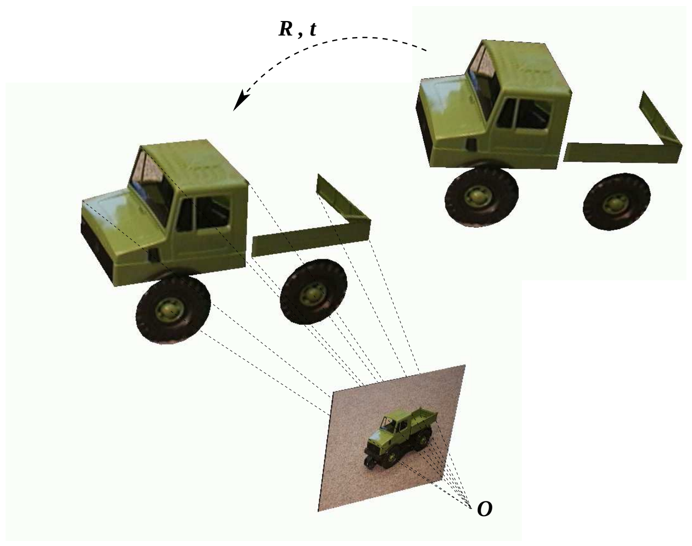
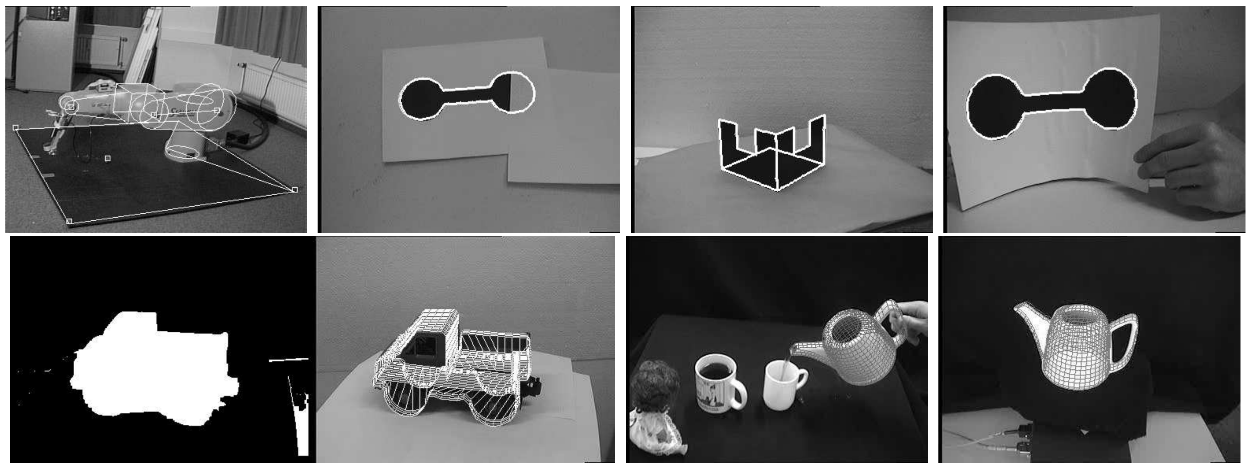
Foundations about 2D-3D Pose Estimation

Bodo Rosenhahn, Christian Perwass and Gerald Sommer





1 What is 2D-3D pose estimation?

Pose estimation has been studied in computer vision since its beginning. It is crucial for many computer and robot vision tasks. Object grasping, manipulation and recognition or self-localization of mobile robots are typical examples for the use of pose estimation. For a definition of the pose problem, we quote Grimson [17]:

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

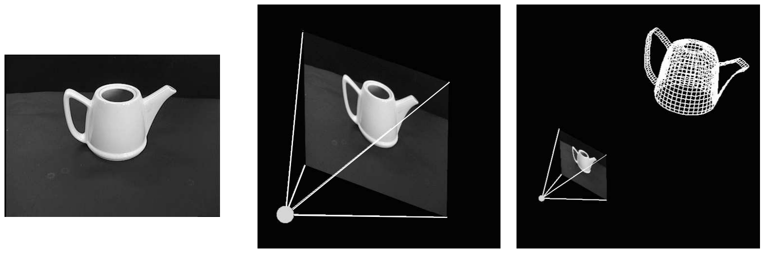


Figure 1: Our assumptions for the 2D-3D pose estimation problem: (left) Given is an image of an object. (middle) The camera is assumed to be calibrated with respect to a world coordinate system. (right) Also given is a 3D object model (in its own object coordinate system).

In this online contribution we present aspects of the so-called “2D-3D pose estimation problem”. With 2D-3D pose estimation we mean to fit 2D sensor data (an image of an object) with a 3D object model. That is, we are interested in estimating a rigid motion (containing both 3D rotation and 3D translation) which minimizes an error measure (which has to be defined) between the image and object data. This is visualized in figure 1: We first observe an object in an image (left). Furthermore the camera is calibrated, so that we can localize a camera coordinate system in the 3D world (visualized with the optical center and the shifted camera plane in the middle image). The object model is shown in the right image in its local coordinate system. Our aim is to compare the 3D object with the 2D image data to define a fit-value. We are interested in estimating a rigid motion which leads to a best fit between the object model and the image data as visualized in figure 2. Of course, there may exist different solutions. Though the problem sounds simple on a first glimpse, there are several important (and partially still open) questions which are discussed in the literature:

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

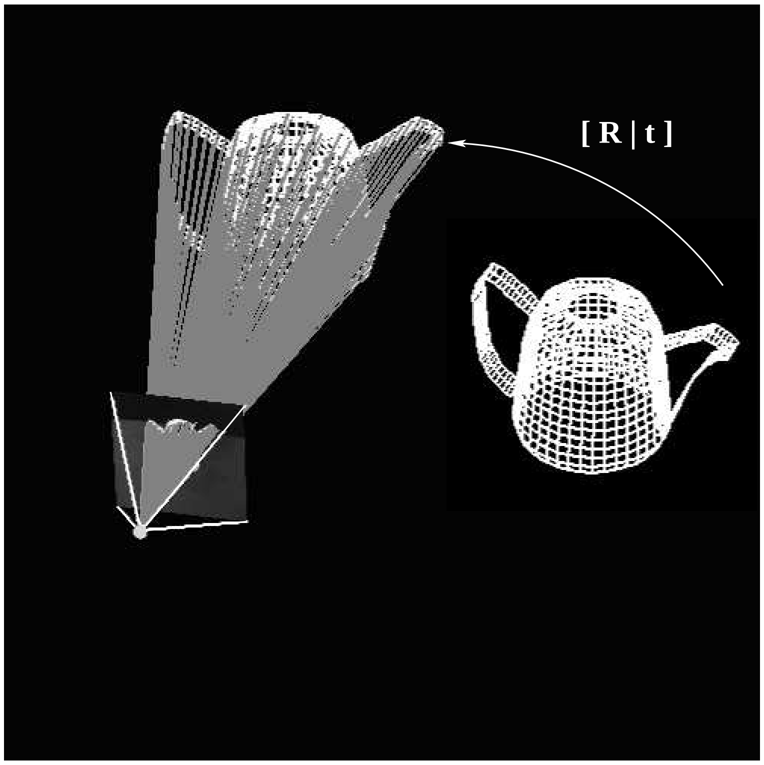


Figure 2: Visualization of the pose estimation problem: The task is to find the rigid motion which leads to a best fit between image data and the object model.

5. How will 3D object models be represented?

Indeed some of the questions can not be answered in a generalized manner. Instead they are dependent on the situation: For some scenes (e.g. industrial applications), a point based representation within an orthographic camera model and a simple corner extractor might be sufficient, but for more general scenes, this can be an inadequate representation. Therefore there is need to deal with the pose problem in a more general and flexible manner. In this work we do not give a long overview about the existing literature (e.g. [5, 12, 17, 25, 26, 29, 30, 42, 44]). Instead we refer to [15, 35] where different works are presented and discussed.

In this contribution we want to point out two main topics. Firstly we want to discuss the geometric scenario of 2D-3D pose estimation and we introduce the Conformal Geometric Algebra to deal with the involved mathematical spaces. The second topic deals with the representation of objects and we describe an object hierarchy which starts with simple 3D points and ends up with the representation of objects via general free-form contours and surface meshes. We further describe how to model kinematic chains and their coupling within a free-form surface mesh and present applications in human motion estimation. Both parts are mainly taken from the reports [35, 36] which are available on the internet. Due to the space limits we will only briefly scratch both topics and refer to the reports for more detailed information.

2 Geometric Aspects of 2D-3D pose estimation

This section deals with geometric foundations to deal with 2D-3D pose estimation. It further contains a brief introduction into main definitions and notations of geometric algebras (GAs). The reader is referred to [33, 34, 10, 11, 40, 20, 4] for more detailed information. We use the language of Clifford algebras, since it has several advantages for geometric applications, such as the pose problem: It provides a compact symbolic representation of entities and their transformations. Entities, such as points, lines or planes are represented as so-called multivectors and a geometric product can be used to model geometric transformations (such as rigid body motions, reflections or intersections) of the entities. Furthermore, certain Clifford algebras unify a lot of mathematical systems, such as the quaternions, dual-quaternions, Plücker geometry, Lie groups and Lie algebras or signal theory which are embedded in geometric algebras to model Euclidean, affine, projective and conformal geometry. These topics are often introduced separately and the interaction of different disciplines becomes even harder. Indeed, a geometric interpretation of e.g. the Fourier transform helps to exploit their properties for different applications, as shown later. The reader is referred to [22] for different homepages and projects in Clifford algebras.

2.1 The pose scenario

In the scenario of figure 2 four mathematical frameworks can be identified: The first one is a projective plane of a camera, embedded in the second framework, a 3D projective space. In these spaces it is possible to project or reconstruct entities by applying e.g. projection matrices. The third one is the framework of kinematics, which contains the map of the *direct affine isometries* [14] and can be used to describe rigid body motions. A set of entities with the property that the distances between any two of the entities never varies in time is called a *rigid body.* A continuous transformation with the property that it preserves distances during transformation is called a *rigid body motion.* A rigid body motion corresponds to the Euclidean transformation group $SE(3)$[14]. Being a transformation by itself, it contains rotations and translations. The fourth framework is the Euclidean space or Euclidean plane, which is important to define distances between entities. The basic definitions of these spaces are the following [14]:

1. The **Euclidean space** is a vector space V with a symmetric positive definite bilinear form (which induces an Euclidean norm).

2. The **affine space** consists of a set of points, a derived vector space and two operations, namely vector addition and scalar multiplication.

3. The **kinematic space** is an affine space with the group of rigid motions as special affine transformation.

4. The **projective space** $\mathcal{P}(V)$ induced by V is the set $(V\backslash\{0\})/\!\sim$ of equivalence classes of nonzero vectors in V under the equivalence relation $\sim$ defined such that

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

Table 1: Stratification of mathematical spaces.

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

Mathematically a projective space $\mathcal{P}(V)$ is a set of equivalence classes of vectors in V. The idea behind projective geometry is to view an equivalence class $(u)_\sim$ as an atomic object, forgetting the internal structure of the equivalence class. For this reason it is customary to call an equivalence class $a = (u)_\sim$ a *point* (the entire equivalence class $(u)_\sim$ is collapsed into a single object, viewed as a point).

The idea is to reach the Euclidean plane or Euclidean space in the end. Only in this way it is possible to cope with the problem of noisy data in a geometric context and to evaluate the quality of the estimated pose. But since the Euclidean space is not well suited to describe projective geometry and kinematics, the aim is to transform (later introduced) constraint equations into a distance measure of the Euclidean space in the very last step. Before this step, the other spaces are used to represent the situation in a suitable way. The spaces of the pose estimation scenario mentioned above are exactly the spaces of the stratification hierarchy which Faugeras introduced in 1995 [13]. The three main representations he considered are the projective, affine and metric ones. All strata are involved in the 2D-3D pose estimation problem.

The stratification hierarchy proposed by Faugeras has its roots in the vector space concepts and assumes points as the represented basic geometric entities. The other spaces are derived from the vector space concepts: Well known is the homogeneous extension to express an Euclidean space as affine space and to use the homogeneous component for distinction between points and directions in the affine space. The projective space as a set of equivalence classes is directly build on the homogeneous vector space concepts. This way to stratify the vision space is clearly motivated by the underlying point concepts of the vector spaces and is shown in the first row of table 1.

We are using geometric algebras to represent and handle different mathematical spaces of geometric meaning. The maximum sized algebra used so far, is an algebra to handle conformal transformations [20]. A transformation is said to be conformal, if it (locally) preserves angles. This algebra contains algebras for modeling projective and Euclidean geometry as sub-algebras. It results in other layers of stratification which we propose for pose estimation as shown in table 1.

2.2 Geometric algebras

A *geometric algebra* [20] is a Clifford algebra (William K. Clifford lived 1845-1879) with a main focus on geometric interpretations. Clifford's contribution to a geometric extension of the real number system to a complete algebraic representation of directed numbers is historically reviewed by M. Yaglom in [43], which also discusses the relation to former works of Grassmann (1809-1877) or Hamilton (1805-1865). The term geometric algebra was introduced by David Hestenes [22](3) in the 1960's, who further developed Clifford algebra in classical geometry and mechanics.

In general, a *geometric algebra* $\boldsymbol{\mathcal{G}}_{p,q,r}$ $(p,q,r,\in \mathbb{N}_0)$ is a linear space of dimension 2^n, $n = p+q+r$, with subspaces, called *blades*. Elements of a geometric algebra are *multivectors*, which are higher-grade algebraic entities in comparison to vectors of a vector space being first-grade entities. A geometric algebra $\boldsymbol{\mathcal{G}}_{p,q,r}$ is constructed from a vector space $\mathbb{R}^{p,q,r}$, endowed with the signature (p,q,r), by applying a *geometric product.* The geometric product defining a geometric algebra is denoted by juxtaposition (e.g., $\boldsymbol{AB}$ for two multivectors $\boldsymbol{A}$ and $\boldsymbol{B}$).

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

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

The geometric product of the same two basis vectors leads to a scalar, whereas the geometric product of two different basis vectors leads to a new entity, which is called a *bivector.* This bivector represents the subspace, spanned by these two vectors. Geometric algebras can be expressed on the basis of *graded elements.* Scalars are of grade zero, vectors of grade one, bivectors of grade two, etc. A linear combination of elements of different grades is called a multivector $\boldsymbol{M}$ and can be expressed as

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

where the operator $\langle.\rangle_i$ denotes the projection of a general multivector to the entities of grade i. A multivector of grade i is called an *i-blade* if it can be written as the outer product of i vectors. A blade $\boldsymbol{A}$ of grade i can be written as $\boldsymbol{A}_{\langle i\rangle}$.

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

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

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

The inner product of a r-blade $\boldsymbol{u}_1\wedge\ldots\wedge\boldsymbol{u}_r$ with a s-blade $\boldsymbol{v}_1\wedge\ldots\wedge\boldsymbol{v}_s$ is defined recursively as

$$(\boldsymbol{u}_1\wedge\ldots\wedge\boldsymbol{u}_r)\cdot(\boldsymbol{v}_1\wedge\ldots\wedge\boldsymbol{v}_s) = \begin{cases} ((\boldsymbol{u}_1\wedge\ldots\wedge\boldsymbol{u}_r)\cdot\boldsymbol{v}_1)\cdot(\boldsymbol{v}_2\wedge\ldots\wedge\boldsymbol{v}_s) & \text{if } r\geq s \\ (\boldsymbol{u}_1\wedge\ldots\wedge\boldsymbol{u}_{r-1})\cdot(\boldsymbol{u}_r\cdot(\boldsymbol{v}_1\wedge\ldots\wedge\boldsymbol{v}_s)) & \text{if } r<s, \end{cases} \tag{2.4}$$

with

$$(\boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_r) \cdot \boldsymbol{v}_1 = \sum_{i=1}^{r} (-1)^{r-i} \boldsymbol{u}_1 \wedge \ldots \wedge \boldsymbol{u}_{i-1} \wedge (\boldsymbol{u}_i \cdot \boldsymbol{v}_1) \wedge \boldsymbol{u}_{i+1} \wedge \ldots \wedge \boldsymbol{u}_r, \quad (2.5)$$

$$\boldsymbol{u}_r \cdot (\boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_s) = \sum_{i=1}^{s} (-1)^{i-1} \boldsymbol{v}_1 \wedge \ldots \wedge \boldsymbol{v}_{i-1} \wedge (\boldsymbol{u}_r \cdot \boldsymbol{v}_i) \wedge \boldsymbol{v}_{i+1} \wedge \ldots \wedge \boldsymbol{v}_s. \quad (2.6)$$

The following example illustrates the definition of the inner product. Let $\mathbf{e}_1, \mathbf{e}_2 \in \mathcal{G}_{2,0}$ then

$$\mathbf{e}_1 \cdot (\mathbf{e}_1 \wedge \mathbf{e}_2) = (\mathbf{e}_1 \cdot \mathbf{e}_1) \wedge \mathbf{e}_2 - \mathbf{e}_1 \wedge (\mathbf{e}_1 \cdot \mathbf{e}_2) = \mathbf{e}_2.$$

The blades of highest grade are n-blades, called *pseudoscalars.* For non-degenerate geometric algebras, there exist two unit n-blades, called the *unit pseudoscalars* $\pm \boldsymbol{I}$.

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

$$\boldsymbol{X}^\star := \boldsymbol{X}\boldsymbol{I}^{-1}. \quad (2.7)$$

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

$$\widetilde{\boldsymbol{A}_{\langle s \rangle}} = \boldsymbol{a}_s \wedge \boldsymbol{a}_{s-1} \wedge \ldots \wedge \boldsymbol{a}_2 \wedge \boldsymbol{a}_1. \quad (2.8)$$

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

$$\boldsymbol{AB} = \frac{1}{2}(\boldsymbol{AB} + \boldsymbol{BA}) + \frac{1}{2}(\boldsymbol{AB} - \boldsymbol{BA}) =: \boldsymbol{A} \overline{\times} \boldsymbol{B} + \boldsymbol{A} \underline{\times} \boldsymbol{B}. \quad (2.9)$$

We cite [33] for the use of the commutator and anticommutator product. Basically, their role is to separate the symmetric part of the geometric product from the antisymmetric part.

We use the conformal geometric algebra for dealing with the pose estimation problem. Therefore, we now give a brief introduction into this particular geometric algebra.

2.3 Conformal geometric algebra

This algebra has been designed for dealing with stereographic projections. Historically, stereographic projections have been proposed for flat maps of the earth. A stereographic projection is visualized in Figure 3 for the 1D case: Here, the 'earth' is a transparent circle, intersected on the equator by an *equatorial line.* Now imagine a light bulb at the *north pole* $\boldsymbol{n}$, which shines through the circle. Each point on the circle casts a shadow on the equatorial line and that is where it is drawn on the map. Ray theorems lead to the following formulas (for the 1D case) for projecting a point on a unit circle and vice versa: A point $\boldsymbol{x}'$ on the circle is given by its angle α:

$$\boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ = \cos(\alpha)\mathbf{e}_1 + \sin(\alpha)\mathbf{e}_+. \quad (2.10)$$

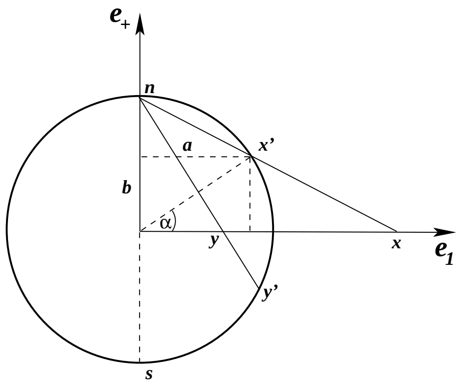


Figure 3: Visualization of a stereographic projection for the 1D case: Points on the circle are projected onto the line or vice versa.

The point on the line then has the coordinates

$$\boldsymbol{x} \;=\; \left(\frac{\cos(\alpha)}{1-\sin(\alpha)}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{2.11}$$

To project a point $x\mathbf{e}_1$ ($x \in \mathbb{R}$) onto the circle, the following equations are valid [32],

$$\boldsymbol{x}' \;=\; a\mathbf{e}_1 + b\mathbf{e}_+ = \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+. \tag{2.12}$$

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

$$\boldsymbol{x}' \;=\; x\mathbf{e}_1 + \frac{1}{2}\left(x^2-1\right)\mathbf{e}_+ + \frac{1}{2}\left(x^2+1\right)\mathbf{e}. \tag{2.13}$$

Note that this is in fact a double-homogeneous representation with $\mathbf{e}$ and $\mathbf{e}_+$ as two homogeneous basis vectors. In [27], $\mathbf{e}$ is defined to have a negative signature, and therefore $\mathbf{e}$ is replaced with $\mathbf{e}_-$, where $\mathbf{e}_-^2 = -1$. Euclidean points, stereographically projected onto the unit circle in Figure 4, are unit points and then represented by the set of null vectors in the new space. A Euclidean point is mapped into the conformal space by

$$\boldsymbol{x} \Rightarrow \boldsymbol{x}' = a\mathbf{e}_1 + b\mathbf{e}_+ + \mathbf{e}_-, \quad \text{with} \quad (\boldsymbol{x}')^2 = a^2 + b^2 - 1 = 0. \tag{2.14}$$

The coordinates (a, b) are the coordinates of a point on the unit circle. Note that each point in Euclidean space is represented by a line of null vectors in the new space, which are the scaled versions of the null vector on the unit circle. This homogeneous representation of a point is used as *point* in conformal geometric algebra (see Figure 4).

Our brief introduction into CGA follows [27]. We start with a *Minkowski plane*, $\mathcal{G}_{1,1}$. Its vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the properties

$$\mathbf{e}_+^2 = 1, \quad \mathbf{e}_-^2 = -1, \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0. \tag{2.15}$$

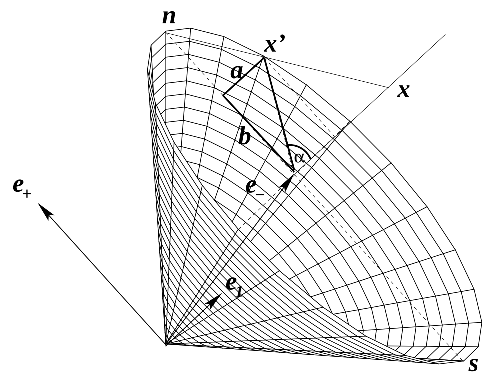


Figure 4: Visualization of the homogeneous model for stereographic projections for the 1D case. All stereographically projected points lie on a cone, which is a null-cone in the Minkowski space. Note that in comparison to Figure 3, the coordinate axes are rotated and drawn perspectively.

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

$$\mathrm{e}_0 := \frac{1}{2}(\mathrm{e}_- - \mathrm{e}_+) \quad \text{and} \quad \mathrm{e} := \mathrm{e}_- + \mathrm{e}_+. \tag{2.16}$$

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

$$\boldsymbol{I}_C \;=\; \mathrm{e}_{+-123} = \boldsymbol{E}\mathrm{e}_{123} = \boldsymbol{E}\boldsymbol{I}_E. \tag{2.17}$$

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

$$\underline{\boldsymbol{x}} \;=\; \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathrm{e} + \mathrm{e}_0. \tag{2.18}$$

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

$$\begin{aligned}\underline{\boldsymbol{x}} &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathrm{e} + \mathrm{e}_0 \\ &= \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2(\mathrm{e}_+ + \mathrm{e}_-) + \frac{1}{2}(\mathrm{e}_- - \mathrm{e}_+) \\ &= \boldsymbol{x} + \left(\frac{1}{2}\boldsymbol{x}^2 - \frac{1}{2}\right)\mathrm{e}_+ + \left(\frac{1}{2}\boldsymbol{x}^2 + \frac{1}{2}\right)\mathrm{e}_-. \end{aligned} \tag{2.19}$$

This is exactly the homogeneous representation of a stereographically projected point onto the circle, given in equation (2.13) for the 1D case. A point representation is

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

Table 2: [27] Entities and their dual representations in CGA.

shorter written if using $\{\mathrm{e}, \mathrm{e}_0\}$, instead of $\{\mathrm{e}_+, \mathrm{e}_-\}$. Algebras $\mathcal{G}_{3,1}$ and $\mathcal{G}_{3,0}$ can be used to represent the projective and Euclidean space; see [18, 21].

We have

$$\mathcal{G}_{4,1} \quad \supseteq \quad \mathcal{G}_{3,1} \supseteq \mathcal{G}_{3,0}. \tag{2.20}$$

It follows that both algebras (i.e., for the projective and Euclidean space) constitute subspaces of the CGA. It is possible to use operators to relate the different algebras and to guarantee the mapping between the algebraic properties [35, 34]. This relation builds a stratification hierarchy, containing the Euclidean, projective and conformal space, which is different to Faugeras' stratification hierarchy [13], containing the Euclidean, affine and projective space.

Since we now have the basic representation of a point in CGA, we are able to define other entities based on this point representation. The entities and their dual representation are summarized in Table 2. As it can be seen, it is possible to express points, lines, planes, circles, spheres and point pairs in this algebra. Lines are given in Plücker form (direction $\boldsymbol{r}$ and moment $\boldsymbol{m}$), and planes are given in Hessian form (normal $\boldsymbol{n}$ and distance d).

There exist different representations of points in CGA. A point can be given as $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathrm{e} + \mathrm{e}_0$. This can be seen as a sphere with radius zero. A point is also given as $\underline{\boldsymbol{X}}^\star = \mathrm{e} \wedge \underline{\boldsymbol{x}} = \boldsymbol{E} + \mathrm{e}\boldsymbol{x}$, which is a point pair involving one point at infinity, and this is called the *affine representation* of a point [27]. For pose estimation we work with dual representation of points, lines and planes as $\underline{\boldsymbol{X}}^\star$, $\underline{\boldsymbol{L}}^\star$ and $\underline{\boldsymbol{P}}^\star$.

2.3.1 Conformal transformations

In CGA, any conformal transformation can be expressed as versor product of the form

$$\sigma\underline{\boldsymbol{x}}' \quad = \quad G\underline{\boldsymbol{x}}G^{-1}, \tag{2.21}$$

where G is called a versor and σ a scalar. Table 3 summarizes the conformal transformations. The first column shows the type of operation performed with the versor product. The second column shows as example the result of a transformation acting on a point. The third column shows the versor which has to be applied and the last column shows the scaling parameter σ which is (sometimes) needed to result in a homogeneous point and to ensure the scaling $\underline{\boldsymbol{x}}' \cdot \mathrm{e} = \underline{\boldsymbol{x}} \cdot \mathrm{e} = -1$.

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

Table 3: [27] Table of conformal transformations, versors and scaling parameters.

To express the pose estimation problem, we need a versor to express the rigid body motion. As mentioned previously, a rigid body motion corresponds to the Euclidean transformation group $SE(3)$. Although being a transformation by itself, it subsumes rotation and translation. The rotation of an entity can be performed just by multiplying the entity from the left with a rotor $\boldsymbol{R}\in\langle\boldsymbol{\mathcal{G}}_{4,1}\rangle_2$, and from the right with its reverse $\widetilde{\boldsymbol{R}}$. For example, a rotation of a point can be written as

$$\underline{\boldsymbol{x}}' \;=\; \boldsymbol{R}\underline{\boldsymbol{x}}\widetilde{\boldsymbol{R}}. \tag{2.22}$$

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

$$\boldsymbol{R} \;=\; \exp\left(-\frac{\theta}{2}\boldsymbol{n}\right)=\cos\left(\frac{\theta}{2}\right)-\boldsymbol{n}\sin\left(\frac{\theta}{2}\right). \tag{2.23}$$

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

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

$$\boldsymbol{T} \;=\; \left(1+\frac{\mathbf{e}\boldsymbol{t}}{2}\right)=\exp\left(\frac{\mathbf{e}\boldsymbol{t}}{2}\right). \tag{2.24}$$

Similar to a rotation, an entity can be translated by multiplying the entity from the left with the translator $\boldsymbol{T}$ and its reverse $\widetilde{\boldsymbol{T}}$ from the right. To express a rigid body motion, the consecutive application of a rotor and translator can be written as their product. Such an operator is a special even grade multivector, called a *motor*.

A motor in CGA can also be defined by so-called *twists*. The idea is to interpret a motor as an element of the Lie group of rigid body motion and to construct it by exponentiation of its generator as a Lie algebra element. In fact, the mentioned twist is the generator of the motor. It is well known, that every rigid body motion can be expressed as a twist or screw motion [31], which is a rotation around a line in space[1]

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

combined with a translation along this line. In CGA, it is possible to use the rotors and translators to express screw motions in space. A screw motion is defined by an axis $\boldsymbol{l}^\star$, a pitch h and a magnitude θ. The *pitch* of the screw is the ratio of translation to rotation, $h := \frac{d}{\theta}$ ($d, \theta \in \mathbb{R}$, $\theta \neq 0$). If $h \to \infty$, then the corresponding screw motion consists of a pure translation along the axis of the screw. The resulting motor takes the form

$$\boldsymbol{M} \;=\; \exp\left(-\frac{\theta}{2}\left(\boldsymbol{l} + \mathbf{e}\boldsymbol{m}\right)\right). \tag{2.25}$$

The bivector $-\frac{\theta}{2}(\boldsymbol{l} + \mathbf{e}\boldsymbol{m})$ in the exponent is a twist. The vector $\boldsymbol{m}$ is a vector in $\mathbb{R}^3$, which can be decomposed into an orthogonal and parallel part with respect to the rotation axis $\boldsymbol{n} = \boldsymbol{l}^\star$.

Note, that the conformal geometric algebra allows us to couple projective geometry with kinematics (due to its double-homogeneous representation) and it allows to represent higher order entities and their transformations. It is therefore well suited for modeling the pose problem!

3 Object Representations for Pose Estimation

In this part we introduce pose estimation algorithms which rely on different kinds of object representations. We start with a simple corner or line based one and continue up to general free-from surface models. Indeed there are still some representations and special cases missing, which have not been exploited, yet. Such aspects are part of future research and there are many thinks left to do ...

3.1 Point and line based pose estimation

Recalling Figure 2, we have the following assumptions: We assume to have extracted 2D image points (e.g., corners in an image of a calibrated camera) and to know their corresponding 3D points (e.g., corners of the assumed 3D object model). To compare these features, we reconstruct projection rays from the image points and claim that the transformed 3D points must be incident with the reconstructed rays. This can be expressed in CGA in the following way: Let $\underline{\boldsymbol{X}}^\star \in \mathcal{G}_{4,1}$ be an object point. The (unknown) transformed point can be written as

$$\underline{\boldsymbol{X}}'^\star \;=\; \boldsymbol{M}\underline{\boldsymbol{X}}^\star\widetilde{\boldsymbol{M}}. \tag{3.26}$$

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

$$\underline{\boldsymbol{L}}_x^\star \;=\; \mathrm{e} \wedge \boldsymbol{O} \wedge \boldsymbol{x} \quad \in \mathcal{G}_{4,1}. \tag{3.27}$$

The vector $\boldsymbol{O} \in \mathcal{G}_{3,1}$ denotes the optical center of the camera. This leads to a Plücker representation of a line given as direction and moment. Using the commutator product $\underline{\times}$ we compare the transformed object point with the 3D projection ray. [35] shows that this is a well suited constraint equation since it expresses a 3D distance measure between the line and the point in 3D, as shown in Figure 5. The constraint equation for pose estimation from 2D-3D point correspondences can now be expressed in the following way:

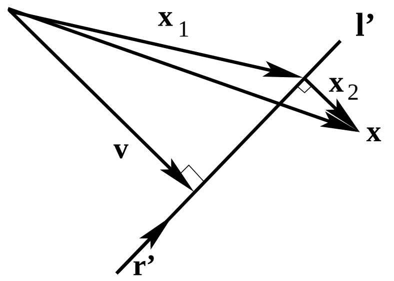


Figure 5: The comparison of the 3D line with the 3D point leads to the 3D perpendicular error vector.

$$\underbrace{\lambda(\underbrace{\underbrace{(\boldsymbol{M}\ \underbrace{\underline{\boldsymbol{X}}^{\star}}_{\text{object point}}\ \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}} \ \underline{\times}\ \underbrace{\mathbf{e}\wedge(\underbrace{\boldsymbol{O}}_{\text{optical center}} \wedge \underbrace{\boldsymbol{x}}_{\text{image point}})}_{\text{projection ray, reconstructed from the image point}}}_{\text{collinearity of the transformed object point with the reconstructed line}})\cdot\mathbf{e}_{+}}_{\text{distance measure between 3D point and 3D line}} = 0. \quad (3.28)$$

In section 2.1 we pointed out the mathematical spaces involved in pose estimation. All these spaces can be tracked back in the above equation: The mathematical spaces involved here are

$$\underbrace{\lambda\,(\underbrace{\underbrace{(\boldsymbol{M}\underbrace{\underline{\boldsymbol{X}}^{\star}}_{CS}\widetilde{\boldsymbol{M}})}_{CS}\ \underline{\times}\ \underbrace{\mathbf{e}\wedge\underbrace{(\underbrace{\boldsymbol{O}}_{PS}\wedge\underbrace{\boldsymbol{x}}_{PP})}_{PS}}_{CS}}_{CS})\cdot\mathbf{e}_{+}}_{ES} = 0. \quad (3.29)$$

Here PP abbreviates *projective plane*, PS *projective space*, CS *conformal space* and ES *Euclidean space.*

3.1.1 Numerical estimation of pose parameters

The unknown rigid motion is given as an exponential function. Therefore, it can not directly be solved in a straightforward way. We propose to linearize the equations (i.e., we use the tangential space, the Lie algebra) and iterate the solutions. This results in a gradient-descent method. The Euclidean transformation of a point $\underline{\boldsymbol{X}}^{\star}$ caused by the motor $\boldsymbol{M}$ is approximated in the following way:

$$\boldsymbol{M}\underline{\boldsymbol{X}}^{\star}\widetilde{\boldsymbol{M}} = \exp\left(-\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right)\underline{\boldsymbol{X}}^{\star}\exp\left(\frac{\theta}{2}(\boldsymbol{l'}+\mathbf{e}\boldsymbol{m'})\right)$$

$$
\begin{aligned}
&\approx (1 - \frac{\theta}{2}(\boldsymbol{l}' + \mathrm{e}\boldsymbol{m}'))\underline{\boldsymbol{X}}^\star(1 + \frac{\theta}{2}(\boldsymbol{l}' + \mathrm{e}\boldsymbol{m}')) \\
&\approx \boldsymbol{E} + \mathrm{e}(\boldsymbol{x} - \theta(\boldsymbol{l}' \cdot \boldsymbol{x}) - \theta\boldsymbol{m}'). \qquad (3.30)
\end{aligned}
$$

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

$$
\boldsymbol{M}\underline{\boldsymbol{X}}^\star\widetilde{\boldsymbol{M}} \approx \boldsymbol{E} + \mathrm{e}(\boldsymbol{x} - \boldsymbol{l}\cdot\boldsymbol{x} - \boldsymbol{m}). \qquad (3.31)
$$

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

$$
\begin{aligned}
0 &= \boldsymbol{M}\underline{\boldsymbol{X}}^\star\widetilde{\boldsymbol{M}} \underline{\times} \underline{\boldsymbol{L}}^\star \\
\Leftrightarrow 0 &= \exp\left(-\frac{\theta}{2}(\boldsymbol{l}' + \mathrm{e}\boldsymbol{m}')\right)\underline{\boldsymbol{X}}^\star \exp\left(\frac{\theta}{2}(\boldsymbol{l}' + \mathrm{e}\boldsymbol{m}')\right) \underline{\times} \underline{\boldsymbol{L}}^\star \\
\Leftarrow\approx\Rightarrow 0 &= (\boldsymbol{E} + \mathrm{e}(\boldsymbol{x} - \boldsymbol{l}\cdot\boldsymbol{x} - \boldsymbol{m}))\underline{\times} \underline{\boldsymbol{L}}^\star. \qquad (3.32)
\end{aligned}
$$

Because of the approximation ($\Leftarrow\approx\Rightarrow$), the unknown motion parameters $\boldsymbol{l}$ and $\boldsymbol{m}$ are linear. This equation contains six unknown parameters of the rigid body motion, which are the twist parameters. The linear equations can be solved for a set of correspondences by applying, for example, the Householder method. From the solution, the pose can be recovered by applying the Rodriguez formula. Iteration leads to a gradient descent approach in a 3D space, which is fast and stable.

As shown in [35] it is possible to express in a very similar manner constraint equations which relate 3D points to 3D planes (reconstructed 2D lines) or 3D lines to 3D planes (reconstructed from 2D image lines). It is also possible to fuse the linearized constraint equations in one system of equations and to solve the equations simultaneously. Furthermore, it is possible to weight the equations and therefore to influence the equations with a confidence measure, which might arise from the image processing. This is shown in figure 6, where point and line features are used separately and weighted-combined for pose estimation.

3.2 Kinematic chains

The approach for point based pose estimation can be extended to kinematic chains by introducing further screw transformations which model revolute or prismatic joints by using special twists [5]. The transformation of a point $\underline{\boldsymbol{X}}^\star{}_{j,i_j}$ on the j-th joint to the base coordinate system is represented by a sequence of motors $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_j$. The constraint equations take the form

$$
(\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_j\underline{\boldsymbol{X}}^\star{}_{j,i_j}\widetilde{\boldsymbol{M}}_j \ldots \widetilde{\boldsymbol{M}}_1)\widetilde{\boldsymbol{M}}) \underline{\times}\, \mathrm{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{j,i_j}) = 0.
$$

The constraint equations can be linearized and iterated in the same manner as for pure point concepts. Figure 7 shows example images for pose estimation of a cupboard and estimation of the opening angle of its door. The pose results is visualized by projecting the transformed object model into the image. Figure 8 shows example images of the

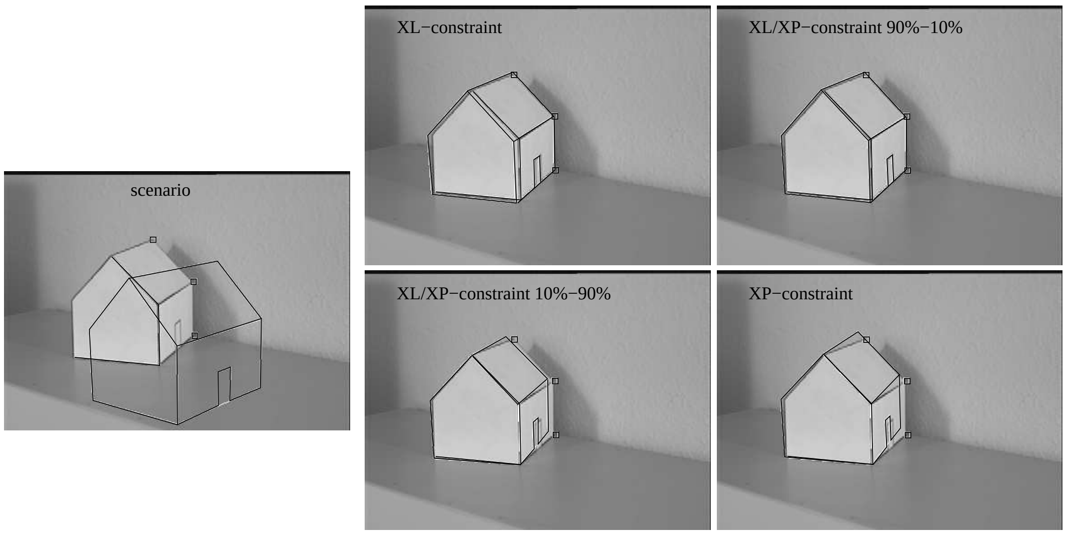


Figure 6: Different weights of constraints for pose estimation. The upper left image shows the pose result for using point-line correspondences. The lower right image shows the pose result for using point-plane correspondences. The other images show pose results obtained with simultaneously used and weighted point-line and point-plane correspondences.

implementation of a *visual remote control* for a manipulator: A person with color markers attached to its body is tracked and the angles of the human arm's movements are estimated and translated to the robot kinematics. So the robot arm follows the human arm movements and therefore the human is able to control the robot through its own movements. Color markers on the finger tips indicate the opening and closing of the gripper.

3.3 Constraint curves

The question we are now concerned with is the representation of higher order curves such as circles, spheres or ellipses.

We will consider curves as generalized geometric entities which result from coupled twist transformations of a point. That is, here we restrict ourselves to model curves by the algebraic constrained motion of points in space. The idea for modeling ellipses is visualized in figure 9, left: We assume two parallel axes in 3D space and a 3D point on the ellipse, and we transform the point around the two axes in a fixed and dependent manner. In this case we use two coupled parallel (not collinear) axes, rotate the point by -2ϕ around the first axis and by ϕ around the second one. The set of all points for $\phi \in [0, \ldots, 2\pi]$ generates an ellipse. This can be seen as the constraint curve of an end-effector transformed by a 2 d.o.f. virtual kinematic chain. Since we use twists to model joints on a manipulator we call these curves twist-generated curves. It is clear that different frequencies and location of the twist axes lead to a family of curves and indeed a subclass of such twist-generated curves are the famous *cycloidal curves* which are here

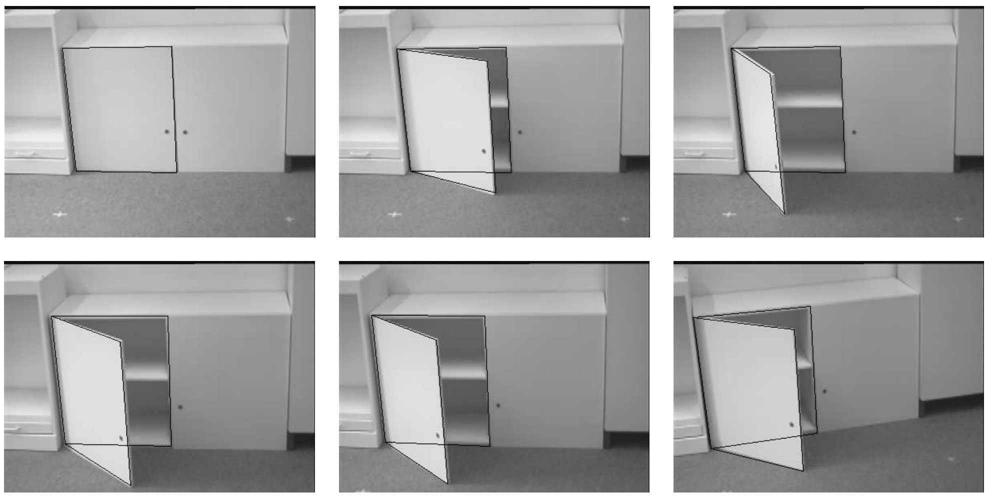


Figure 7: Example images for pose estimation of a cupboard and estimation of the doors' opening angle.

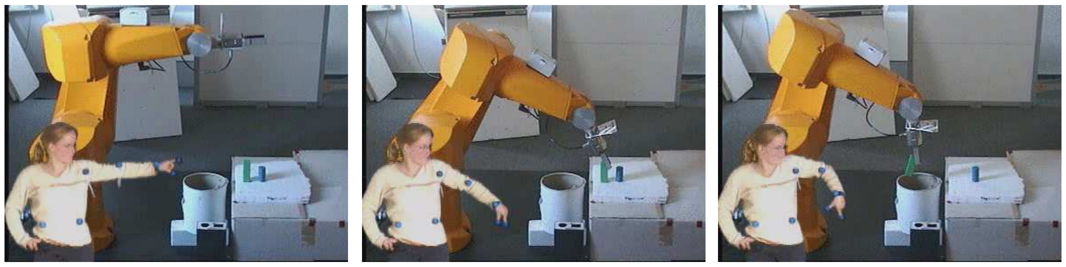


Figure 8: Example images for visual remote controlling of the robot.

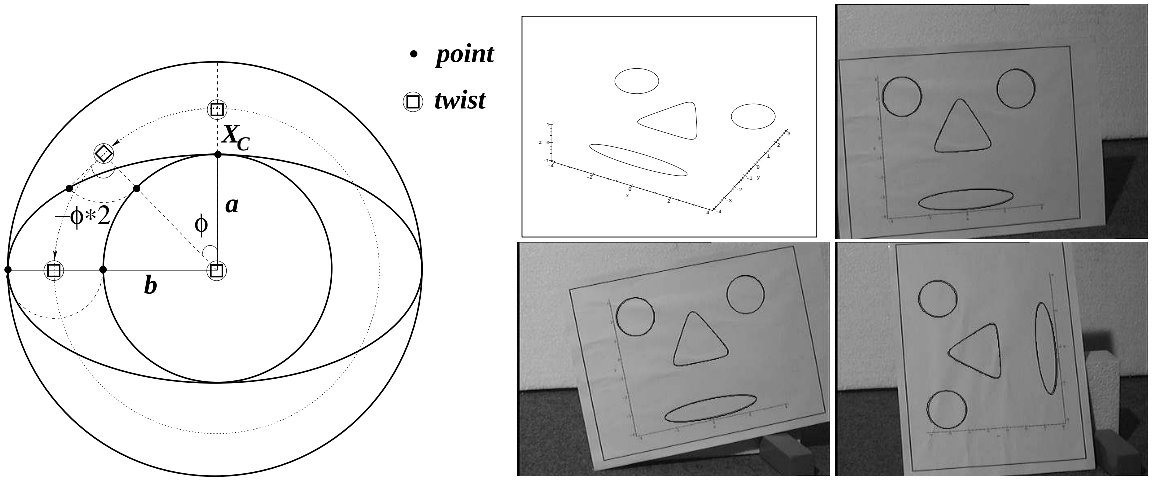


Figure 9: Left: An ellipse generated by two coupled twists. Right: Pose estimation of an object containing one ellipse, two circles and one deltoid.

embedded in 3D space. Cycloidal curves are defined as curves generated from circles rolling on circles or lines. Examples are cardioids, nephroids, ellipses, roses, astroids, etc.

Figure 9 right, shows example pose results of an object model which contains an ellipse, two circles and one deltoid. An interesting aspect is, that the rigid transformation

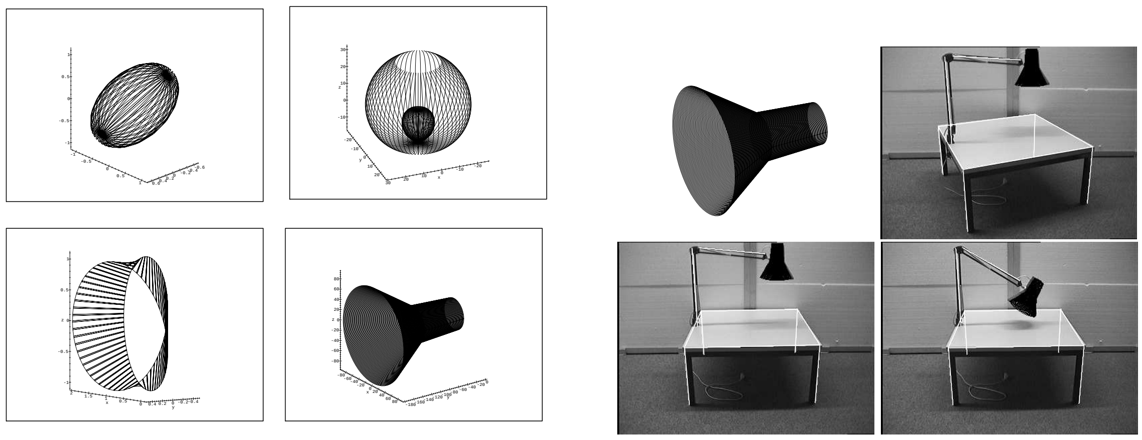


Figure 10: Left: 3D-3twist generated surfaces. Right: Pose estimation of a 2twist surface, connected via a 2 d.o.f. kinematic chain to a table.

of cycloidal curves can easily be estimated by just transforming the generating twist axes and the starting point on the curve. Furthermore is the construction of constraint curves not solely restricted to points. It is also possible to use e.g. lines or circles and to generate higher order constraint surfaces or volumes in space by transforming these entities with respect to different axes in space. This leads to special ruled surfaces which can be used as object entities. It is also possible to generate 3twist curves. For this, a point is transformed with respect to three twists. Surfaces can be modeled by rotating, e.g., the 2twist generated curves around a third twist with a second independent angle ϕ_2, leading to 3twist generated surfaces. Examples are shown in figure 10, left. Figure 10 right, shows experimental results of a 2twist surface. The object model is a lamp shade connected via a 2 d.o.f. kinematic chain to a table. In this experiment, the pose parameters and the angles of the kinematic chain are estimated. The lamp shade itself is modeled by two cone parts generated from circles. Note, that this curve representation belongs to a so-called parametric representation of a curve [7].

3.4 Free-form curves

So far we have presented 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 [16, 2]. In figure 11 left, a closed curve (a rose) is shown in the

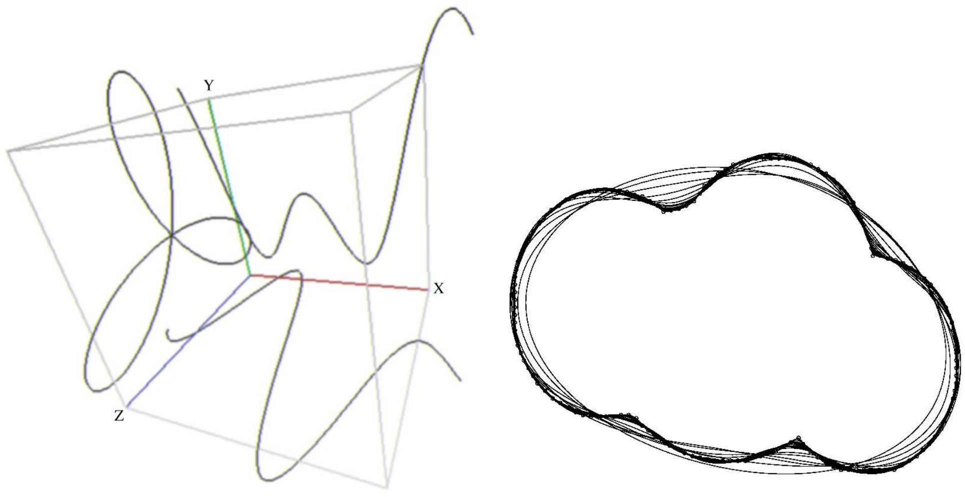


Figure 11: Left: Projections of a 1-parametric curve interpreted as time dependent function. Right:Different approximation levels of a 3D object contour.

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. The x-axis can be interpreted as an additional time axis. If we project $C_S(\phi)$ onto the xy-plane and xz-plane, we obtain the two other periodic curves (or signals) 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 $n+1$. These signals can now be interpreted and analyzed by using a discrete Fourier transform and inverse discrete Fourier transform. Indeed the twists of the virtually coupled kinematic chains can be interpreted as phase vectors in a complex plane generating a plane cycloidal curve. The constraint equations take the form

$$\left(\boldsymbol{M}\left(\mathsf{e}\wedge\left(\sum_{m=1}^{3}\sum_{k=-N}^{N}\boldsymbol{p}_k^m\exp\left(\frac{2\pi k\phi}{2N+1}\boldsymbol{l}_m\right)+\mathsf{e}_-\right)\right)\widetilde{\boldsymbol{M}}\right)\ \underline{\times}\ (\mathsf{e}\wedge(\boldsymbol{O}\wedge\boldsymbol{x}))=0.$$

The interpretation of this equation is simple: The innermost part contains the substituted Fourier descriptors in the conformal space. This is then coupled with the unknown rigid body motion (the motor $\boldsymbol{M}$) and compared with a reconstructed projection ray, given in a Plücker representation. The main point is the coupling of a spectral representation of contours within the perspective pose estimation problem. The advantage of this approach is, that a low-pass description of an object model is available which can be applied to gain a multi-scale representation of the object with respect to its complexity: This is shown in figure 11 right, by using only a subset of the Fourier descriptors.

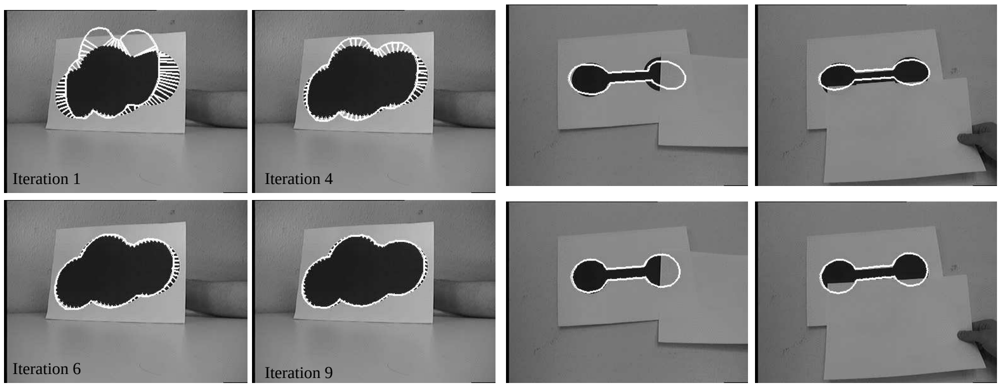


Figure 12: Left: Visualization of the ICP-algorithm during the iteration. Right: Different pose results for distorted image data. The first row shows results obtained with the non-modified ICP algorithm. The second row shows pose results obtained with the outlier-elimination during the ICP algorithm.

The main problem during pose estimation is to determine suited correspondences between 2D image features and points on the 3D model curve. Therefore a version of an ICP-algorithm [45] is used and applied. 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 on the
    3D contour to each projection ray.
(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. Note, that the point based pose estimation algorithm is involved in the contour based pose estimation algorithm. The principle of the ICP-algorithm during tracking is visualized in figure 12 left. There the chosen correspondences are also visualized by projecting them onto the image. Note, that the correspondences are chosen in the 3D space, since the 3D contour points are related to 3D projection rays.

The robustness of the algorithm with respect to distorted image data is shown in figure 12 right. In this image sequence (containing 450 images) the image contour is distorted 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. The first row of figure 12 shows the results obtained with a non-modified ICP-algorithm. It is already clarified, that the constraint equations express a geometric distance measure in the 3D space. Therefore it is easy to detect outliers and to implement an algorithm which automatically eliminates their equations. Some results of the modified algorithm are shown in the second row of figure 12 right. We call this procedure the *outlier-elimination* method. As can be seen, the obtained results are much better. But indeed, these examples give just a guess about the stability of the proposed method as it is impossible to compensate falsely extracted contours or too much missing information.

3.5 Pose estimation of multiple contours

Many 3D objects can more easily be represented by a set of 3D contours expressing the different aspects of the object. We will now extend the object model to a set of 3D contours. The main problem is, how to deal with occluded or partially occluded contour parts of the object. Here, the contours are assumed as rigidly coupled to each other.

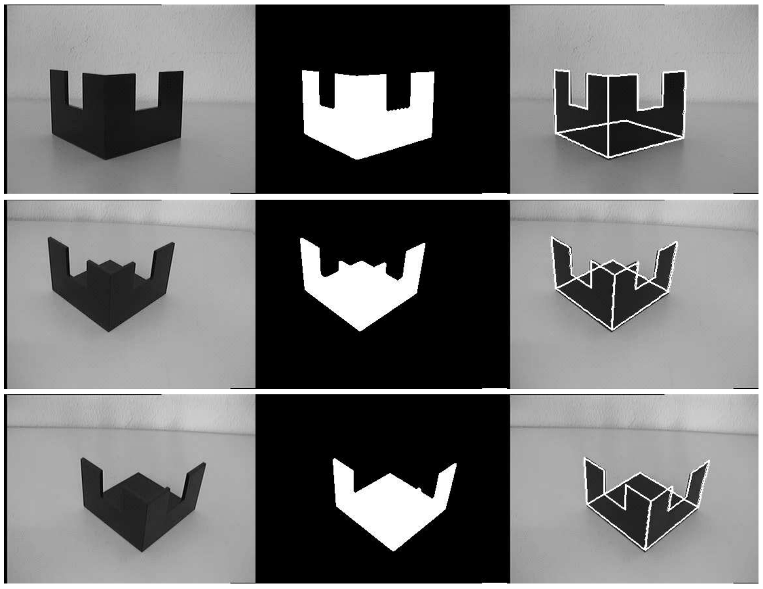


Figure 13: Pose results of an object with partially occluded contours. The left image shows the original image. The middle image shows the extracted silhouette and the right image visualizes the pose result.

This means that the pose of one contour automatically defines the pose of the other one. A modified ICP-algorithm leads to results shown in figure 13. Here the main idea is that all contours vote for a corresponding projection ray, but only the one with the smallest spatial distance is taken as correspondence.

3.6 Free-form Surfaces

The last extension we want to present is the representation of free-form surfaces and their embedding in the 2D-3D pose estimation problem. The idea is to extend the 1D-signal approach for 3D contours to a 2D signal model leading to 3D surfaces. We assume a two-parametric surface of the form

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

This means, we have three 2D functions $f^i(\phi_1, \phi_2) : \mathbb{R}^2 \to \mathbb{R}$ acting on the different base vectors $\mathbf{e}_i$. For a discrete number of sampled points, $f^i_{n_1,n_2}$, ($n_1 \in [-N_1, N_1]$; $n_2 \in [-N_2, N_2]$; $N_1, N_2 \in \mathbb{N}$) on the surface, we can now interpolate the surface by using a 2D discrete Fourier transform (2D-DFT) and then apply an inverse 2D discrete Fourier transform (2D-IDFT). The surface can therefore be approximated as a series expansion

$$F(\phi_1, \phi_2) \;\simeq \sum_{i=1}^{3} \sum_{k_1=-N_1}^{N_1} \sum_{k_2=-N_2}^{N_2} \boldsymbol{p}^i_{k_1,k_2} \exp\left(\frac{2\pi k_1 \phi_1}{2N_1+1}\, \boldsymbol{l_i}\right) \exp\left(\frac{2\pi k_2 \phi_2}{2N_2+1}\, \boldsymbol{l_i}\right).$$

Here we have replaced the imaginary unit $i = \sqrt{-1}$ with three different rotation axes, represented by the bivectors $\boldsymbol{l_i}$, with $\boldsymbol{l_i}^2 = -1$. The complex Fourier series coefficients are contained in the vectors $\boldsymbol{p}^i_{k_1,k_2}$ that lie in the plane spanned by $\boldsymbol{l_i}$. We will call them phase vectors. These vectors can be obtained by a 2D-DFT of the sample points $f^i_{n_1,n_2}$ on the surface. Again we gain the possibility of a low-pass object description by using only a subset of Fourier descriptors. Example images of a car model are shown in figure 14.

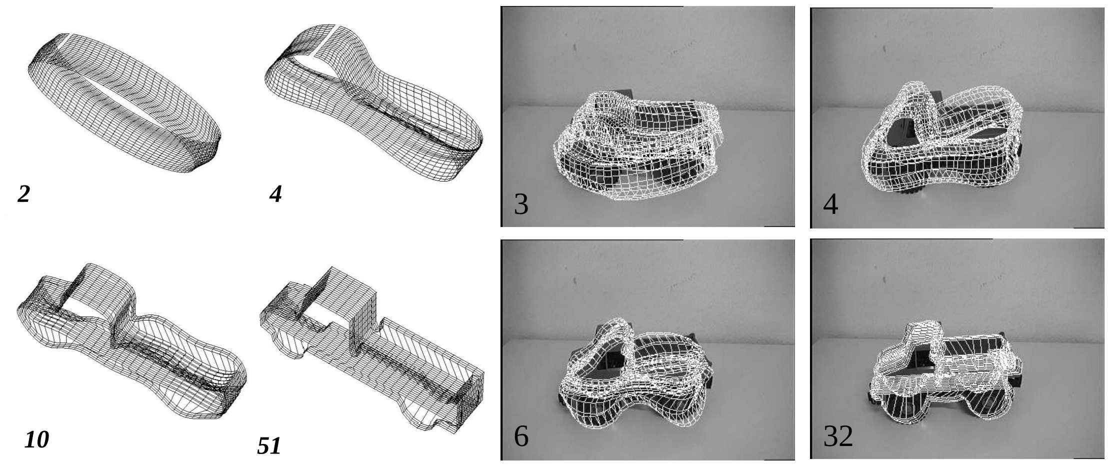


Figure 14: Left: Different approximation levels of the surface model. In the examples 2, 4, 10 and 51 Fourier descriptors are used. Right: Pose results of the low-pass contours during the ICP cycle.

We now continue with the algorithm for silhouette based pose estimation of surface models. In our scenario, we assume to have extracted the silhouette of an object in an image. To compare points on the image silhouette with the surface model, the idea is to work with those points on the surface model which lie on the outline of a 2D projection of the object.

This means we work with the 3D silhouette of the surface model with respect to the camera. To obtain this, the idea is to project the whole surface on a virtual image. Then the contour is calculated and from the image contour the 3D silhouette of the surface model is reconstructed. The contour model is then used within our contour based pose estimation algorithm. Note, that we make use of our contour-based pose estimation algorithm which is based on our point based pose estimation algorithm to deal with silhouette based free-form pose estimation of surface models.

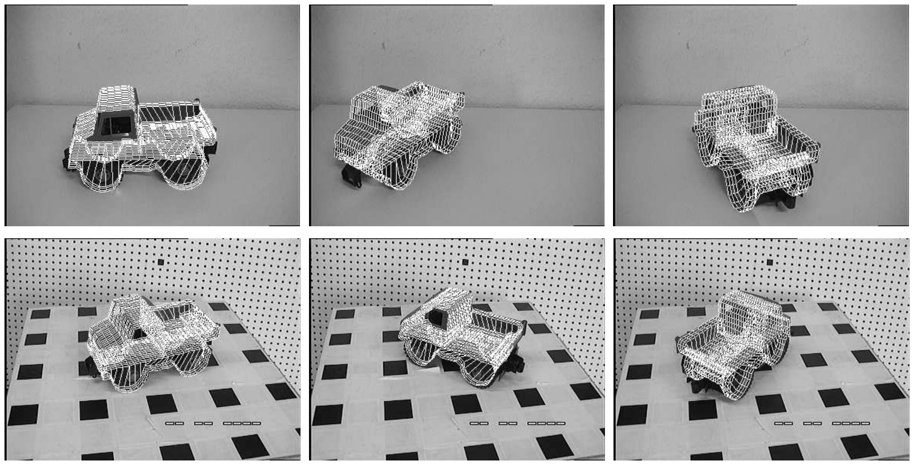


Figure 15: Different pose results of the object model.

The convergence behavior of the algorithm is shown in figure 14. As can be seen, we refine the pose results by using a low-pass approximation of the surface and by adding successively higher frequencies during the iteration. Figure 15 shows different pose results obtained with our algorithm. Note, that our algorithms are even able to deal with non-homogeneous background and with camera noise.

3.7 Dealing with multiple free-form surface patches

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

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

The main point is that we still work with one virtual 3D silhouette of the object which is now generated from the including free-form patches. This requires a modification of

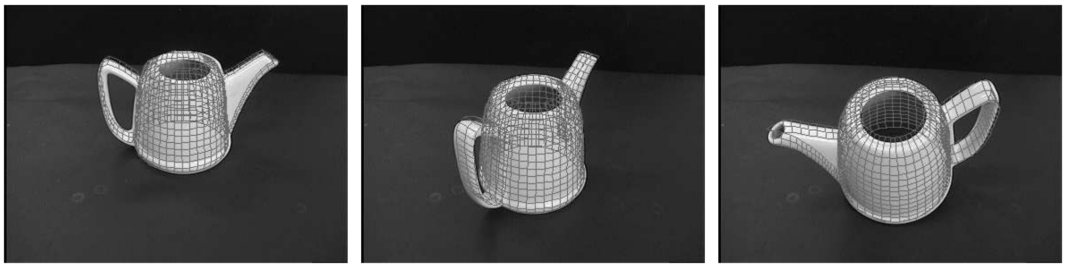


Figure 16: First pose results of the tea pot. The projected object model visualizes the quality of the 3D pose.

the 3D reconstruction algorithm from the virtual image, but the basic pose estimation algorithm is the same. It is still a contour based pose estimation algorithm including our outlier elimination method within the ICP-algorithm:

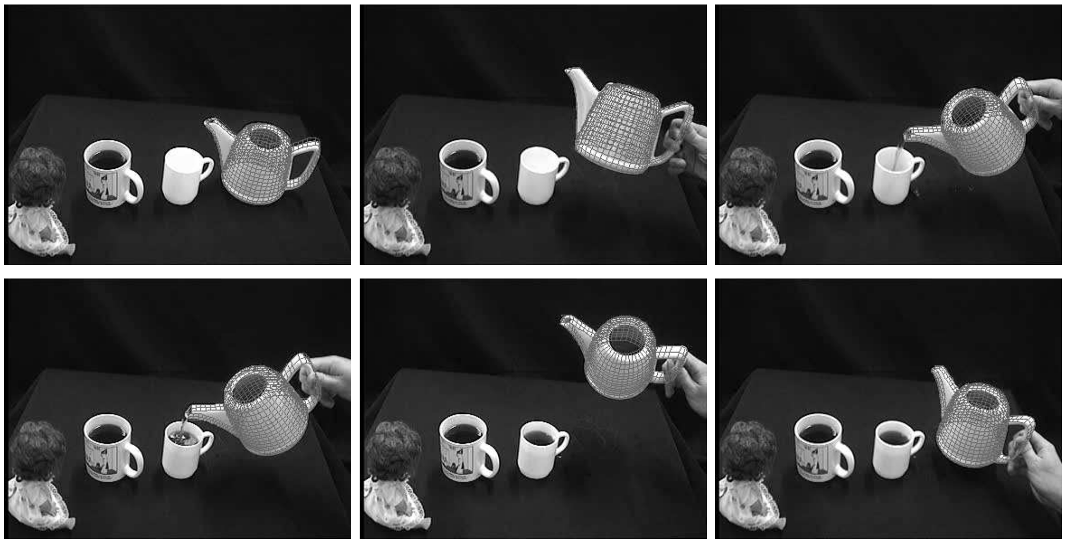


Figure 17: Example images of the tracked tea pot. The hand grasping the tea pot leads to outliers during the image silhouette extraction which are detected and eliminated during the pose estimation.

We assume an extracted image silhouette and start with the reconstruction of the image contour points to 3D projection rays. This reconstruction is only estimated once for each image. Then the parts of the object model are projected in a virtual image. Since we assume the surface parts as rigidly coupled, we extract and reconstruct one 3D silhouette of the surface model. To obtain this, we use in principle the approach presented in the previous section, but extent the contour reconstruction with the contour number.

Then we apply the 3D contour on our contour based pose estimation algorithm, which contains an ICP-algorithm and our gradient descent method for pose estimation. We then transform the surface model with the pose calculated from the contour based pose estimation algorithm and increase the low-pass approximation of each surface patch. Since the aspect of the object model can change after the iterated rigid transformations, we generate a new 3D silhouette: The algorithm continues with a new projection of the object model in a virtual image and the loop repeats till the algorithm converges. Figure 16 shows first pose results of the object model. It can be seen that partially occluded surface parts are no problem for the algorithm.

Figure 17 shows further example images during an other image sequence containing 350 images. The main aspect of this image sequence is to show that our algorithm is even able to deal with outliers during image processing which are caused by the human hand grasping the tea pot. This is done by our outlier elimination method as described in the previous section.

3.8 Coupling surfaces with kinematic chains

We now introduce how to couple kinematic chains within the surface model and present a pose estimation algorithm which estimates the pose and angle configurations simultaneously.

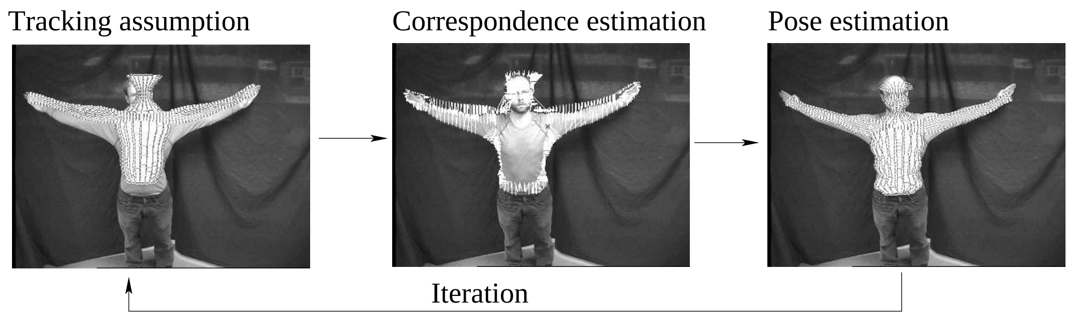


Figure 18: The basic algorithm: Iterative correspondence and pose estimation.

A surface is given in terms of three 2-parametric functions with respect to the parameters ϕ_1 and ϕ_2. Furthermore, we assume a set of joints J_i. By using an extra function $\mathcal{J}(\phi_1, \phi_2) \rightarrow [J_i | J_i : i\text{th. joint}]$, we are able to give every node a joint list along the kinematic chain. Note, that we use $[,]$ and not $\{,\}$, since the joints are given ordered along the kinematic chain. Since the arms contain two kinematic chains (for the left and right arm separately), we introduce a further index to separate the joints on the left arm from the ones on the right arm. The joints themselves are represented as objects in an extra field (a look-up table) and their parameters can be accessed immediately from the joint index numbers. Furthermore, it is possible to transform the location of the joints in space (as clarified in section 2.1). For pose estimation of a point $\underline{\boldsymbol{X}}^{\star}_{n,i_n}$ attached to the nth joint along the kinematic chain, we generate constraint equations of the form

$$(\boldsymbol{M}(\boldsymbol{M}_1 \ldots \boldsymbol{M}_n \underline{\boldsymbol{X}}^{\star}{}_{n,i_n} \widetilde{\boldsymbol{M}_n} \ldots \widetilde{\boldsymbol{M}_1}) \widetilde{\boldsymbol{M}}) \underline{\times}\, \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{n,i_n}) \;=\; 0.$$

To solve a set of such constraint equations we linearize the motor $\boldsymbol{M}$ with respect to the unknown twist Ψ and the motors $\boldsymbol{M}_i$ with respect to the unknown angles θ_i. The twists Ψ_i are known a priori.

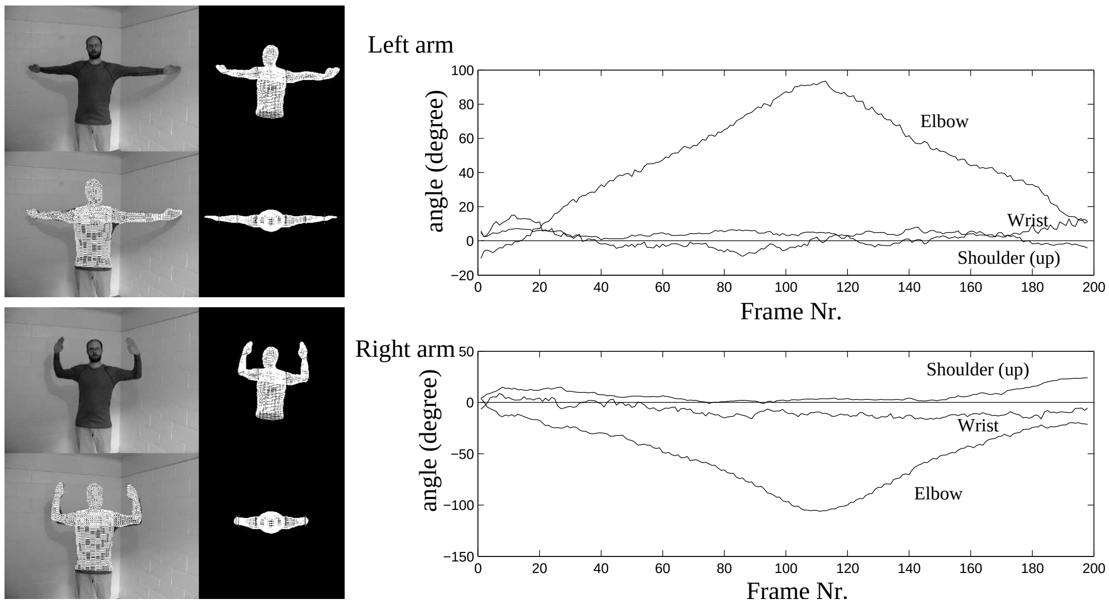


Figure 19: Left: First pose results with a 6 DOF kinematic chain. Right: Angles of the left and right arm during the tracked image sequence.

The basic pose estimation algorithm is visualized in figure 18: We start with simple image processing steps to gain the silhouette information of the person by using a color threshold and a Laplace operator. Then we project the surface mesh in a virtual image and estimate its 3D contour. Each point on the 3D contour carries a given joint index. Then we estimate the correspondences by using an ICP-algorithm, generate the system of equations, solve them, transform the object and its joints and iterate this procedure. During iteration we start with a low-pass object representation and refine it by using higher frequencies. This helps to avoid local minima during iteration.

First results of the algorithm are shown on the left of figure 19: The figure contains two pose results; it shows on each quadrant the original image and overlaid the projected 3D pose. The other two images show the estimated joint angles in a virtual environment to visualize the error between the ground truth and the estimated pose. The tracked image sequence contains 200 images. In this sequence we use just three joints on each arm and neglect the shoulder (back) joint. The right diagram of figure 19 shows the estimated angles of the joints during the image sequence. The angles can easily be identified with the sequence. Since the movement of the body is continuous, the estimated curves are also relatively smooth.

4 Summary

In these pages we gave a brief overview on foundations and developed algorithms for 2D-3D pose estimation. We began with a brief introduction to Clifford algebras and described how to model the pose scenario in the conformal geometric algebra. We used the language of Clifford algebras to develop and extend pose scenarios for various types of objects. Therefore we started with simple point and line based representations and continued via kinematic chains, cycloidal curves (twist generated curves) to general free-form contours and surfaces. We concluded with the coupling of surface models with kinematic chains to end up in the field of human motion estimation. Indeed, there are still many aspects on the pose scenario untouched and our future research will continue in various directions. The reader is welcome to check our homepages now and then ...

References

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

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

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

[4] Bouma T.A. From unoriented subspaces to blade operators In *Applied Geometric Algebras for Computer Science and Engineering (AGACSE), Dorst L., Doran C. and Lasenby J. (Editors)*, Birkhäuser Verlag, pp. 59-68, 2001.

[5] 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.

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

[7] 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.

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

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

[10] Dorst L. The inner products of geometric algebra. In *Applied Geometric Algebras for Computer Science and Engineering (AGACSE), Dorst L., Doran C. and Lasenby J. (Editors)*, Birkhäuser Verlag, pp. 35-46, 2001.

[11] Dorst L. Honing geometric algebra for its use in the computer sciences. In [40], pp. 127-152, 2001.

[12] 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.

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

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

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

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

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

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

[19] Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [40], pp. 3-23, 2001.

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

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

[22] Homepages Clifford (geometric) algebra
`http://www.ks.informatik.uni-kiel.de`
`http://www.clifford.org/`
`http://modelingnts.la.asu.edu/GC_R&D.html`
`http://www.mrao.cam.ac.uk/~clifford/`
`http://www.science.uva.nl/ga/`
`http://clifford.physik.uni-konstanz.de/~fauser/P_cl_people.shtml`
`http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Clifford.html`

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

[24] Lorusso A., Eggert D.W. and Fisher R.B. A comparison of four algorithms for estimating 3D rigid transformations. *Machine Vision and Applications*, Vol. 9, No. 5/6, pp. 272-290, 1997.

[25] 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.

[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 [40], pp. 27-52, 2001.

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

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

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

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

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

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

[34] Perwass C. and Hildenbrand D. Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space. An Introductory Tutorial. *Technical Report 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003. Available at `http://www.informatik.uni-kiel.de/reports/`

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

[36] Rosenhahn B., Sommer G. and Klette R. Pose Estimation of Free-form Objects (Theory and Experiments). *Technical Report CITR-TR-137*, Centre for Image Technology and Robotics, Tamaki Campus, University of Auckland, 2004. Available at `http://www.citr.auckland.ac.nz/techreports/`

[37] Rosenhahn B., Perwass Ch. and Sommer G. Free-form pose estimation by using twist representations *Algorithmica*, Received Mai 1, 2002, revised January 21, 2003, Communicated by R.C. Veltkamp, No. 38, pp 91-113, 2004.

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

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

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

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

[42] 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.

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

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

[45] 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.