Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computer Vision and Image Understanding 95 (2004) 54–71
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

Registration without ICP

Helmut Pottmann, Stefan Leopoldseder,* and Michael Hofer

Geometric Modeling and Industrial Geometry Group, Vienna University of Technology, Wiedner Hauptstrasse 8–10, A-1040 Wien, Austria

Received 10 February 2002; accepted 19 April 2004

Abstract

We present a new approach to the geometric alignment of a point cloud to a surface and to related registration problems. The standard algorithm is the familiar ICP algorithm. Here, we provide an alternative concept which relies on instantaneous kinematics and on the geometry of the squared distance function of a surface. The proposed algorithm exhibits faster convergence than ICP; this is supported both by results of a local convergence analysis and by experiments.

Keywords: Registration; Instantaneous kinematics; Squared distance function; Geometric optimization

1. Introduction

We investigate the following registration problem. Suppose that we have a *CAD model* from which a workpiece has been produced. This workpiece has been scanned with some 3D measurement device (laser range scanning, light sectioning, etc) resulting in a *3D data point cloud* from the surface of this workpiece. Thereby, the CAD model shall describe the 'ideal' shape of the object and will be available in a coordinate system that is different to that of the 3D data point set. For the goal of shape inspection it is of interest to find the optimal Euclidean motion (translation and rotation) that aligns, or registers, the point cloud to the CAD model. This makes it

* Corresponding author. Fax: +43-1-58801/11399.
E-mail addresses: pottmann@geometrie.tuwien.ac.at (H. Pottmann), stefan@geometrie.tuwien.ac.at (S. Leopoldseder), hofer@geometrie.tuwien.ac.at (M. Hofer).

1077-3142/$ - see front matter
doi:10.1016/j.cviu.2004.04.002

possible to check the given workpiece for manufacturing errors and to visualize and classify the deviations.

A well-known standard algorithm to solve such a registration problem is the iterative closest point (ICP) algorithm of Besl and McKay [1], which we will briefly summarize in Section 1.1. For an overview of the recent literature on this topic we refer to [5,6,19,20]. ICP is an iterative algorithm which in each step applies a motion to the current position of the point cloud. The motion is such that the points move in a least squares sense as close as possible to their closest points on the model shape. In Section 2, we investigate the squared distance function d^2 of a surface and see that ICP actually works with local quadratic approximants to d^2 which are very good for points far away from the surface, but not good at all for points close to the surface. In Section 3, we review basic facts from instantaneous kinematics. Our alternative approach to the registration problem, which is based on instantaneous kinematics and local quadratic approximants to d^2, is presented in Section 4. The new method shows a faster convergence behavior and is also applicable for other types of registration and positioning problems. This is demonstrated at hand of a number of examples. Theoretical support for the fast convergence is given in Section 5, where we survey the main results of a detailed study of the rate of convergence of registration algorithms [14]. Finally, in Section 6 we outline the many directions for future research which are opened by the present concept.

1.1. The ICP algorithm

The *iterative closest point (ICP) algorithm* has been introduced by Besl and McKay [1]. Independently, Chen and Medioni [4] proposed a similar algorithm, which we will address later on in this paper. An excellent summary with new results on the acceleration of the ICP algorithm has been given by Rusinkiewicz and Levoy [20], who also suggest that *iterative corresponding point* is a better expansion for the abbreviation ICP than the original *iterative closest point*.

The point set ('data' shape) is rigidly moved (registered, positioned) to be in best alignment with the CAD model ('model' shape). This is done iteratively: in the first step of each iteration, for every data point the closest point on the surface ('normal footpoint') of the CAD model is computed. This is the most time consuming part of the algorithm and has to be implemented efficiently. As a result of this first step one obtains a point sequence $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots)$ of closest model shape points to the data point sequence $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots)$. Each point $\mathbf{x}_i$ corresponds to the point $\mathbf{y}_i$ with the same index.

In the second step of each iteration the rigid motion m is computed such that the moved data points $m(\mathbf{x}_i)$ are closest to their corresponding points $\mathbf{y}_i$, where the objective function to be minimized is

$$F = \sum_{i=1}^{N} \|m(\mathbf{x}_i) - \mathbf{y}_i\|^2. \tag{1}$$

This least squares problem can be solved explicitly, see e.g., [1,7,10]. The translational part of m brings the center of mass of X to the center of mass of Y. The

rotational part of m can be obtained as the unit eigenvector that corresponds to the maximum eigenvalue of a symmetric 4×4 matrix. The solution eigenvector is nothing but the unit quaternion description of the rotational part of m.

After this second step the positions of the data points are updated via $X_{\text{new}} = m(X_{\text{old}})$. Now step 1 and step 2 are repeated, always using the updated data points, until the change in the mean-square error falls below a preset threshold. Since the value of the objective function decreases both in step 1 and 2, the ICP algorithm always converges monotonically to a local minimum.

2. Local quadratic approximants of the squared distance function of curves and surfaces

The algorithm we are proposing heavily relies on local quadratic approximants to the squared distance function of the surface Φ to which the point cloud should be registered. For a derivation and proofs of the following results we refer the reader to [17]. For a better understanding, we first present local quadratic approximants to planar curves and then generalize the obtained results to surfaces and space curves.

2.1. Local quadratic approximants of the squared distance function to a planar curve

In Euclidean 3-space $\mathbb{R}^3$, we consider a planar C^2 curve $\mathbf{c}(t)$ with parameterization $(c_1(t), c_2(t), 0)$. The Frenet frame at a curve point $\mathbf{c}(t)$ consists of the unit tangent vector $\mathbf{e}_1 = \dot{\mathbf{c}}/\|\dot{\mathbf{c}}\|$ and the normal vector $\mathbf{e}_2(t)$, see Fig. 1. The two vectors form a right-handed Cartesian system in the plane. With $\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2 = (0,0,1)$ this system is extended to a Cartesian system Σ in $\mathbb{R}^3$. Coordinates with respect to Σ are denoted by (x_1, x_2, x_3). The system Σ depends on t and shall have the curve point $\mathbf{c}(t)$ as origin.

At least locally, the shortest distance of a point $\mathbf{p} = (0, d, 0)$ on the x_2-axis (curve normal) is its x_2-coordinate d. For each t, locally the graph points $(0, d, d^2)$ of the squared distance function form a parabola p in the normal plane of $\mathbf{c}(t)$. The graph surface Γ of the squared distance function d^2 of $\mathbf{c}(t)$ therefore contains parabolas in the cross-sections with vertical planes orthogonal to the curve $\mathbf{c}(t)$. The distance function d^2 of a smooth curve $\mathbf{c}(t)$ is smooth except for the points of the cut locus of the curve $\mathbf{c}(t)$. This can be seen very clearly for the graph surface Γ of d^2 in Fig. 2.

When we now study local quadratic (Taylor) approximants of the squared distance function d^2 in a point $\mathbf{p}$, we exclude points of the cut locus of $\mathbf{c}(t)$. For points $\mathbf{p}$ of the cut locus the closest footpoint on the curve $\mathbf{c}(t)$ is not unique, and the distance function d^2 is not differentiable.

Consider a point $\mathbf{p}$ in π whose coordinates in the Frenet frame at the normal footpoint $\mathbf{c}(t_0)$ are $(0, d)$, see Fig 1. The curvature center $\mathbf{k}(t_0)$ at $\mathbf{c}(t_0)$ has coordinates $(0, \rho)$. Here, ρ is the inverse curvature $1/\kappa$ and thus has the same sign as the curvature, which depends on the orientation of the curve.

Proposition 1. *In the Frenet frame, the second order Taylor approximant F_d of the squared distance function d^2 at $(0, d)$ is given by*

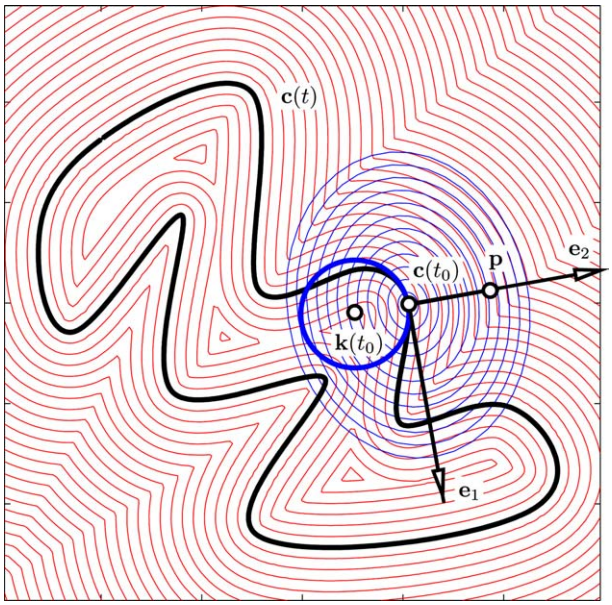

Fig. 1. Planar curve $\mathbf{c}(t)$ with Frenet frame $\mathbf{e}_1$, $\mathbf{e}_2$ in $\mathbf{c}(t_0)$. The squared distance function d^2 to this curve and the local quadratic approximant of this function in the point $\mathbf{p}$ are visualized by level sets.

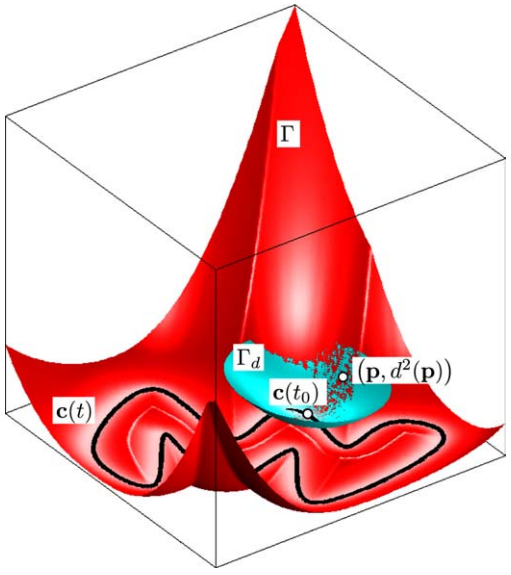

Fig. 2. Axonometric view of Fig. 1. Planar curve $\mathbf{c}(t)$, graph surface Γ of its squared distance function d^2, and graph surface of a local quadratic approximant at the point $\mathbf{p}$.

$$F_d(x_1, x_2) = \frac{d}{d-\rho} x_1^2 + x_2^2. \tag{2}$$

For a derivation of this result and a discussion of the different types of the graph surface Γ_d of F_d we refer the reader to [17]. In Figs. 1 and 2, the second order Taylor approximant F_d at the point $\mathbf{p}$ is depicted. The graph surface Γ_d of F_d and the graph surface Γ of d^2 have second order contact in the point $(\mathbf{p}, d^2(\mathbf{p}))$.

2.2. Local quadratic approximants of the squared distance function of a surface

Consider an oriented surface $\mathbf{s}(u, v)$ with a unit normal vector field $\mathbf{n}(u, v) = \mathbf{e}_3(u, v)$. At each point $\mathbf{s}(u, v)$, we have a local right-handed Cartesian system whose first two vectors $\mathbf{e}_1, \mathbf{e}_2$ are determined by the principal curvature directions. The latter are not uniquely determined at an umbilical point. There, we can take any two orthogonal tangent vectors $\mathbf{e}_1, \mathbf{e}_2$. We will refer to the thereby defined frame as *principal frame* $\Sigma(u, v)$. Let κ_i be the (signed) principal curvature to the principal curvature direction $\mathbf{e}_i$, $i = 1, 2$, and let $\rho_i = 1/\kappa_i$. Then, the two principal curvature centers at the considered surface point $\mathbf{s}(u, v)$ are expressed in Σ as $\mathbf{k}_i = (0, 0, \rho_i)$. The quadratic approximant F_d to the squared distance function d^2 at $\mathbf{p} = (0, 0, d)$ is the following.

Proposition 2. *The second order Taylor approximant of the squared distance function of a surface at a point* $\mathbf{p}$ *is expressed in the principal frame at the normal footpoint via*

$$F_d(x_1, x_2, x_3) = \frac{d}{d-\rho_1} x_1^2 + \frac{d}{d-\rho_2} x_2^2 + x_3^2. \tag{3}$$

Let us look at two important special cases.

- For $d = 0$ we obtain

$$F_0(x_1, x_2, x_3) = x_3^2. \tag{4}$$

This means that the second order approximant to d^2 at a surface point $\mathbf{p}$ is the same for the surface Φ and for its *tangent plane* at $\mathbf{p}$. Thus, if we are close to the surface, the squared distance function of the tangent plane at the closest point to the surface is a very good approximant. At least at first sight it is surprising that the tangent plane, which is just a first order approximant, yields a second order approximant when we are considering the squared distance function d^2, of surface and tangent plane, respectively.

- For $d = \infty$ we obtain

$$F_\infty(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2. \tag{5}$$

This is the squared distance to the footpoint on the surface.

We see that distances to normal footpoints, which are used in ICP, are just good if we are in a greater distance to the surface Φ. In the vicinity of the surface, it is much better to use other local quadratic approximants. The simplest

one is the squared distance to the tangent plane at the normal footpoint. Registration typically starts with a rough guess of the correct position obtained for example via principal component analysis, matching special surface features or taking into account some preknowledge on surface and point cloud. Hence for optimal alignment we typically need several iteration steps in the vicinity of the surface. This is the reason why we are not minimizing distances to the normal footpoints.

For our alignment algorithm, cf. Section 4, we prefer the Taylor approximants to be nonnegative, because then we are guaranteed to minimize positive definite quadratic functions in each iteration step. Thus, if one of the coefficients $d/(d-\rho_i)$ in (3) is negative we replace it by zero or by $|d|/(|d|+|\rho_i|)$, see [17] for details.

2.3. Local quadratic approximants of the squared distance function of a space curve

In case that boundary curves of surfaces are involved, it is also useful to know about local quadratic approximants of the squared distance function d^2 of a space curve. Given a point $\mathbf{p}$ in $\mathbb{R}^3$, the shortest distance to a C^2 space curve $\mathbf{c}(t)$ occurs along a normal of the curve or at a boundary point of it. The latter case is trivial and thus we exclude it. At the normal footpoint $\mathbf{c}(t_0)$ we form a Cartesian system with $\mathbf{e}_1$ as tangent vector and $\mathbf{e}_3$ in direction of the vector $\mathbf{p}-\mathbf{c}(t_0)$. This *canonical frame* can be viewed as limit case of the principal frame for surfaces, when interpreting the curve as a pipe surface with vanishing radius. By this limit process, we can also show the following result.

Proposition 3. *The second order Taylor approximant of the squared distance function of a space curve* $\mathbf{c}(t)$ *at a point* $\mathbf{p}$ *is expressed in the canonical frame* Σ *at the normal footpoint via*

$$F_d(x_1,x_2,x_3)=\frac{d}{d-\rho_1}x_1^2+x_2^2+x_3^2. \tag{6}$$

Here, $(0,0,\rho_1)$ *are the coordinates (in* Σ*) of the intersection point of the curvature axis of* $\mathbf{c}(t)$ *at the footpoint* $\mathbf{c}(t_0)$ *with the perpendicular line* $\mathbf{pc}(t_0)$ *from* $\mathbf{p}$ *to* $\mathbf{c}(t)$.

3. Instantaneous kinematics

For the algorithm we propose, some knowledge about kinematics is essential. Thus, in this section we briefly outline the basic facts we are using later on. Consider a differentiable one-parameter rigid body motion in Euclidean 3-space. Introducing Cartesian coordinate systems in the moving system Σ and in the fixed system Σ_0, the time-dependent position $\mathbf{x}_0(t)$ of a point $\mathbf{x}\in\Sigma$ in the fixed system is given by

$$\mathbf{x}_0(t)=\mathbf{a}(t)+M(t)\mathbf{x}. \tag{7}$$

Here, the time dependent orthogonal matrix $M(t)$ represents the spherical component of the motion, and $\mathbf{a}(t)$ describes the trajectory of the origin of the moving system. All arising functions shall be C^1. By differentiation we get the velocity vectors. It is well-known that the velocity vector field is linear at any time instant. More precisely, at any time instant there exist vectors $\mathbf{c}, \bar{\mathbf{c}}$ such that the velocity vector $\mathbf{v}(\mathbf{x})$ of any point $\mathbf{x}$ of the moving body can be computed as

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}. \tag{8}$$

Note that in this formula all arising vectors are represented in the same system; this may be the moving or the fixed system. The meaning of $\mathbf{c}, \bar{\mathbf{c}}$ is as follows: $\bar{\mathbf{c}}$ represents the velocity vector of the origin, and $\mathbf{c}$ is the so-called Darboux vector (vector of angular velocity).

It is well-known that only very special one-parameter motions have a constant, i.e., time-independent velocity vector field. These motions are:

- A *translation* with constant velocity (if $\mathbf{c} = 0$).
- A uniform *rotation* about an axis (if $\mathbf{c} \cdot \bar{\mathbf{c}} = 0$).
- A uniform *helical motion* (if $\mathbf{c} \cdot \bar{\mathbf{c}} \neq 0$).

Thus, up to the first differentiation order, any motion agrees locally with one of these motions. The most general case is that of a uniform helical motion, which is the superposition of a rotation with constant angular velocity about an axis G and a translation with constant velocity parallel to G. If the moving body rotates about an angle α, the translation distance is $p \cdot \alpha$. The constant factor p is referred to as *pitch* of the helical motion. For more details on helical motions and the close relations to line geometry we refer to [18]. The Plücker coordinates $(\mathbf{g}, \bar{\mathbf{g}})$ of the axis G, the pitch p and the angular velocity ω are computed from $\mathbf{c}, \bar{\mathbf{c}}$ as

$$\mathbf{g} = \frac{\mathbf{c}}{\|\mathbf{c}\|}, \qquad \bar{\mathbf{g}} = \frac{\bar{\mathbf{c}} - p\mathbf{c}}{\|\mathbf{c}\|}, \qquad p = \frac{\mathbf{c} \cdot \bar{\mathbf{c}}}{\mathbf{c}^2}, \qquad \omega = \|\mathbf{c}\|. \tag{9}$$

Recall that the Plücker coordinates of a line G consist of a direction vector $\mathbf{g}$ and the moment vector $\bar{\mathbf{g}} = \mathbf{p} \times \mathbf{g}$, where $\mathbf{p}$ represents an arbitrary point on G.

4. Registration of a point cloud to a CAD model using instantaneous kinematics and quadratic approximants of the squared distance function

In the ICP algorithm the data points $\mathbf{x}_i$ are moved towards their closest points $\mathbf{y}_i$ on the model surface Φ. Instead of moving $\mathbf{x}_i$ towards $\mathbf{y}_i$ we aim at bringing the points just closer to the surface Φ. For this, we employ local quadratic approximants of the squared distance function of Φ. As we have seen in Section 2.2, the squared distance functions of the tangent planes of Φ approximate the squared distance function of Φ very well in the vicinity of the surface. The aim is the same as for ICP. We would like to apply a motion to the point cloud such that the sum

$$f = \sum_{i=1}^{N} d^2(m(\mathbf{x}_i), \Phi) \tag{10}$$

of squared distances of the displaced points $m(\mathbf{x}_i)$ to the model surface Φ becomes minimal. Let us first give an overview of the proposed algorithm and then study the individual steps in more detail. The new algorithm iteratively applies the following steps:

1. To each point $\mathbf{x}_i$ of the current position of the point cloud, compute a local quadratic approximant F_i of the squared distance function of the surface Φ, as outlined in Section 2. In the simplest case, take as F_i the squared distance function of the tangent plane at the point $\mathbf{y}_i \in \Phi$, which is closest to $\mathbf{x}_i$. This step is used to quadratically approximate the function f to be minimized.
2. Compute a velocity vector field, which attaches to each point a velocity vector $\mathbf{v}(\mathbf{x}_i)$ such that the quadratic function $\sum F_i(\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i))$ assumes a minimal value. This step estimates a motion towards the model surface, but does not yet represent a Euclidean motion. From the point of view of optimization, we use a quadratic approximation of f and a linearization of the constraint (i.e., displacement by a rigid body motion), but we do not yet fulfil the constraint.
3. From the velocity field we compute a Euclidean displacement which displaces the points $\mathbf{x}_i$ in nearly the same way as the velocity vectors (used for the minimization in the previous step) would do. In terms of optimization, this is the step where we project onto the constraint manifold.

The details to the individual steps are as follows.

Step 1. We explain this step for squared tangent plane distances, and comment on more general quadratic approximants of the squared distance function later on. For each data point $\mathbf{x}_i \in X$ determine the nearest point $\mathbf{y}_i$ of the surface of the CAD model and determine the tangent plane there. Let $\mathbf{n}_i$ denote a unit normal vector of this tangent plane in $\mathbf{y}_i$. If $\mathbf{y}_i$ is no boundary point of the surface, $\mathbf{x}_i$ lies on the surface normal in $\mathbf{y}_i$, i.e., $\mathbf{x}_i = \mathbf{y}_i + d_i\mathbf{n}_i$ with d_i denoting the oriented Euclidean distance of $\mathbf{x}_i$ to $\mathbf{y}_i$.

In case that $\mathbf{y}_i$ is a boundary point, one will define $\mathbf{n}_i = (\mathbf{x}_i - \mathbf{y}_i)/\|\mathbf{x}_i - \mathbf{y}_i\|$, i.e., $\mathbf{n}_i$ is orthogonal to the boundary curve in $\mathbf{y}_i$, pointing in the direction of $\mathbf{x}_i$. Again we have $\mathbf{x}_i = \mathbf{y}_i + d_i\mathbf{n}_i$.

Note that depending on the application one may reject a data point $\mathbf{x}_i$ in the minimization process, if its closest surface point $\mathbf{y}_i$ lies on the boundary. This is necessary, for instance, when partial scans of the same object are registered.

For a triangulated surface one may estimate the tangent plane at $\mathbf{y}_i$ by local methods, e.g., as a local regression plane, and thus define the surface normal vector $\mathbf{n}_i$. Then, the data point $\mathbf{x}_i$ will not lie exactly on the surface normal in $\mathbf{y}_i$ and we have $\mathbf{x}_i = \mathbf{y}_i + d_i\mathbf{n}_i + \mathbf{t}_i$, where $\mathbf{t}_i$ is a vector parallel to the tangent plane in $\mathbf{y}_i$. Since we are here interested in squared tangent plane distances only, this tangential component $\mathbf{t}_i$ does not matter. All the following formulae of Step 2 are still valid.

Step 2. A linearization of the motion is equivalent to the use of instantaneous kinematics. The use of instantaneous kinematics for registration appears in other papers as well (see e.g., [3,5]), maybe for the first time in [2].

The velocity vector field of an instantaneous helical motion is given by $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. To each point $\mathbf{x}_i$ we attach a velocity vector $\mathbf{v}(\mathbf{x}_i) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i$. The distance of $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ to the tangent plane of the parametric surface in the point $\mathbf{y}_i$ is given by

$$d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i). \tag{11}$$

Now, minimization of the objective function (which is quadratic in $\mathbf{c}, \bar{\mathbf{c}}$)

$$F(C) := F(\mathbf{c}, \bar{\mathbf{c}}) = \sum_i (d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i))^2, \tag{12}$$

yields the pair $(\mathbf{c}, \bar{\mathbf{c}})$ that determines the helical motion whose velocity vector field we are using. The minimization can be solved using a system of linear equations. For that we rewrite (11) as

$$d_i + \mathbf{n}_i \cdot \bar{\mathbf{c}} + (\mathbf{x}_i \times \mathbf{n}_i) \cdot \mathbf{c} \;=\; d_i + (\mathbf{x}_i \times \mathbf{n}_i, \mathbf{n}_i) \begin{pmatrix} \mathbf{c} \\ \bar{\mathbf{c}} \end{pmatrix} \;=\; d_i + A_i C, \tag{13}$$

where A_i and $C := (\mathbf{c}, \bar{\mathbf{c}})^{\mathrm{T}}$ are one-by-six and six-by-one matrices, respectively. We use this notation to rewrite the objective function (12) as

$$\begin{aligned} F(C) &= \sum_i (d_i + A_i C)^2 \\ &= \sum_i d_i^2 + 2 \sum_i d_i A_i C + \sum_i C^{\mathrm{T}} A_i^{\mathrm{T}} A_i C \\ &= D + 2B^{\mathrm{T}} C + C^{\mathrm{T}} A C, \end{aligned} \tag{14}$$

where A is a symmetric, in general positive definite six-by-six matrix, B is a column vector with six entries, and D is just some scalar.

It is well-known that the unique minimum of the quadratic function $F(C)$ solves the linear system

$$AC + B = 0. \tag{15}$$

Remark 1. Instantaneous kinematics as described in Section 3 has been used in the context of reverse engineering of 'kinematic surfaces,' i.e., planes, general cylinders, surfaces of revolution, and helical surfaces (cf. [16,18]). These surfaces are characterized by the fact that there exists a vector field $\mathbf{v}(\mathbf{x}) = \mathbf{c} + \bar{\mathbf{c}} \times \mathbf{x}$ such that for each surface point $\mathbf{p}$ the vector $\mathbf{v}(\mathbf{p})$ is tangential to the surface in $\mathbf{p}$.

In the context of registration, these kinematic surfaces play a special role as well. After the registration of a point cloud to such a surface, the point cloud can still be moved tangentially to the surface without increasing the objective function $F(C)$ in Eq. (14). Thus, in the special case of kinematic surfaces the linear system (15) gets ill-conditioned. Whereas the standard ICP algorithm heavily punishes tangential movement (which slows the convergence behavior), minimizing $F(C)$ in Eq. (14) does not restrict tangential movement at all.

It is straightforward to combine the functional $F(C)$ with a functional

$$F'(C) := \sum_i (\mathbf{x}_i - \mathbf{y}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2,$$

which describes the sum of squared distances of the points $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ to the normal footpoints $\mathbf{y}_i$. Minimizing the quadratic functional $\bar{F}(C) = F(C) + \omega F'(C)$, where ω is a small but positive weight, again leads to the solution of a linear system.

Step 3. Moving each point $\mathbf{x}_i$ by $\mathbf{v}(\mathbf{x}_i)$, i.e., $\mathbf{x}_i \mapsto \mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$, (as we have assumed for the minimization) would not yield a Euclidean rigid body motion, but an affine one. Therefore, we use the underlying helical motion determined by $(\mathbf{c}, \bar{\mathbf{c}})$ from which we can calculate axis G and pitch p with Eq. (9).

We apply a rotation about this axis G through an angle of $\alpha = \arctan \|\mathbf{c}\|$ and a translation parallel to G by the distance $p \cdot \alpha$ (see Fig. 3). This motion brings each point $\mathbf{x}_i$ to a position $\mathbf{x}_i'$ close to $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ which has been used for the minimization in (12).

Using the underlying helical motion is furthermore justified by the fact that for $\mathbf{x}_i$ we do not know the exact corresponding point on the surface anyway, we are moving the point closer to the tangent plane, and we iterate the whole procedure to find the optimal match.

As a termination criterion for the iteration we use the change in the mean squared distances of $\mathbf{x}_i$ to the surface. We terminate the algorithm if this value falls below a certain threshold.

Example 1. The main application we have in mind is the quality inspection of industrial products. Here, the goal is to find the best alignment between the (exact) CAD model of a given workpiece, and a dense point cloud which has been obtained from the workpiece with a 3D scanning device.

In our first example the technical object is an air intake, which is represented as a triangulated surface model. The size of the object is approximately $0.25 \times 0.24 \times 0.18$ U. Let us first take a set of data points X (containing 2000 points), generated synthetically, where no Gaussian noise is added. Theoretical results on the convergence behavior of our algorithm are discussed in Section 5, and our experimental results support the results of quadratic convergence in case of a zero residual problem, i.e., in the case of X fitting exactly onto the target surface.

Fig. 4 shows the the triangulated surface model, together with the initial position of the point cloud X and its final position after 12 iteration steps of our algorithm. In each iteration step, a helical motion (cf. Fig. 3) is applied to the data points, until the point cloud reaches its final position.

Section 5 is devoted to the evaluation of the convergence rate of our algorithm and of the standard ICP algorithm. Let $X_j = \{\mathbf{x}_{i,j}\}$ denote the position of the data

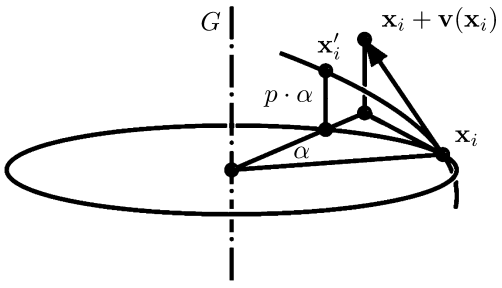

Fig. 3. New position $\mathbf{x}_i'$ of a point $\mathbf{x}_i$.

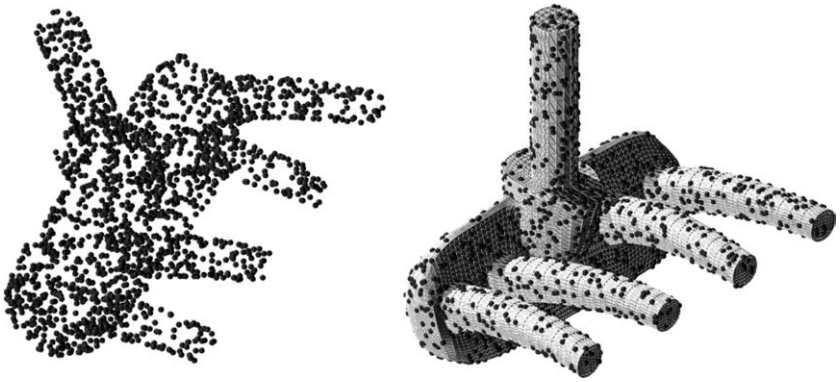

Fig. 4. Registration of a point cloud to a surface. Initial position (point cloud displaced to the upper left), and final position (point cloud in correct alignment with surface model).

point cloud X after iteration j, and $X^* = \{\mathbf{x}_i^*\}$ the final position (minimizer). For the convergence rate analysis we use the root mean squared error

$$E(j) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_{i,j} - \mathbf{x}_i^*\|^2}.$$

Note that $E(j)$ defines a distance of the point cloud X_j to the final position X^*, i.e., an error measure in the sense of optimization. $E(j)$ is not the value of the objective function F of Eq. (14) which is minimized in each iteration step of the algorithm.

In Table 1 the errors $E(j)$ for our algorithm and for the standard ICP algorithm are given. Our algorithm stops after 12 iterations with an error $E(12) = 1.4\text{e}-13$,

Table 1
Root mean squared errors for zero residual problem

j	Our algorithm			Standard ICP		
	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$
0	0.303740	—	—	0.303740	—	—
1	0.193313	0.6364	2.0953	0.067109	0.2209	0.7274
2	0.099390	0.5141	2.6596	0.042261	0.6297	9.3837
3	0.055732	0.5607	5.6417	0.030992	0.7333	17.3532
4	0.041180	0.7389	13.2581	0.024573	0.7928	25.5829
5	0.031754	0.7711	18.7247	0.019396	0.7893	32.1200
6	0.025268	0.7957	25.0590	0.014387	0.7417	38.2431
7	0.019248	0.7617	30.1473	0.010413	0.7237	50.3015
8	0.010184	0.5290	27.4857	0.007595	0.7293	70.0463
9	0.002835	0.2784	27.3398	0.005602	0.7376	97.1224
10	1.43e−4	0.0505	17.8300	0.004199	0.7495	133.7838
11	2.28e−7	0.0015	11.1393	0.003191	0.7600	180.9824
12	1.40e−13	6.23e−7	2.7254	0.002470	0.7740	242.5067
...	—	—	—	...	...	...
100	—	—	—	7.46e−12	0.8003	8.63e+10

Quadratic convergence for our algorithm and linear convergence of standard ICP algorithm.

whereas ICP still has $E(100) = 7.46\text{e}-12$ after 100 iterations. Furthermore, we have clear numerical evidence that our algorithm exhibits quadratic convergence for a zero residual problem, whereas ICP shows linear convergence in this case, see Section 5 for details. The quotient $E(j)/E(j-1)^2$ is approximately constant for our algorithm, whereas it tends to infinity for the ICP algorithm. For ICP the quotient $E(j)/E(j-1)$ is approximately a constant (smaller than 1), showing linear convergence.

After looking at the zero residual case we will now consider data points with Gaussian noise. Here, we expect linear convergence both in our algorithm and in the ICP algorithm, according to Section 5. We take the same surface model and the same initial position of the point cloud (again 2000 data points), but now Gaussian noise ($\sigma = 0.0005$) is added. Except for the noise there is no difference to the situation in Fig. 4.

Table 2 shows the convergence behavior for the disturbed data set which yields linear convergence both for our algorithm and for the ICP algorithm. Our algorithm stops after seventeen iterations. Still the quotient $E(j)/E(j-1)$ is lower for our algorithm, except for the first iteration step. This is not surprising, since in the first iteration almost no tangential movement is necessary, thus the ICP algorithm is superior. In the later iterations of the registration there is usually a substantial tangential movement involved, and this slows down the ICP algorithm considerably.

Remark 2. We have described the algorithm in its simplest form. There are many ways to improve it, and actually many ideas for improvement of ICP and related registration algorithms (cf. e.g. [20,22]) work as well. For example, we will not work in each step with all data points, but just with a random sample. Of course, more sophisticated sampling methods, e.g., by choosing data points with a good distribution of estimated normals (cf. [9,20]), can be applied as well. Furthermore, one may reject a chosen data point, e.g., if its distance to the normal footpoint exceeds some threshold. Moreover, it is straightforward to extend the objective function (12) to a weighted scheme. There are 3D measurement devices that supply for each data

Table 2
Root mean squared errors for point data with Gaussian noise

j	Our algorithm			Standard ICP		
	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$
0	0.303180	—	—	0.303180	—	—
1	0.176332	0.5816	1.9184	0.064936	0.2142	0.7064
2	0.073704	0.4180	2.3704	0.040396	0.6221	9.5800
3	0.032686	0.4435	6.0171	0.029369	0.7270	17.9972
...				...	...	...
15	1.39e−10	0.2264	3.67e+8	1.33e−3	0.8088	4.93e+2
16	2.93e−11	0.2099	1.50e+9	1.07e−3	0.8092	6.09e+2
17	8.42e−12	0.2876	9.82e+9	8.74e−4	0.8131	7.57e+2
...	—	—	—	...	...	...
100	—	—	—	2.76e−10	0.7856	2.23e+9

Linear convergence both for our algorithm and for the standard ICP algorithm.

point a tolerance for the occurring measurement errors. These can be included in the objective function to downweight outliers. This is especially important for a precise final alignment, but has less impact on the convergence speed of the iterative registration algorithm. The inclusion of more complete knowledge on the measurement error properties (see [13]) seems to be possible; this is an interesting topic for future research.

As indicated above, we can further improve the quadratic approximation of f by the use of second order Taylor approximants, say F_i, to the squared distance function at the current data point position $\mathbf{x}_i$. In view of Section 2.2, it is more complicated to compute these F_i's, but the remaining part of the algorithm is the same. In step 2, we still have to minimize a quadratic function F, and in step 3 we perform the same position correction. Working with general Taylor approximants F_i is more subtle, however. To make sure that F is positive definite, we use *nonnegative* quadratic approximants to d^2. One way to compute those has been presented in [17].

Example 2. In the second example, data points have been taken from a human face by a data aquisition method using color-coded structured light. This data contains 102761 points but only 300 randomly chosen points are taken in each iteration step to determine the iterative motion. The point data shall be aligned with a model shape of a 'generic' face, which is given as a triangulated mesh, see Fig. 5. The goal of this application is to bring the point data of the faces in a standard position such that further processing steps can be applied to the data.

Note that the face data points and the triangulated model shape do not come from the same face, so the data will not necessarily fit to the model very well. One further important point is, that the point data set and the model surface are given in different scales, the size of the model surface is approximately 0.85 times the size of the point data. Therefore, an extension of our algorithm is used which allows a

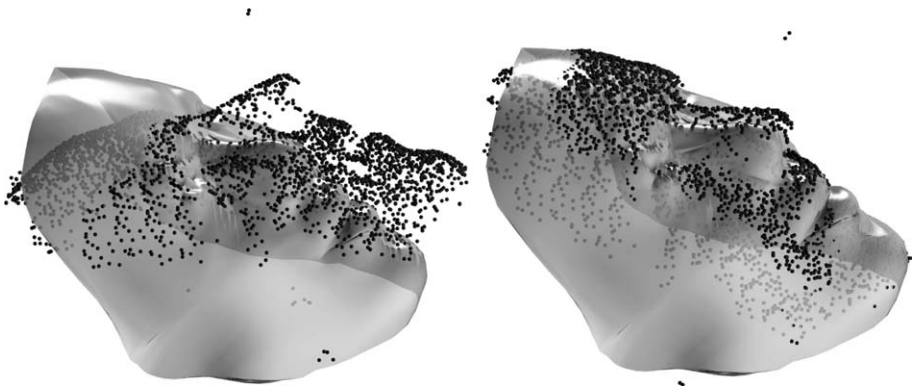

Fig. 5. Registration of a 3D laser scan data of a human face to a model shape. Initial position (left), and final position (right) of the point data.

uniform scaling, i.e., a similarity. This is only a minor change in the presented algorithm, since the velocity vector field is still linear and just has one more real parameter σ,

$$\mathbf{v}(\mathbf{x}) = \sigma\mathbf{x} + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}. \tag{16}$$

In this application of face registration, many different face data sets might be registered with one fixed model shape. Here, it is especially appropriate to precompute the distance information of the model shape and store it in a hierarchical spatial data structure, that will be briefly discussed in the following. After this preprocessing step we have a computation time for the registration of less than one second for a (non-optimized) test implementation on a PC with 1.6 MHz. In general there are about 10–20 iterations necessary till convergence of our registration algorithm.

The spatial data structure mentioned above shall be called d^2-tree henceforth. The main idea is to decompose the surrounding space of the model shape into cubes that form an octree data structure, the d^2-tree. In each of the cube cells of the octree we store a quadratic approximant of the squared distance function d^2 of the model shape. One way to construct the d^2-tree in a top-down fashion is described in [12], but there is still enough room for improvement. In order to be memory efficient the data structure must be hierarchical, with smaller cells in the near field of the model surface and also in the areas of the cut locus of the model surface.

The d^2-tree allows fast registration for industrial inspection, because the necessary quadratic approximants of the given model shape can be quickly retrieved from that structure. For each sample point one just takes the quadratic approximant which is stored in the smallest cell containing the sample point. It is no longer necessary to compute footpoints of sample points in each iteration of the registration procedure.

In Fig. 6 a planar slice of the octree data structure of the face model shape is depicted. One can see that smaller cells are used near the surface and larger cells further away. For reasons of visualization, the depth of the octree has been limited.

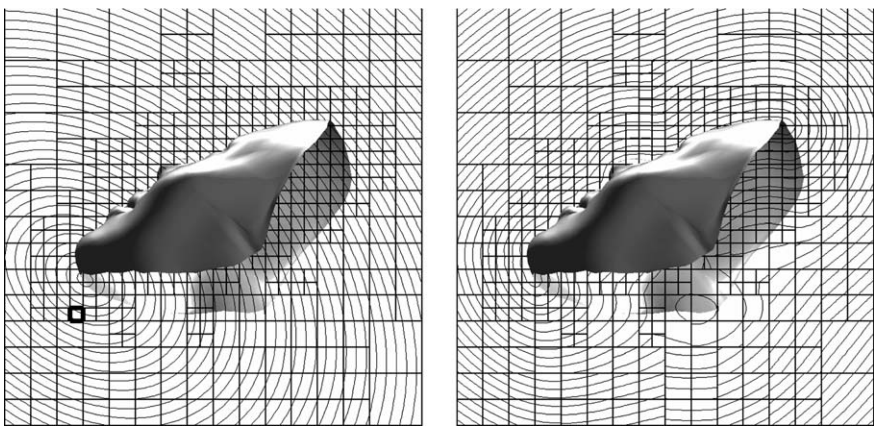

Fig. 6. Planar slices through the octree data structure d^2-tree. Level sets of *one* local quadratic approximant of d^2 (left), and level sets of *all* local quadratic approximants, restricted to their defining cells (right).

On the left of Fig. 6 one node cell of the octree has been marked. The local quadratic approximant stored in this marked cell is evaluated within the planar slice and is depicted by several of its level sets. The local quadratic approximant is a function defined on the whole surrounding space and is not restricted to the cell where it is stored. On the right of Fig. 6 all the local quadratic approximants are depicted but each of them is drawn only in its corresponding cell.

5. Convergence behavior

The examples given above provide some experimental evidence on the different convergence behavior of various registration algorithms. For a theoretical foundation of these results and a better understanding, it is necessary to study registration from the viewpoint of optimization. This has recently been done by the first author of the present paper [14]. We summarize here those results, which are important in the present context; proofs are given in [14].

The minimization of the objective function (10) is a constrained optimization problem, or more precisely, a constrained *nonlinear least squares problem* [8,11]. Throughout this discussion we denote the current and next position of the data point cloud by X_c and X_+, respectively. The final position (minimizer) is X^*. The individual points of these clouds are denoted by $\mathbf{x}_{i,c}, \mathbf{x}_{i,+}, \mathbf{x}_i^*$. The squared distance between two positions, say X_c and X^*, is defined as sum of squared distances of the corresponding data points,

$$\|X_c - X^*\|^2 := \sum_{i=1}^{N} \|\mathbf{x}_{i,c} - \mathbf{x}_i^*\|^2.$$

5.1. Convergence of ICP

The ICP algorithm of Besl and McKay turns out to be a kind of gradient descent and exhibits *linear convergence*: The distance of the iterates to the minimizer X^* decreases according to

$$\|X_+ - X^*\| \leqslant C\|X_c - X^*\|, \tag{17}$$

for some constant $C \in (0,1)$. The direction, from which the minimum is approached influences the constant C. Tangential moves of X along Φ give rise to a constant C close to 1, and thus to poor convergence.

5.2. Convergence of refined algorithms without rigidity constraint

A numerical algorithm relies on certain approximations of f and the constraints. In order to separate the effects caused by the approximation of f from the rigidity constraint, we first consider *affine registration*. This means that the displacement m of the data point cloud is not a rigid body motion, but an affine map. For affine registration, the displacement vector field $\mathbf{v}(\mathbf{x})$ is a general linear vector field, of the form

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + A \cdot \mathbf{x},$$

with a regular matrix A. It is used in Step 2 of our algorithm instead of the velocity field of a rigid body motion; Step 3 is not necessary.

In this way, one gets rid of the rigidity constraint. In fact, this simplification is well motivated if the model shape does not allow affine transformations in itself and if the deviation between data point cloud and model shape is close to zero (in optimization, one speaks of a small residual problem). Now, the actually desired minimum X^* with rigidity constraint is also an isolated minimum within the affine group. This means that one can remove the rigidity constraint; in practice one will consider rigidity only in the last iteration and make sure that the final position X^* results via rigid body motion of the initially given cloud X of measurement points.

For affine registration, we have the following convergence behavior:

- Registration based on squared tangent plane distances, which has been proposed already by Chen and Medioni [4], corresponds to a Gauss–Newton iteration, one of the most prominent optimization methods for the solution of nonlinear least squares problems [11]. For a *good initial position and a zero residual problem* (X fits exactly onto Φ), the algorithm has *quadratic convergence*, i.e., there is a constant K such that

$$\|X_+ - X^*\| \leqslant K\|X_c - X^*\|^2. \tag{18}$$

It is also well-known that the method works well for small residual problems and good initial positions. This corresponds very nicely to our experimental results.

- Using Taylor approximants of the squared distance function, the method is a *Newton algorithm*, and thus it *converges quadratically*, even for a larger residual (X does not fit well onto Φ). It is good that we use nonnegative quadratic approximants, because this avoids an indefinite Hessian of f, which could lead to divergence.

5.3. Convergence of refined algorithms with rigidity constraint

Let us now include the rigidity constraint. Then, the algorithms discussed above behave as follows:

- Linearization of the motion with help of a velocity field and projection onto the constraint manifold with help of a helical motion according to Section 4 does not affect the convergence behavior, if we have a *zero residual problem*. Thus, both squared tangent plane distances and more general local quadratic approximants of the squared distance function result in an algorithm with *quadratic convergence*. This also explains the good performance for small residual problems.
- For a larger residual problem, one has to modify the helical motion in Step 3 of each iteration. We propose to use the Armijo rule [11] to define a factor by which one reduces the rotation angle ϕ (and the translational part accordingly) if an iteration does not yield sufficient reduction in the value of the objective function f. This, however, gives just *linear convergence*.
- It is described in [14] how a second order motion approximant together with the squared distance field approximants of the present paper yield *quadratic*

convergence even for a problem with a larger residual. This requires just a few simple modifications of the algorithm proposed in the present paper.

6. Conclusion and future research

A geometric analysis of the ICP algorithm reveals the following fact: Using closest points on the model surface as corresponding points will rarely give fast convergence. This is so since the approximation of the squared distance function of the model shape with the help of squared distance functions of surface points does not work well in the vicinity of the surface. As an alternative, we provided a new framework for registration using better quadratic approximants of the squared distance function and instantaneous kinematics. Further work has to be done in order to satisfy the practical needs. Here is a list of some extensions.

- In extension of [17], we have to investigate other quadratic approximants of the squared distance function of a surface. In particular, we need an efficient way of computing local quadratic approximants if the model shape is just given as a point cloud. One way is to convert the point cloud into a triangulated manifold, see e.g., [21] for point sets from very noisy measurements. Another approach is working directly on the point cloud. The simplest way to accomplish this is to use a fast sweeping technique to compute the distance function d of the model shape [23,24] on a grid and then derive local quadratic approximants of d^2 from it. An alternative, on which we are currently working, propagates the entire required information (distance, canonical frame and principal curvatures at the closest point) with a sweeping method through the grid. This information shall be stored in a spatial hierarchical data structure, such that the necessary quadratic approximants can be quickly computed from that structure.
- Related to the previous item, it seems to be interesting to look at a hierarchical representation of the quadratic approximants of d^2, and use it efficiently in the various iteration steps of the registration procedure.
- The use of instantaneous kinematics allows us to extend the idea to the simultaneous registration of more than two geometric objects (partial scans); this has been outlined in [15], but requires further studies.

Acknowledgments

Part of this research has been carried out within the Competence Center *Advanced Computer Vision* and has been funded by the K*plus* program. This work was also supported by the Austrian Science Fund under grant P16002-N05 and by the innovative project "3D Technology" of Vienna University of Technology. H. Pottmann is grateful for support by the Institute of Mathematics and Its Applications at the University of Minnesota; main ideas of the present work could be developed during a stay at IMA in spring 2001.

References

[1] P.J. Besl, N.D. McKay, A method for registration of 3D shapes, IEEE Trans. Patt. Anal. Mach. Intell. 14 (1992) 239–256.

[2] P. Bourdet, A. Clément, Controlling a complex surface with a 3 axis measuring machine, Ann. CIRP 25 (1976) 359–361.

[3] P. Bourdet, A. Clement, A study of optimal-criteria identification based on the small-displacement screw model, Ann. CIRP 37 (1988) 503–506.

[4] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, Image Vis. Comput. 10 (1992) 145–155.

[5] D.W. Eggert, A.W. Fitzgibbon, R.B. Fisher, Simultaneous registration of multiple range views for use in reverse engineering of CAD models, Comput. Vis. Image Understand. 69 (1998) 253–272.

[6] D.W. Eggert, A. Larusso, R.B. Fisher, Estimating 3D rigid body transformations: a comparison of four major algorithms, Mach. Vis. Appl. 9 (1997) 272–290.

[7] O.D. Faugeras, M. Hebert, The representation. recognition, and locating of 3D objects, Int. J. Robotic Res. 5 (1986) 27–52.

[8] C. Geiger, C. Kanzow, Theorie und Numerik Restringierter Optimierungsaufgaben, Springer, Heidelberg, 2002.

[9] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, M. Levoy, Geometrically stable sampling for the ICP algorithm, Proc. 4th Int. Conf. 3D Imag. Model. (3DIM) (2003) 260–267.

[10] B.K.P. Horn, Closed form solution of absolute orientation using unit quaternions, J. Opt. Soc. A 4 (1987) 629–642.

[11] C.T. Kelley, Iterative Methods for Optimization, SIAM, Philadelphia, 1999.

[12] S. Leopoldseder, H. Pottmann, H.K. Zhao, The d^2-tree: a hierarchical representation of the squared distance function. Tech. Rep. 101 (2003). Institute of Geometry, Vienna University of Technology. Available from <http://www.geometrie.tuwien.ac.at/leopoldseder/t_rep101.pdf>.

[13] I.S. Okatani, K. Deguchi, A method for fine registration of multiple view range images considering the measurement error properties, Comput. Vis. Image Understanding 87 (2002) 66–77.

[14] H. Pottmann, Geometry and convergence analysis of registration algorithms, Tech. Rep. 117 (2004). Institute of Geometry, Vienna University of Technology. Available from <http://www.geometrie.tuwien.ac.at/ig/papers/tr117.pdf>.

[15] H. Pottmann, S. Leopoldseder, M. Hofer, Simultaneous registration of multiple views of a 3D object. Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXIV, Part 3A, Commission III, pp. 265–270.

[16] H. Pottmann, T. Randrup, Rotational and helical surface reconstruction for reverse engineering, Computing 60 (1998) 307–322.

[17] H. Pottmann, M. Hofer, Geometry of the squared distance function to curves and surfaces, in: H.-C. Hege, K. Polthier (Eds.), Visualization and Mathematics III, Springer, Heidelberg, 2003, pp. 221–242.

[18] H. Pottmann, J. Wallner, Computational Line Geometry, Springer-Verlag, Berlin, Heidelberg, New York, 2001.

[19] M. Rodrigues, R. Fisher, Y. Liu (Eds.), Special issue on registration and fusion of range images. Computer Vision and Image Understanding, vol 87, 2002, pp. 1–131.

[20] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: Proc. 3rd Internat. Conf. on 3D Digital Imaging and Modeling, Quebec, 2001.

[21] R. Sara, R. Bajcsy, Fish-scales: representing fuzzy manifolds, Proc. IEEE Conf. ICCV 98 (1998) 811–817.

[22] D. A. Simon, Fast and Accurate Shape-Based Registration. Ph.D. Thesis, Carnegie Mellon University, 1996.

[23] R. Tsai, Rapid and accurate computation of the distance function using grids, J. Comput. Phys. 178 (2002) 175–195.

[24] H.K. Zhao. A fast sweeping method for Eikonal equations. Mathematics of Computation, 73 (2004) to appear.