P1,R) is smaller than dist(P,R) then
                choose P1 as new P
```

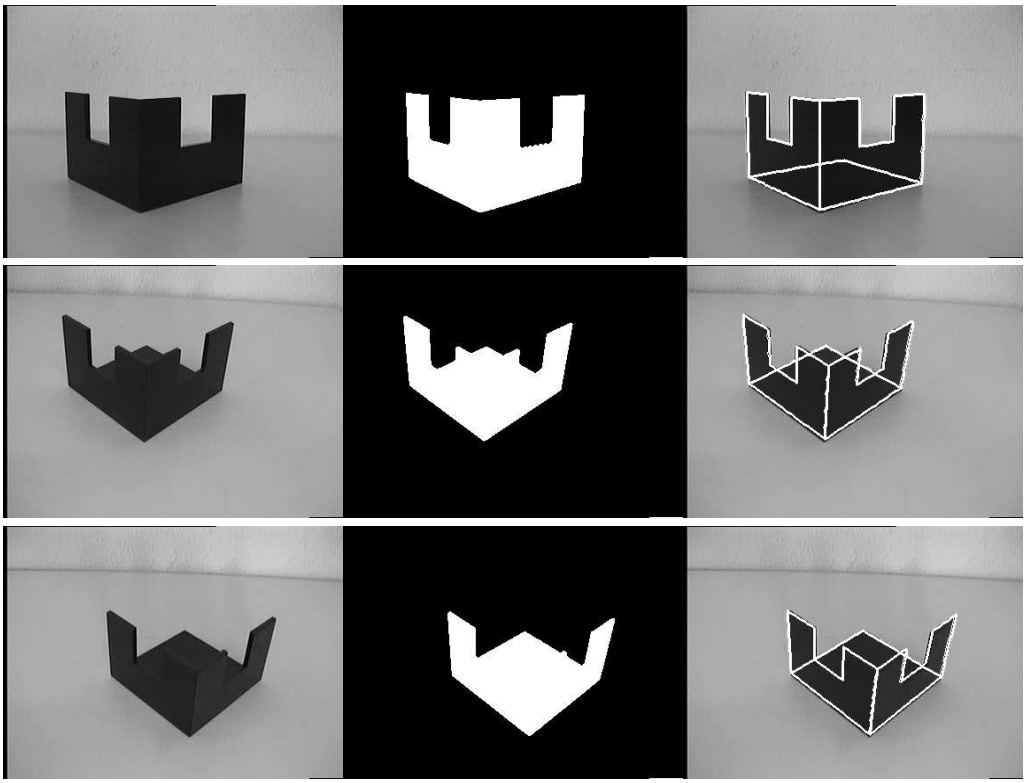

Fig. 4.14: Pose results of an object with partially occluded contours. The left image shows the original image. The middle image shows the extracted silhouette (from which the boundary contour is extracted) and the right image visualizes the pose result. Note that also the occluded parts of the model are drawn and uniquely determined by the visible parts.

```
(d) Use (P,R) as correspondence set.
(e) Estimate pose with this correspondence set
(f) Transform contours, goto (b)
```

The idea is to apply our ICP-algorithm not to one image contour and one 3D contour, but now to one image contour and a set of 3D contours.

This implies: For each extracted image point must exist one model contour and one point on this contour, which corresponds to this image point. Note, that the reverse is in general not possible.

Figure 4.14 visualizes the problem of partially occluded contour points. The only image information we use is the observed boundary contour of the object. By using a priori knowledge (e.g. assuming a tracking assumption), the pose can be recovered uniquely. This means, our algorithm can infer the

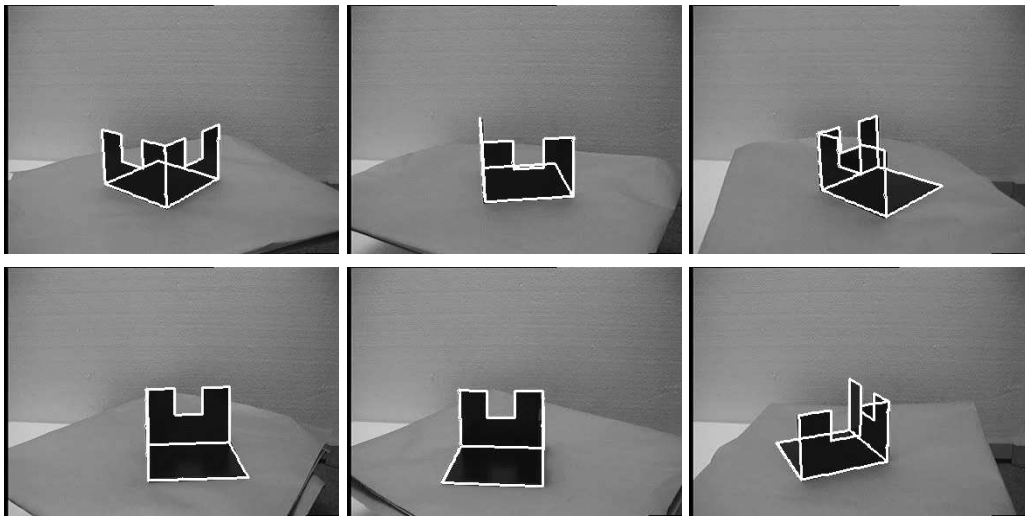

Fig. 4.15: Pose results of an image sequence containing different aspect changes and degenerate situations.

position of hidden components from the visible components.

The computing time is proportional to the number of used contours. While we need 200ms for each image in the experiments of figure 4.12, we now need 600ms for each image. But we gain a more general concept, since we are not restricted to one special view to the object. Instead we can deal with aspect changes of the contour in an efficient manner. This is demonstrated in figure 4.15 in case of quiet different aspects of a 3D object. The images are taken from an image sequence containing 325 images. In this image sequence we put the object on a turn table and make a 360^o degree turn of the whole object. The aspects of the objects are changing and half-side models can not be used any more, but just the whole object. Our tracking algorithm does not fail and is even able to cope with degenerate situations.

In our last experiment, we use as object model the shape of a 3D tree. The contour approximations are shown in figure 4.16. As can be seen in the close-up, here also much descriptors (around 50) are needed to get a sufficient approximation of the model. Pose results of an image sequence, containing 735 images are shown in figure 4.17. The interesting part of this model, in contrast to the previous ones, is not only its complexity: This model contains two *nested* contours and is therefore much more complicated than the previous ones. Because of its complexity (the number of Fourier coefficients and the nested contours) we need a computing time of three seconds for each image on a 2 Ght Linux machine.

From the two last experiments result the possibility to model objects with

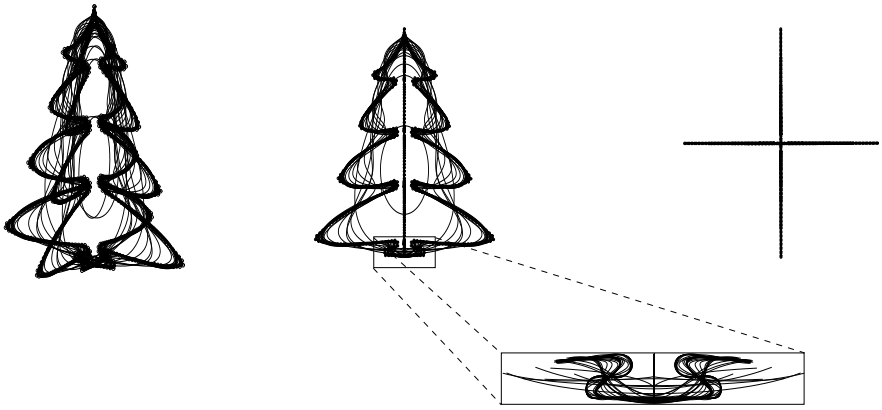

Fig. 4.16: One perspective, frontal and top view with approximations of the tree model. The close up visualizes the complexity of the object model.

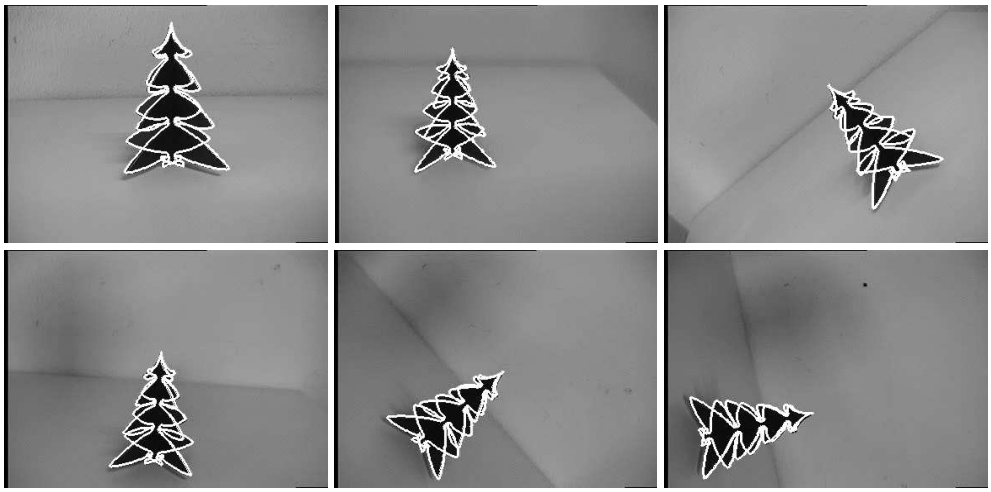

Fig. 4.17: Pose results of the tree model during an image sequence.

contours representing different aspects of the object and to fuse these within our scenario.

5. DISCUSSION

This work concerns the problem of 2D-3D pose estimation of 3D free-form contours as one single geometric entity. We assume the knowledge of a 3D object, which contains one or more contours modeling the aspects of the object. Furthermore we assume a calibrated camera and observe the silhouette of the object in the camera. The aim is to estimate the pose (rotor $\boldsymbol{R}$ and translator $\boldsymbol{T}$), which leads to a best fit between image and model data. The first topic we concern is the modeling of 3D free-form contours. Therefore, we start with algebraic curves (the cycloidal curves) and model them by coupled twists. We then derive the connection of twist-generated curves with Fourier descriptors and explain how to estimate 3D closed contours by using 3D Fourier descriptors. In contrast to an explicit or implicit definition definition of algebraic curves, we propose an operational definition which keeps geometric transparency and fits within the scenario of pose estimation in a conformal manner. The second topic we concern is the coupling of this curve representation within the 2D-3D pose estimation problem. Therefore we use the conformal geometric algebra, which contains also Euclidean and projective geometry as sub-algebras. This representation is used to compare the 3D contour with (from image points) reconstructed projection rays. This leads to constraint equations, which are solved by using a gradient descend method, combined with an ICP-algorithm. The formulas are compact and easy to interpret.

The last section concerns experiments which show the efficiency of our algorithm on different image sequences with different object models (planar, non-planar, curved, angular etc.). We discuss the behavior of our algorithm with respect to boundary distortions and the complexity of object models. Besides, we investigate the time performance on different machines. Furthermore, we deal with partially occluded object features and nested contours. Using Fourier descriptors allows us to deal with low-pass information of contours and therefore to reduce the computing time significantly. Our algorithm proofs as real-time capable, efficient and robust.

APPENDIX

A. POSE ESTIMATION OF PROJECTED CONTOURS

Here we give some mathematical detail on how the Fourier descriptors of a closed curve are related to the Fourier descriptors of a projection (not affine) of the same closed curve.

As was shown before we can write an arbitrary closed planar curve as

$$g(\theta) = \sum_u \boldsymbol{R}_u \, \boldsymbol{p}_u \, \widetilde{\boldsymbol{R}}_u,$$

where $\boldsymbol{R}_u = \exp(-\frac{1}{2}\theta u \boldsymbol{l})$, $\theta = 2\pi\phi/T$ and $u \in \mathbb{Z}$. The set $\{\boldsymbol{p}_u\}$ are the phase vectors. We assume that this contour lies in the $\boldsymbol{e}_1\boldsymbol{e}_2$-plane of a 3D-Euclidean vector space with basis $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$. Typically, we also make the assumption that $g(\theta)$ can be represented by a finite number of phase vectors.

Now we rotate and translate the whole curve $g(\theta)$ and denote the new curve as $g'(\theta)$. Our goal is to recover the rotation and translation of the original curve $g(\theta)$ from the original curve itself and the projection of $g'(\theta)$ onto an image plane. Without loss of generality we can set the image plane to be perpendicular to $\boldsymbol{e}_3$ and passing through $\boldsymbol{e}_3$. In order to see what the projection of $g'(\theta)$ onto this image plane looks like, we need to write $g'(\theta)$ in terms of the vector basis and translation and rotation coefficients. It turns out that $g'(\theta)$ can be written as

$$g'(\theta) = (g'_0(\theta) + \boldsymbol{t}^{\parallel}) + \Big(\sum_{u \neq 0} g'_u(\theta) \Big) + \boldsymbol{t}^{\perp}, \tag{1.1}$$

where $g'_0(\theta)$ is a constant vector in the $\boldsymbol{e}_1\boldsymbol{e}_2$-plane, $\boldsymbol{t}^{\parallel}$ is the translation vector of the curve in the $\boldsymbol{e}_1\boldsymbol{e}_2$-plane and $\boldsymbol{t}^{\perp}$ is the translation vector of the curve along the $\boldsymbol{e}_3$-axis. The components $g'_u(\theta)$ can be written as

$$g'_u(\theta) = \cos(u\theta)\, \boldsymbol{c}_u^1 + \sin(u\theta)\, \boldsymbol{c}_u^2 + \Big(\alpha_u \cos(u\theta) + \beta_u \sin(u\theta)\Big)\, \boldsymbol{e}_3,$$

where $\boldsymbol{c}_u^1$ and $\boldsymbol{c}_u^2$ are vectors in the $\boldsymbol{e}_1\boldsymbol{e}_2$-plane and $\alpha_u, \beta_u \in \mathbb{R}$ are scalar factors. These vectors and the scalars depend on the phase vectors for frequency u of the original curve $g(\theta)$ and on the rotation angles we want to

recover. In order to project $g'(\theta)$ onto the image plane we simply write

$$g'_P(\theta) = \frac{(g'(\theta) \cdot \boldsymbol{e}_1)\boldsymbol{e}_1 + (g'(\theta) \cdot \boldsymbol{e}_2)\boldsymbol{e}_2}{g'(\theta) \cdot \boldsymbol{e}_3} + \boldsymbol{e}_3. \tag{1.2}$$

The curve $g'_P(\theta)$ is what we observe in the image plane. The question now is, how the Fourier coefficients we obtain from the observed curve $g'_P(\theta)$ are related to the Fourier coefficients of the original curve $g(\theta)$ in terms of the rotation and translation parameters we are looking for. As we mentioned before, here we have the additional problem that in general we cannot know how to sample the observed curve $g'_P(\theta)$, so that this sampling is equivalent to the sampling of $g(\theta)$. However, suppose for the moment that we could somehow overcome this problem. Then we are faced with evaluating the Fourier coefficient of $g'_P(\theta)$. Let us first of all consider evaluating the Fourier coefficients for a particular frequency u. This leads to integrals of the form

$$I_v = \int_0^{2\pi} \frac{\cos(ux)\, e^{-ivx}}{a\cos(ux) + b\sin(vx) + c}\, dx,$$

where I_v is proportional to the Fourier coefficient for frequency v. By making the substitutions $\cos(ux) = \exp(iux) + \exp(-iux)$ and $\sin(ux) = \exp(iux) - \exp(-iux)$ and reordering the terms we find

$$I_v = \int_0^{2\pi} \frac{\left(1 + e^{-i2ux}\right) e^{-ivx}}{we^{-i2ux} + 2c\, e^{-iux} + w^*}\, dx,$$

where $w = a + ib$ and $w^* = a - ib$. We can solve this integral by using the method of residues. To do that we make the substitution $z = \exp(iux)$ and then integrate over the unit circle C in the complex plane. The integral then becomes

$$I_v = \frac{1}{w} \int_C \frac{(1 + z^2)\, z^{(v/u)-1}}{z^2 + n^*\, z + m^*}\, \frac{dz}{iu}, \tag{1.3}$$

where $n^* = 2c/w$ and $m^* = w^*/w$. The solution to this integral is given by the residue formula. The residues can be found by finding the roots of the denominator of equation (1.3). For the present case this is straight forward since we only have a quadratic equation in the denominator. However, recall that equation (1.2) has a sum of terms for different frequencies in the denominator. Therefore, in order to find the Fourier coefficients of $g'_P(\theta)$ in general, we would need to solve for the roots of polynomials of arbitrary degree analytically. Since this is not possible, we cannot use this approach to recover the translation and rotation components we were looking for.

BIBLIOGRAPHY

[1] Arbter K. Affine-invariant fourier descriptors. In *From Pixels to Features. Simon J.C. (Ed.)* Elsevier Science Publishers, pp. 153-164, 1989.

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

[3] Arbter K., Snyder W.E., Burkhardt H. and Hirzinger G. Application of affine-invariant fourier descriptors to recognition of 3-D objects *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 12, No. 7, pp. 640-647, 1990.

[4] 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.

[5] Bayro-Corrochanno E., Daniilidis K., and Sommer G. Motor algebra for 3D kinematics. The case of the hand–eye calibration. *Journal of Mathematical Imaging and Vision, JMIV*, Vol. 13, No. 2, pp. 79-99, 2000.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

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

[12] Czopf A., Brack C., Roth M. and Schweikard A. 3D-2D registration of curved objects. *Periodica Polytechnica*, Vol. 43, No. 1, pp. 19-41, 1999.

[13] 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.

[14] Faugeras O. Stratification of three-dimensional vision: projective, affine and metric representations. *Journal of Optical Society of America*, Vol.12, No.3, 1995.

[15] Fenske A. Affin-invariante Erkennung von Grauwertmustern mit Fourierdeskriptoren. *Mustererkennung 1993*, pp. 75-83, Springer-Verlag, 1993.

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

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

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

[19] Hestenes D. Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, No.7, pp.65–77, 1994.

[20] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [45], pp. 3-23, 2001.

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

[22] Kauppinen H., Seppänen T. and Pietikäinen M. An experimental comparison of autoregressive and fourier-based descriptors in 2D shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 17, No. 2, pp. 201-207, 1995.

[23] 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.

[24] Huber D.F. and Hebert M. Fully automatic registration of multiple 3D data sets. *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum(CVBVS 2001)*, 2001.

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

[26] 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

[27] Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [45], pp. 27-52, 2001.

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

[29] Lin C.-S. and Hwang C.-L. New forms of shape invariants from elliptic fourier descriptors. *Pattern Recognition*, Vol. 20, No. 5, pp. 535-545, 1987.

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

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

[32] McCarthy J.M. Introduction to Theoretical Kinematics *MIT Press, Cambridge, Massachusetts, London, England*, 1990.

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

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

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

[36] Perwass C. and Lasenby L. A unified description of multiple view geometry. In [45], pp. 337-369, 2001.

[37] Reiss T.H. Recognizing Planar Objects Using Invariant Image Features. LNCS 676, *Springer Verlag*, 1993.

[38] Rooney J. A comparison of representations of general spatial screw displacement In *Environment and Planning B*, Vol. 5, pp. 45-88, 1978.

[39] Rosenhahn B., Zhang Y. and Sommer G. Pose estimation in the language of kinematics. In: *Second International Workshop, Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Sommer G. and Zeevi Y.Y. (Eds.)* , LNCS 1888, Springer-Verlag, Heidelberg, pp.284-293, 2000.

[40] 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.

[41] 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, 2002.

[42] Rosenhahn B. and Sommer G. Pose estimation from different entities. Part I: The stratification of mathematical spaces. Part II: Pose constraints. *Technical Report 0206*, University Kiel, 2002.

[43] Rusinkiewicz S. and Levoy M. Efficient variants of the ICP algorithm. Available at http://www.cs.princeton.edu/ smr/papers/fasticp/. Presented at *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

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

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

[46] Stark K. A method for tracking the pose of known 3D objects based on an active contour model. *Technical Report TUD / FI 96 10*, TU Dresden, 1996.

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

[48] 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.

[49] 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.

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

[51] 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.

[52] Zahn C.T. and Roskies R.Z. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, Vol. 21, No. 3, pp. 269-281, 1972.