Algorithmica (2004) 38: 91–113
DOI: 10.1007/s00453-003-1044-3

Algorithmica

Free-Form Pose Estimation by Using Twist Representations[1]

Bodo Rosenhahn,[2] Christian Perwass,[2] and Gerald Sommer[2]

Abstract. In this article we discuss the 2D–3D pose estimation problem of 3D free-form contours. We observe objects of any 3D shape in an image of a calibrated camera. Pose estimation means estimating the relative position and orientation of the 3D object to the reference camera system. While cycloidal curves are derived as orbits of coupled twist transformations, we apply a spectral domain representation of 3D contours as an extension of cycloidal curves. Their Fourier descriptors are also related to twist representations. A twist is an element of $se(3)$ and is a pair containing two 3D vectors. In a matrix representation, its exponential leads to an element of $SE(3)$ and therefore to a rigid motion. We show that twist representations of objects can numerically efficiently and easily be applied to the free-form pose estimation problem. The pose problem itself is formalized as an implicit problem and we gain constraint equations, which have to be fulfilled with respect to the unknown rigid body motion.

Key Words. 2D–3D pose estimation, Free-form curves, Twists, Fourier descriptors, Cycloidal curves, ICP.

1. Introduction. This contribution concerns the 2D–3D pose estimation problem of 3D free-form curves. Pose estimation itself is one of the oldest computer vision problems and algebraic solutions with different camera models have been proposed for several variations of this problem. Pioneering work was done in the 80s and 90s by Lowe [30], [31], Grimson [17] and others. In their work point correspondences are used. More abstract entities can be found in [25], [51], [26], [22], [48], and [7]. Entities discussed are circles, cylinders, kinematic chains or other multipart curved objects. Works concerning free-form curves can be found in [12] and [45]. In these works contour point sets, affine snakes or active contours are used for visual servoing. An overview of free-form object representations is given in [9]. 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 separated in characteristic object features (like edges or corners, etc.) and then applied to the different problems. Secondly, the object can be modeled as itself, e.g. in the 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. However, especially in natural environments with curved shapes and surfaces, feature extraction and matching is a problem. Then there

[1] This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).
[2] Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany. {bro,chp,gs}@ks.informatik.uni-kiel.de.

Received May 1, 2002; revised January 21, 2003. Communicated by R. C. Veltkamp.
Online publication October 24, 2003.

is a need to deal with an object as a whole, or one single entity and not features of the object. We call such objects free-form objects, as a general class of objects and quote Besl [6] for a definition: *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, airplanes, human faces, organs or terrain maps are typical examples for free-form objects.

The main problem we are concerned with is the algebraic coupling of free-form contours within the pose estimation problem. Therefore we use as a link between these different topics the *twists*. Twists are well known from Lie groups and Lie algebras and are mostly applied to rigid body motions. In this work we use twists on the one hand within our pose estimation problem, and on the other hand to model the object contours. This unification enables a compact description of the pose 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 (mostly 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. ICP algorithms are mostly applied to 2D or 3D point sets. Instead, we later use it for comparison of a trigonometric interpolated function with reconstructed projection rays. Different works concerning ICP algorithms can be found in [42], [11], [23], and [50].

To solve the pose problem of free-form contours, we start with the pose estimation problem for entities like points, lines and planes. Then we consider 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 [27]. We 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 interpolation 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 description of the contour for an approximation, leading to a speed up of the algorithm. As shown later, 3D cycloidal curves are strongly connected to Fourier descriptors [2], [1], [16] as a 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 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. 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 curves.

1.1. *Preliminary Work.* Our recent work [38]–[41] can be summarized in the scenario of Figure 1. 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. Further, we assume extracted corresponding features in an image of a calibrated camera. The

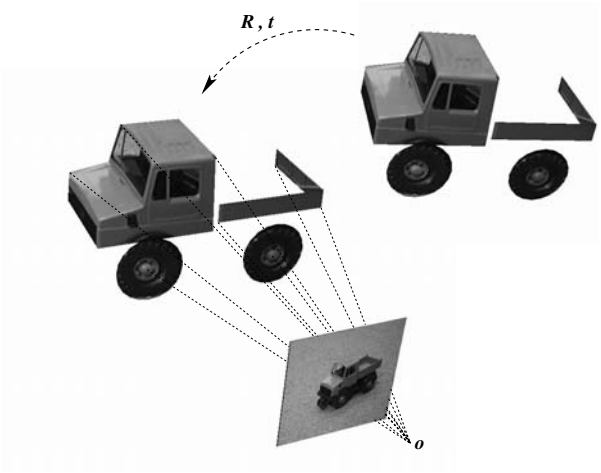

Fig. 1. The scenario. The assumptions are the camera model, the model of the object (consisting of points, lines, circles and kinematic chains) and corresponding extracted entities on the image plane. The aim is to find the pose $(\boldsymbol{R}, \boldsymbol{t})$ of the model, which leads to the best fit of the object with the entities actually extracted.

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 entities actually extracted. 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 through projective reconstruction from the image entity. This idea will be used to formulate the scenario in three dimensions. We are therefore working in a kinematic framework. Note that after reconstruction the reconstructed entities are independent from the camera.

As mentioned before, there exist many scenarios in which it is not possible to extract point-like features as corners or edges, but only general contours. Besides, there exist 3D objects which cannot be adequately represented by primitive object features such 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 [27]. There exist many ways to define algebraic curves. 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 an intersection of a cone with a plane. 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 elements 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 [10].

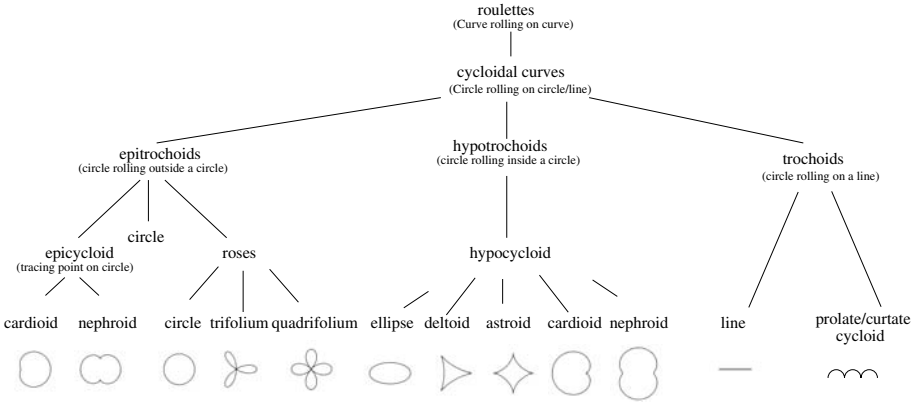

Fig. 2. Tree of algebraic curves.

Here we concentrate on a subclass of the *roulettes*, the *cycloidal curves*, which are circles rolling on circles or lines. Figure 2 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 2 also 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. Indeed it is possible to generalize cycloidal curves to for example 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 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 interpolate functions trigonometrically, we then have the direct link to use cycloidal curves for any free-form contour. Note that these curves are mostly defined in the 2D plane. For our scenario of pose estimation, we extend these curves to 2D or 3D curves in 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 is the involved mathematical spaces which are elements of the stratification hierarchy proposed by Faugeras [13]. On the one hand we are interested in estimating 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 both projective and kinematic geometry in one language. We 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 3D space. This part is also possible in classical affine geometry, but in

the next section we then combine this formalization within our pose estimation problem and then it is necessary to use conformal geometry: The image entities are projectively reconstructed to, e.g. projection rays and then transformed into conformal lines (Plücker lines [37]). So the projective aspect of the pose problem is transformed into a kinematic one and then combined within the kinematic description of the cycloidal curves and their pose.

2.1. *Introduction to Conformal Geometric Algebra.* In this section we introduce the main properties of conformal geometric algebra (CGA) [28]. The aim is to clarify the notations. A more detailed introduction to geometric algebras can be found in [44].

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 with vectors of a vector space as first-order entities. A geometric algebra $\mathcal{G}_{p,q}$ 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 decrease the grade of the algebraic entities, respectively. We demonstrate this effect in the case of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{G}_{p,q}$:

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

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

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

is an inhomogeneous multivector.

For later use we introduce the commutator $(\underline{\times})$ and anticommutator $(\overline{\times})$ products for any two multivectors,

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

The reader should consult [34] or [35] to become more familiar with the commutator and anticommutator product. Their role is to separate the symmetric part of the geometric product from the antisymmetric one.

To introduce CGA, we follow [28] and start with the *Minkowski plane* $\mathcal{G}_{1,1}$, which has an orthonormal basis $\{\mathrm{e}_+, \mathrm{e}_-\}$, defined by the properties

$$\mathrm{e}_+^2 = 1, \qquad \mathrm{e}_-^2 = -1 \quad \text{and} \quad \mathrm{e}_+ \cdot \mathrm{e}_- = 0.$$

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

$$\mathrm{e}_0 := \tfrac{1}{2}(\mathrm{e}_- - \mathrm{e}_+) \quad \text{and} \quad \mathrm{e} := \mathrm{e}_- + \mathrm{e}_+.$$

The vector e_0 can be interpreted as the origin, and the vector e as a point at infinity. Furthermore, we define $\boldsymbol{E} := \mathrm{e} \wedge \mathrm{e}_0$.

In the case of an n-dimensional vector space the Minkowski model $\mathcal{G}_{n+1,1}$ will be used, therefore enlarging the geometric algebra of the n-dimensional vector space by two additional basis vectors, which define a *null space*. For the 3D vectors space $\mathbb{R}^3$ we gain $\mathcal{G}_{4,1}$, which contains $2^5 = 32$ elements.

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

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

both algebras for the projective and Euclidean space constitute subspaces of the linear space of 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, containing the Euclidean, affine and projective space.

The basis entities of the 3D conformal space are spheres $\underline{\boldsymbol{s}}$, containing the center $\boldsymbol{p}$ and the radius ρ, $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathrm{e} + \mathrm{e}_0$. A point $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathrm{e} + \mathrm{e}_0$ is nothing else but a degenerate sphere with radius $\rho = 0$, which can easily be seen from the representation of a sphere. A point $\underline{\boldsymbol{x}}$ is on a sphere $\underline{\boldsymbol{s}}$ iff $\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} = 0$. Since the following relationship holds (see, e.g. [18]),

$$\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} = 0 \quad \Leftrightarrow \quad \underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star = 0,$$

we can also use the dual representation (denoted with a *) of entities. Dual points, lines and planes can be expressed as $\underline{\boldsymbol{X}}^\star = \mathrm{e} \wedge \underline{\boldsymbol{x}}$, $\underline{\boldsymbol{L}}^\star = \mathrm{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{P}}^\star = \mathrm{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$. Note that since we later work with the entities in their dual representation, we neglect the * in the following.

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

2.2. *Rigid Body Motions of Geometric Entities and Twists in CGA*. In this section the estimation of rigid body motions in CGA is discussed. 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. A rigid motion can be separated in a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$. 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 $\boldsymbol{R} = \exp(-(\theta/2)\boldsymbol{l})$. 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 the rotation. This is similar to the use of quaternions for estimating rotations. 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*, $\boldsymbol{T} = (1 + \mathrm{e}\boldsymbol{t}/2) = \exp(\mathrm{e}\boldsymbol{t}/2)$. This translator is a special rotor, similar to a translator in the dual quaternion algebra. The main difference of CGA to quaternions and dual quaternions [5] is that there is no need to transform the entities in another space to apply geometric transformations. This means for example that we do not have to encode

a point as a quaternion, compute quaternionic multiplications and transform it back to a vector. Instead the entities remain in their natural representation.

Rotations and translations can be estimated by applying rotors and translators as versor products [20], e.g. $\underline{\boldsymbol{X}}' = \boldsymbol{R}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{R}}$ or $\underline{\boldsymbol{X}}'' = \boldsymbol{T}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{T}}$.[3] To express a rigid body motion, we can apply multiplied rotors and translators consecutively. We denote such an operator (it is a special even-grade multivector) as a motor $\boldsymbol{M}$, which is an abbreviation of "moment and vector". The rigid body motion of for example a point $\underline{\boldsymbol{X}}$ can be written as $\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$, see also [5]. However, as mentioned before, this does not only hold for point concepts. Other entities like lines, plane, circles and spheres can be transformed in the same manner.

Following, e.g. [32], [43], and [33], a rigid body motion of points can be expressed by a rotation around a line in space followed by a translation along this line. This results from the fact that for every $g \in SE(3)$ there exists a $\xi \in se(3)$ and a $\theta \in \mathbb{R}$ such that $g = \exp(\xi\theta)$. Such transformations 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 [15]. A motor describing a twist transformation can be written as

$$\begin{aligned} \boldsymbol{M} &= \exp\left(-\frac{\theta}{2}\,(\boldsymbol{l} + \mathrm{e}\boldsymbol{m})\right) \\ &= \exp\left(-\frac{\theta}{2}\Psi\right). \end{aligned}$$

A twist can be seen as an infinitesimal version of a screw motion and describes a line in space with an angle θ and a *pitch* h, the ratio of translation to rotation. If the pitch h is zero, the resulting motion is a rotation of an entity (e.g. a point $\underline{\boldsymbol{X}}$) around a line $\underline{\boldsymbol{L}}^{\star}$ in the space. To gain a twist representation, the general idea 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 $\boldsymbol{M}$ can be interpreted as the exponential of a twist, with the form

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

The motion of a point can then be decomposed as

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

We call such a transformation a *general rotation*. Whereas in Euclidean geometry, Lie algebras and Lie groups are only applied to point concepts, motors can also be applied to other entities, like lines, planes, circles, spheres, etc. 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 to other entities.

[3] $\widetilde{A}$ denotes the reverse of A and $A^{\star}$ denotes the dual representation of A.

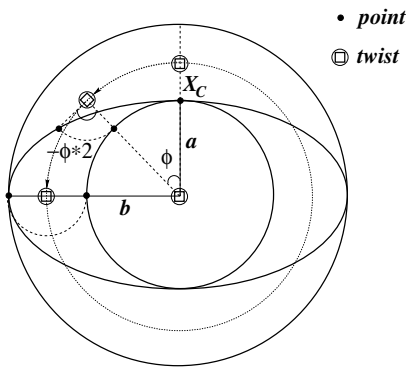

Fig. 3. A conic generated by two coupled twists.

2.3. *Cycloidal Curves*. While in the last section we stated that twists can be considered as generators of the Lie group of rigid body motions for a certain set of entities, 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 explain how to generate such curves as twist depending functions in CGA. For example, conics are not entities which can be directly described in CGA. The idea for modeling conics is visualized in Figure 3: We assume two parallel twists modeling general rotations 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 $\varphi \in [0 \cdots 2\pi]$ generates a conic as the orbit of the generated 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. By nD-$m$twist curves we mean n-dimensional curves, generated by m twists with $n, m \in \mathbb{N}$. Note, to model 3D cycloidal curves we restrict the twists to modeling general rotations, but indeed twist curves are more general than cycloidal curves. In the context of the 2D–3D pose estimation problem we use the curves as 3D object entities. So we mean 3D-$m$twist curves if we speak of just $m$twist curves.

We start with very simple cycloidal curves. The simplest one consists of one point (a point on the curve) and one twist modeling a general rotation. Rotating the point around the twist leads to the parameterized generation of a circle: the transformation can be expressed with a suitable motor $\boldsymbol{M}_\varphi$ 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 on is simply given by

$$\underline{\boldsymbol{X}}_Z^\varphi = \boldsymbol{M}_\varphi \underline{\boldsymbol{X}}_Z \widetilde{\boldsymbol{M}}_\varphi, \qquad \varphi \in [0 \cdots 2\pi].$$

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

Now we can continue and wrap a second twist, also modeling a general rotation, 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 2D curves in 3D space, whereas we get 3D curves in 3D space for nonparallel twist axes.

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

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

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 4 shows further examples of curves, which can be very easily generated by two coupled twists. Note that the archimedic spiral is also a 2twist generated curve. To gain an archimedic spiral, one twist has to be a translator. All these curves are given in 3D space. In Figure 4 only projections are shown. Table 1 gives an overview of some well-known entities, interpreted as twist generated curves as well as twist generated surfaces.

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

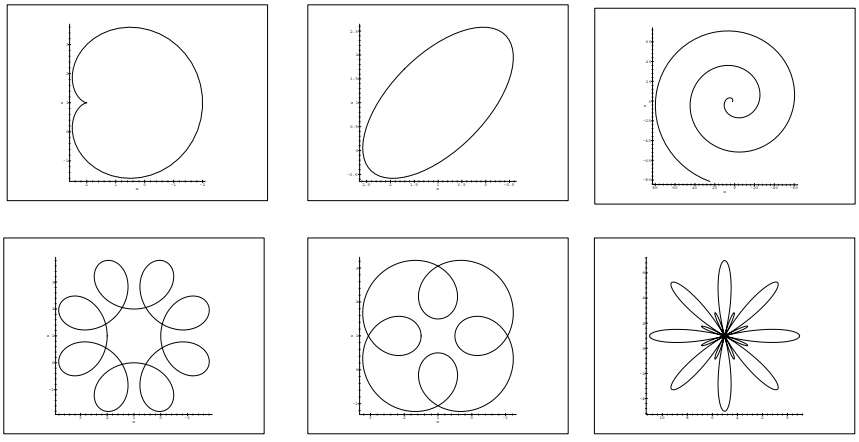

Fig. 4. Curves generated from 3D-2twists with parallel axes.

Table 1. Well known 3D entities as $m$twist curves or surfaces.

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

3. Estimating Twists from a Given Closed Curve. So far we have discussed how a set of multiplicatively coupled twists, modeling general rotations, 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 [16], [49], [3], [24] and affine pose estimation [3], [36] of closed contours. We will show here that a set of coupled twists, modeling general rotations, 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

$$\boldsymbol{R}_i^{\varphi} := \exp\left(-\frac{\pi u_i \varphi}{T}\,\boldsymbol{l}\right),$$

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, $\widetilde{\boldsymbol{R}}_i^{\varphi} = \exp(\pi u_i \varphi / T\,\boldsymbol{l})$. Recall that $\boldsymbol{l}^2 = -1$ and we can therefore write the exponential function as

$$\exp(\varphi\,\boldsymbol{l}) = \cos(\varphi) + \sin(\varphi)\boldsymbol{l}.$$

For two twists modeling general rotations, a 2twist generated curve may then be written in Euclidean space as follows:

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

where $\boldsymbol{p}_0 \equiv \boldsymbol{t}_2$, $\boldsymbol{p}_1 \equiv \boldsymbol{t}_1 - \boldsymbol{t}_2$, $\boldsymbol{p}_2 \equiv \boldsymbol{x}_C - \boldsymbol{t}_1$, $\boldsymbol{V}_1^{\varphi} \equiv \boldsymbol{R}_2^{\varphi}$, $\boldsymbol{V}_2^{\varphi} \equiv \boldsymbol{R}_2^{\varphi}\boldsymbol{R}_1^{\varphi}$ and $\lambda_i = 2\pi u_i / T$. Note that for planar curves the rotors $\boldsymbol{R}_1^{\varphi}$ and $\boldsymbol{R}_2^{\varphi}$ 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

$$\boldsymbol{x}_C^{\varphi} = \boldsymbol{p}_0 + \boldsymbol{p}_1\,\widetilde{\boldsymbol{V}}_1^{2\varphi} + \boldsymbol{p}_2\,\widetilde{\boldsymbol{V}}_2^{2\varphi}.$$

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

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

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 $C(\varphi)$ can be expressed as a series expansion

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

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.

The expansion of the previous equation is again closely related to the standard Fourier series expansion of a real, scalar-valued function. In Figure 5 a closed curve created by two coupled twists is shown in the yz-plane. Suppose that instead of $C(\varphi)$ we consider $C_S(\varphi) := C(\varphi) + 2\pi\varphi/T\,\boldsymbol{e}_1$, where $\boldsymbol{e}_1$ is the unit vector along the x-axis. If we project $C_S(\varphi)$ onto the xy-plane and xz-plane, we obtain the other two 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 [16], [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 [4], [14], [46]. The same relations that allow one to construct affine invariant Fourier descriptors also allow for affine pose estimation. This works in the following way. Consider a closed curve that lies on a plane which is tilted with respect to an observer. This curve is projected with an affine camera onto an image plane. The pose of the plane in space can then be estimated 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, 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 nth degree polynomials. Since this is not possible in general, we cannot follow this approach. We therefore investigated a different approach for the pose estimation of projected closed curves, which are discussed in the following.

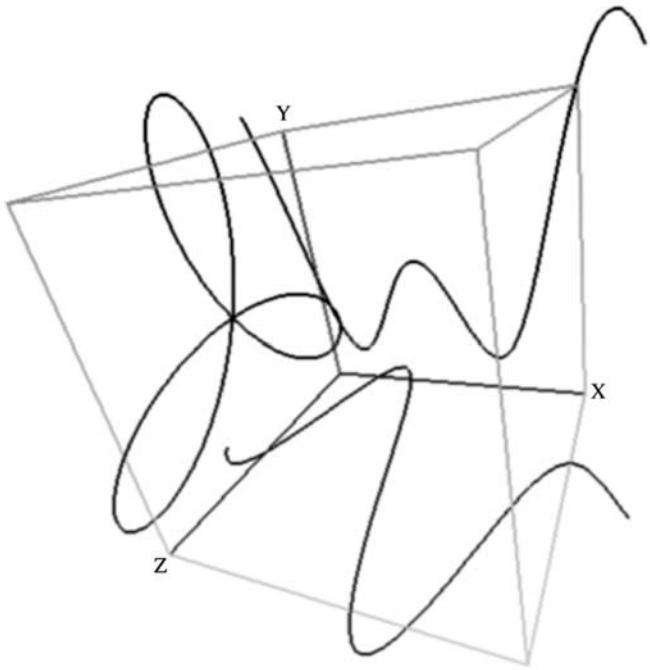

Fig. 5. Projections of a curve created by coupled twists.

4. 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 continue to embed these entities in the 2D–3D pose estimation problem.

4.1. *Pose Estimation in Stratified Spaces*. To define the pose problem we quote Grimson [17]: *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 them. 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 conformal algebra, because we formalize the pose estimation problem in 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 $\boldsymbol{M}\underline{\boldsymbol{X}}\tilde{\boldsymbol{M}}$. Let $\boldsymbol{x}$ be an image point on a projective plane. The projective reconstruction from an image point in CGA can be written as $\underline{\boldsymbol{L}}_x = \mathrm{e} \wedge \boldsymbol{o} \wedge \boldsymbol{x}$. This leads to a reconstructed projection ray, containing the optical center $\boldsymbol{o}$ of the camera, see, e.g. Figure 1, the image point $\boldsymbol{x}$ and the vector e as the point at infinity. Note that $\boldsymbol{o} \wedge \boldsymbol{x}$ formalizes the reconstructed ray in projective geometry. The expression $\mathrm{e} \wedge \boldsymbol{o} \wedge \boldsymbol{x}$ represents the reconstructed ray in conformal geometry and is therefore given in the same language that 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 contains a distance measure in the Euclidean space (see, e.g. [39] for the proofs). Thus, the constraint equation of pose estimation from image points reads

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

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 a 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. However, in contrast to other approaches, where the minimization of errors has to be computed directly on the manifold of the geometric transformations [8], [47], 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.

4.2. *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

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

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

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

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 parentheses on the left contain the parameterized generation of the cycloidal curve. The outer parentheses contain 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 $\boldsymbol{M}$ and the angle φ for each point correspondence. An example of pose estimation of cycloidal curves is shown in Figure 6. 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.

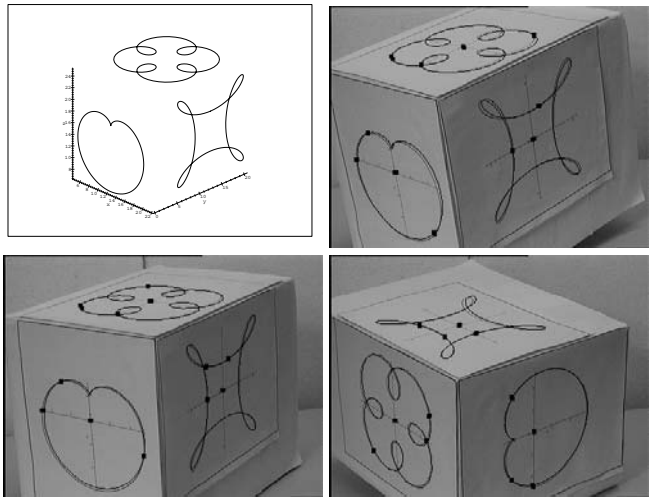

Fig. 6. Pose estimation of an object, containing a cardioid and two cycloids.

4.3. *Pose Estimation of Free-Form Contours*. So far we have 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

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

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

$$\mathrm{e} \wedge (C(\varphi) + \mathrm{e}_-) = \mathrm{e} \wedge \left(\left(\sum_{k=-N}^{N} \boldsymbol{R}_k^{\varphi}\, \boldsymbol{p}_k\, \widetilde{\boldsymbol{R}}_k^{\varphi} \right) + \mathrm{e}_- \right).$$

Substituting this expression into the pose constraint equation leads to

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

The interpretation of this equation is also simple: The innermost parentheses contain the Fourier descriptors in the Euclidean space. The next parentheses transform this

expression in the homogeneous space and then in the conformal space. Afterwards it is coupled with the unknown rigid body motion (the motor $\boldsymbol{M}$) and compared with a reconstructed projection ray, also given in conformal space.

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

4.4. *Estimation of Pose Parameters*. The main question now is how to solve a set of constraint equations for multiple (different) features with respect to the unknown motor $\boldsymbol{M}$. Since a motor is a polynomial of infinite degree (see, e.g. its series expression), this is a nontrivial task, especially in the case of real-time estimations. Furthermore, we have multivector equations and so far there exists no *multivector-SVD* or any other numerical algorithm for solving such Clifford equations. This is the reason why we now return to matrix calculus by linearizing (and iterating) the equations. The idea is to gain linear equations with respect to the generators of the motor. We use the exponential representation of motors and expand the motors with a Taylor series up to first order. This leads to a mapping of the above-mentioned global motion transformation to a twist representation, which enables incremental changes of pose. That means, we do not search for the parameters of the Lie group $SE(3)$ to describe the rigid body motion [15], but for the parameters which generate their Lie algebra $se(3)$ [33]. This leads to linear equations in the generators of the unknown 3D rigid body motion. For the sake of simplicity we 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:

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

Setting $\boldsymbol{l} := \theta\boldsymbol{l}'$ and $\boldsymbol{m} := \theta\boldsymbol{m}'$ leads to

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

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

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

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

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 $\theta := \|\boldsymbol{l}\|, \boldsymbol{l}' := \boldsymbol{l}/\theta$ and $\boldsymbol{m}' := \boldsymbol{m}/\theta$. Once the six twist parameters $\theta, \boldsymbol{l}'$ and $\boldsymbol{m}'$ are determined from $\boldsymbol{l}$ and $\boldsymbol{m}$, the group action can be recovered by applying the famous Rodrigues' formula (1840) [15].

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 five to eight) are sufficient to get a good approximated pose result. The algorithm itself corresponds to a gradient descent method applied in 3D space.

5. Experiments. In this section we present experimental results of free-form contour pose estimation.

So far we have identified the connection of 3D $m$twist cycloidal curves and the discrete Fourier transformation. The aim is now 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 their phase coefficients $\boldsymbol{p}_k$ are known.
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 subsample the point set to get a uniformly 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 the image points to projection rays.
(b) Estimate the nearest point of each image point 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 algorithms, e.g. discussed in [42] and [50]. However, whereas it is mostly applied to sets of 2D or 3D points we apply it to a trigonometric interpolated function and from image points reconstructed projection rays.

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 a 25 pixel deviation. A projection of the object model used is shown in Figure 7. The discrete points and the different approximation levels are shown. The model itself consists of 90 contour points, is planar and has a width and height of

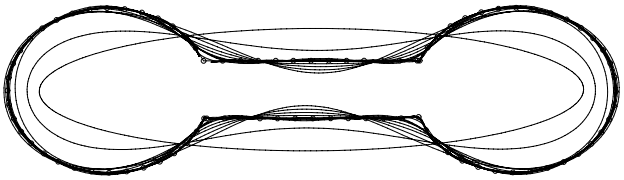

Fig. 7. The different approximation levels of the 3D object contour.

24 × 8 cm. Pose estimation results at different iterations are shown in Figure 8. The white 2D contour is the transformed and projected 3D object model overlaid with the image. Note that the iteration number shows the number of ICP cycles. Since the pose estimation itself is performed by an iteration procedure, we have in the algorithm two nested loops. However, as mentioned before, the pose estimation itself can be carried out quickly and only few iterations are done in the inner loop of the algorithm.

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 iterations. This is basically a multiresolution 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 the pose to a certain degree of accuracy. Then we increase the number of used Fourier coefficients and proceed to estimate the pose. The idea is to start with a rough approximation of the pose by using the low-pass informations and to refine it during the iterations. This is shown in Figure 9. In this example the iteration number corresponds directly to the number of used Fourier coefficients *plus 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 9 shows that the result is nearly perfect. Figure 10 shows pose results of an image sequence containing 540 images.

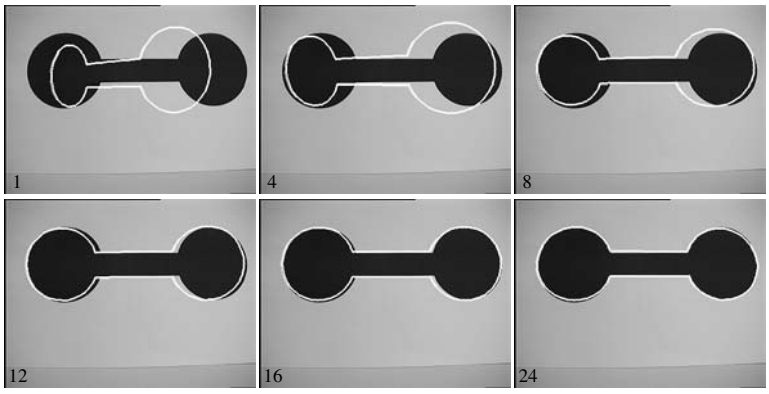

Fig. 8. Pose results during the iterations.

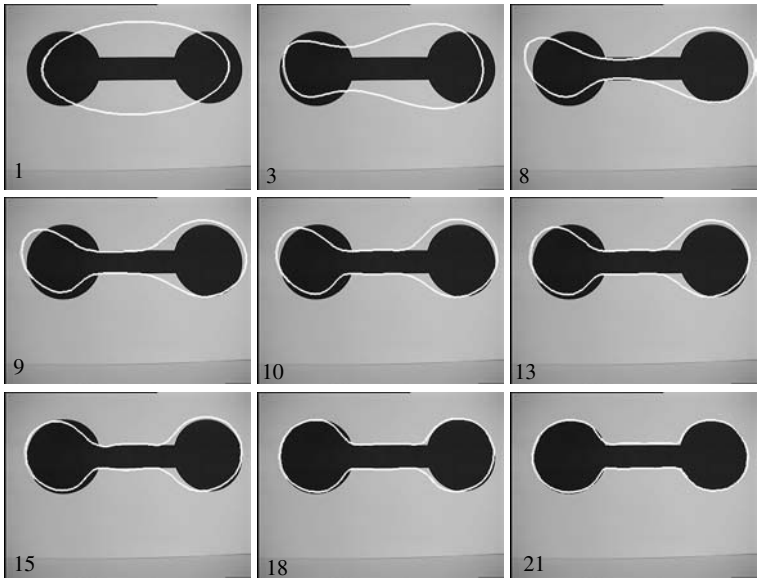

Fig. 9. Pose results of the low-pass contour during the iterations.

The accuracy and time performance of the algorithm is dependent on the number of object and image points spanning the contours in two and three dimensions, respectively. Furthermore, we can use the low-pass information of the 3D contour for approximations. We made several experiments with changing contour approximations and image points.

In the first experiment we compare the results of our algorithm for different degrees of contour approximation. On the one hand we use constant degrees over the iteration (3, 4, 5, 10, 20, 30, 40) and on the other hand increasing degrees, similar to Figure 9. To test the accuracy of our algorithmf we simply compare the translational error vector with ground truth. The result is shown in Figure 11. It can be seen that the algorithm converges after less than ten iterations. Then the error vectors do not change any more. It is clear that the use of less degrees leads to fast, but more inaccurate, results. Our *increasing degree* method finds a good optimum between computing time and accuracy of the result.

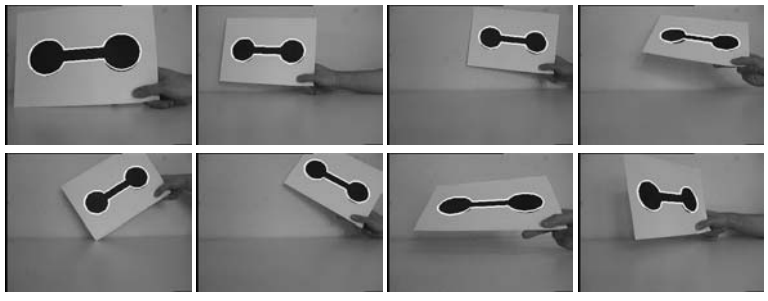

Fig. 10. Different pose results of the free-form contour.

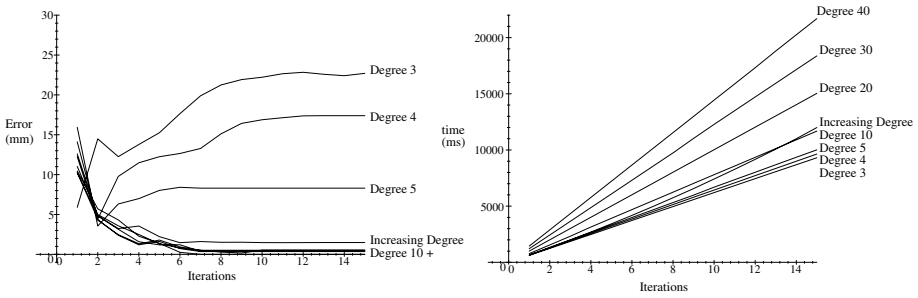

Fig. 11. Accuracy and computing time of the algorithm for a constant number of image points (80) and different approximation levels of the contour.

The algorithm converges after nine iterations, with computing times between 5 and 15 s on a SUN Ultra 10.

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 regular sampled image points and compared the accuracy and computing time. The result is presented in Figure 12. It can be seen that the number of image points used affects both the computing time and the accuracy. However, 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 contours with subsampled image points lead to faster and comparable results. In this scenario the computing time can be reduced from 35 s to less than 1 s without introducing nontolerable 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

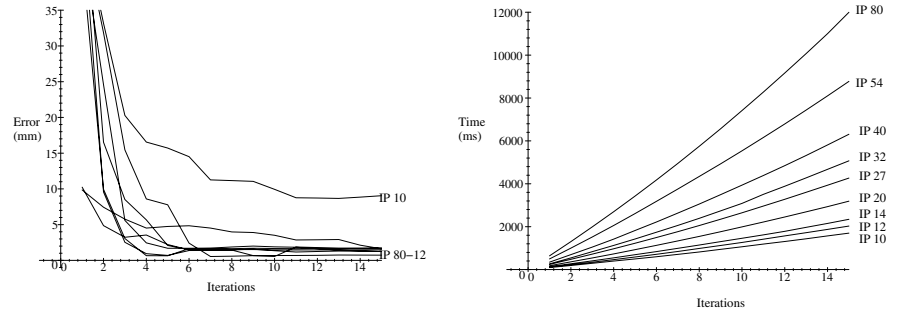

Fig. 12. Accuracy and computing time of the algorithm for changing image points.

Table 2. Computing time of the increasing degree method for the same scenario on different machines for 90 model points, 80 image points and 25 iterations.

Computer	Pose (ms)	Min. search	Total (25 iterations)
Pentium IV 2 GHz	5	up to 20 ms	405 ms
Pentium III 850 MHz	15	up to 25 ms	783 ms
Sparc Ultra 10	325	up to 80 ms	10,295 ms ($\sim$ 10 s)
Sparc Ultra 1	8,921	up to 667 ms	232,265 ms ($\sim$ 3.5 min)
Sparc 4	13,322	up to 1,505 ms	356,811 ms ($\sim$ 6 min)
Sparc 10	23,509	up to 1,743 ms	622,975 ms ($\sim$ 10 min)

in Table 2. As can be seen, the computing time for the increasing degree method, with 80 image points is 783 ms on a standard Linux 850 MHz machine. With slightly adapted parameters we reached 40 ms computing time with 1 mm pose deviation from ground truth. However, this value is dependent on the scenario and the object model. Many ideas to speed the algorithm up can also be found in [42]. We have not improved 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.

Figure 13 shows the computing times for an image sequence containing 520 images. The computing time for each image varies between 20 ms and 55 ms. The average computing time is 34 ms, which is equivalent to 29 fps. These results were achieved with a 2 GHz Pentium IV computer.

The robustness of our algorithm with respect to distorted image data is shown in Figure 14. In this image sequence (containing 450 images) we distort the image contour by covering parts of the contour with 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 the proposed method. It is not possible to compensate for totally wrong extracted contours or too much missing information.

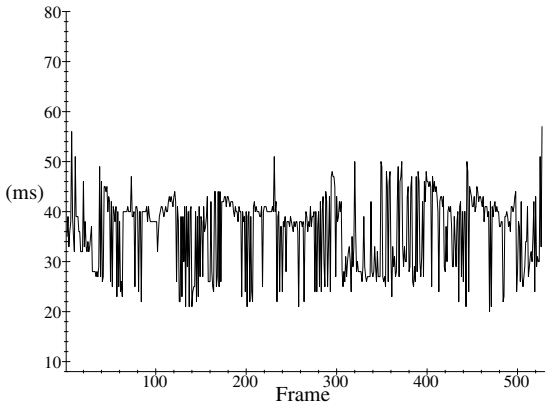

Fig. 13. Computing times for an image sequence containing 520 images.

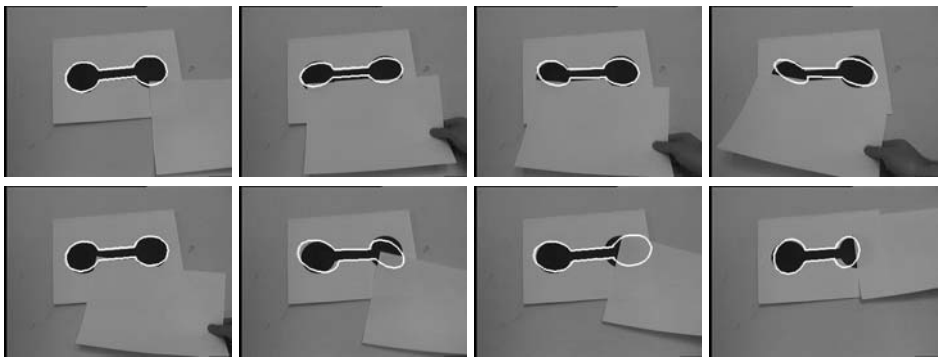

Fig. 14. Different pose results for distorted image data.

6. Discussion. This work presents an algorithm for free-form pose estimation. Therefore we start with simple point features and embed them in the family of cycloidal curves. Cycloidal curves give an interesting link between classical geometry and signal theory and are well suited to be used in the context of pose estimation. Therefore, we make use of conformal geometric algebra, which subsumes projective and kinematic geometry. One advantage is also the compact representation of the entities involved: just a set of twists, frequencies and a point on the contour are needed to express a twist generated curve, which is much easier to interpret than equivalent polynomials. The geometric derivation of the interpolation further allows the use of approximations of the contour. This can decrease the computing time immensely. So far Fourier descriptors are often used for object recognition but seldom for pose estimation. More popular are snakes or active contours. Since Fourier descriptors are nothing more than $m$twist generated curves, they can be defined in both the 2D and 3D space and low-pass information can be used within the pose problem.

References

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

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

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

[4] 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, 1990, pp. 640–647.

[5] Bayro-Corrochanno E., Daniilidis K. and Sommer G. Hand–eye calibration in terms of motions of lines using geometric algebra. In *Proceedings of the* 10*th Scandinavian Conference on Image Analysis*, Vol. 1, 1997, pp. 397–404.

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

[7] Bregler C. and Malik J. Tracking people with twists and exponential maps. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998, pp. 8–15.

[8] Chiuso A. and G. Picci. Visual tracking of points as estimation on the unit sphere. In *The Confluence of Vision and Control* (Kriegman D., Hager G., and Morse S., Eds). Springer-Verlag, New York, 1998, pp. 90–105.
[9] 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, 2001, pp. 166–210.
[10] O'Connor J.J. and Robertson E.F. Famous Curves Index. http://www-history.mcs.st-andrews.ac.uk/history/Curves/Curves.html.
[11] Czopf A., Brack C., Roth M. and Schweikard A. 3D–2D registration of curved objects. *Periodica Polytechnica*, Vol. 43, No. 1, 1999, pp. 19–41.
[12] Drummond T. and Cipolla R. Real-time tracking of multiple articulated structures in multiple views. In *Proceedings of the* 6*th European Conference on Computer Vision*, *ECCV* 2000, *Dublin*, *Ireland*, Part II, 2000, pp. 20–36.
[13] Faugeras O. Stratification of three-dimensional vision: projective, affine and metric representations. *Journal of the Optical Society of America*, Vol. 12, No. 3, 1995, pp. 465–484.
[14] Fenske A. Affin-invariante Erkennung von Grauwertmustern mit Fourierdeskriptoren. In *Mustererkennung* 1993 (Pöppel S.J. and Handels M., Eds.). Springer-Verlag, Berlin, 1993, pp. 75–83.
[15] Gallier J. *Geometric Methods and Applications for Computer Science and Engineering*. Springer-Verlag, New York, 2001.
[16] Granlund G. Fourier preprocessing for hand print character recognition. *IEEE Transactions on Computers*, Vol. 21, 1972, pp. 195–201.
[17] Grimson W.E.L. *Object Recognition by Computer*. The MIT Press, Cambridge, MA, 1990.
[18] Hestenes D. The design of linear algebra and geometry. *Acta Applicandae Mathematicae*, Vol. 23, 1991, pp. 65–93.
[19] Hestenes D. Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, No. 7, 1994, pp. 65–77.
[20] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [46], 2001, pp. 3–23.
[21] Hestenes D. and Ziegler R. Projective geometrie with Clifford algebra. *Acta Applicandae Mathematicae*, No. 23, 1991, pp. 25–63.
[22] 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, 1995, pp. 225–243.
[23] Huber D.F. and Hebert M. Fully automatic registration of multiple 3D data sets. *Proceedings of the IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum* (*CVBVS* 2001), 2001, pp. 637–650.
[24] 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, 1995, pp. 201–207.
[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. and Pauli J., Eds.). LNCS 1296. Springer-Verlag, Berlin, 1997, pp. 718–725.
[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) (Sandini G., Ed.). LNCS 588. Springer-Verlag, Berlin, 1992, pp. 829–833.
[27] Lee X. A Visual Dictionary of Special Plane Curves. http://xahlee.org/SpecialPlaneCurves_dir/specialPlaneCurves.html.
[28] Li H. Generalized homogeneous coordinates for computational geometry. In [46], 2001, pp. 27–52.
[29] Lin C.-S. and Hwang C.-L. New forms of shape invariants from elliptic Fourier descriptors. *Pattern Recognition*, Vol. 20, No. 5, 1987, pp. 535–545.
[30] Lowe D.G. Solving for the parameters of object models from image descriptions. In *Proceedings of the ARPA Image Understanding Workshop*, 1980, pp. 121–127.
[31] Lowe D.G. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, Vol. 31, No. 3, 1987, pp. 355–395.
[32] McCarthy J.M. *Introduction to Theoretical Kinematics*. MIT Press, Cambridge, MA, 1990.
[33] Murray R.M., Li Z. and Sastry S.S. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[34] Perwass C. Applications of Geometric Algebra in Computer Vision. Ph.D. thesis, Cambridge University, 2000. Available at http://www.perwass.de.
[35] Perwass C. and Lasenby L. A unified description of multiple view geometry. In [46], 2001, pp. 337–369.
[36] Reiss T.H. *Recognizing Planar Objects Using Invariant Image Features*. Springer–Verlag, New York, 1993.
[37] Rooney J. A comparison of representations of general spatial screw displacement. *Environment and Planning B*, Vol. 5, 1978, pp. 45–88.
[38] Rosenhahn B., Granert O. and Sommer G. Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering* (Dorst L., Doran C. and Lasenby J., Eds.). Birkhäuser-Verlag, Basel, 2001, pp. 373–383.
[39] Rosenhahn B. and Lasenby J. Constraint Equations for 2D–3D Pose Estimation in Conformal Geometric Algebra. Technical Report CUED/F - INFENG/TR.396, Engineering Department, Cambridge University, 2000.
[40] Rosenhahn B. and Sommer G. Adaptive pose estimation for different corresponding entities. In *Pattern Recognition*, 24*th DAGM Symposium* (Van Gool L., Ed.). LNCS 2449. Springer-Verlag, Berlin, 2002, pp. 265–273.
[41] Rosenhahn B., Zhang Y. and Sommer G. Pose estimation in the language of kinematics. *Proceedings of the Second International Workshop—Algebraic Frames for the Perception–Action Cycle*, *AFPAC* 2000. LNCS 1888. Springer-Verlag, Berlin, 2000, pp. 284–293.
[42] Rusinkiewicz S. and Levoy M. Efficient variants of the ICP algorithm. Available at http://www.cs.princeton.edu/ smr/papers/fasticp/. Presented at the Third International Conference on 3D Digital Imaging and Modeling (3DIM), 2001.
[43] Selig J.M. *Geometric Foundations of Robotics*. World Scientific, Singapore, 2000.
[44] Sommer G., editor. *Geometric Computing with Clifford Algebra*. Springer-Verlag, New York, 2001.
[45] 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.
[46] Tello R. Fourier descriptors for computer graphics. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, Vol. 25, No. 5, 1995, pp. 861–865.
[47] Ude A. Filtering in a unit quaternion space for model-based object tracking. *Robotics and Autonomous Systems*, Vol. 28, No. 2-3, August 1999, pp. 163–172.
[48] Walker M.W. and Shao L. Estimating 3-d location parameters using dual number quaternions. *CVGIP*: *Image Understanding*, Vol. 54, No. 3, 1991, pp. 358–367.
[49] Zahn C.T. and Roskies R.Z. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, Vol. 21, No. 3, 1972, pp. 269–281.
[50] Zang Z. Iterative point matching for registration of free-form curves and surfaces. *IJCV*: *International Journal of Computer Vision*, Vol. 13, No. 2, 1999, pp. 119–152.
[51] Zerroug, M. and Nevatia, R. Pose estimation of multi-part curved objects. *Proceedings of the Image Understanding Workshop* (*IUW*), 1996, pp. 831–835.