International Journal of Computer Vision 67(3), 277–296, 2006

DOI: 10.1007/s11263-006-5167-2

Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes

HELMUT POTTMANN
Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Austria
pottmann@geometrie.tuwien.ac.at

QI-XING HUANG, YONG-LIANG YANG AND SHI-MIN HU
Department of Computer Science and Technology, Tsinghua University, China
shimin@tsinghua.edu.cn

Received October 12, 2004; Revised May 24, 2005; Accepted August 31, 2005

First online version published in March, 2006

Abstract. The computation of a rigid body transformation which optimally aligns a set of measurement points with a surface and related registration problems are studied from the viewpoint of geometry and optimization. We provide a convergence analysis for widely used registration algorithms such as ICP, using either closest points (Besl and McKay, 1992) or tangent planes at closest points (Chen and Medioni, 1991) and for a recently developed approach based on quadratic approximants of the squared distance function (Pottmann et al., 2004). ICP based on closest points exhibits local linear convergence only. Its counterpart which minimizes squared distances to the tangent planes at closest points is a Gauss–Newton iteration; it achieves local quadratic convergence for a zero residual problem and—if enhanced by regularization and step size control—comes close to quadratic convergence in many realistic scenarios. Quadratically convergent algorithms are based on the approach in (Pottmann et al., 2004). The theoretical results are supported by a number of experiments; there, we also compare the algorithms with respect to global convergence behavior, stability and running time.

Keywords: registration, rigid registration, kinematics, optimization, ICP algorithm, distance function, convergence analysis

1. Introduction

Registration plays an important role in 3D model acquisition and geometry processing (Bernardini and Rushmeier, 2002). Individual overlapping scans of an object, initially available in different coordinate systems, have to be optimally positioned in a single system. This requires the simultaneous registration of a number of point clouds. Another industrial application of registration is the following: For the goal of shape inspection it is of interest to find the optimal Euclidean motion (translation and rotation) that aligns a cloud of measurement points of a workpiece to the CAD model from which it has been manufactured. This makes it possible to check the given workpiece for manufacturing errors and to visualize and classify the deviations. The latter registration problem concerns only two systems. It is basic to the entire family of rigid registration problems and thus we will investigate this problem in detail. The simultaneous registration of more than two systems can be done along similar lines, but its description requires more care on issues

such as choice of the initial position, automatic detection of the overlapping regions of different scans, etc., and thus we will describe it in a separate paper.

Previous Work. A well-known standard algorithm to solve the present registration problem is the *iterative closest point (ICP) algorithm* of Besl and McKay (1992). Independently, Chen and Medioni (1991) proposed a similar algorithm. Although these two algorithms are based on similar ideas, we will see later that the difference—from the viewpoint of optimization—is not marginal at all. Most of the literature is based on these algorithms and deals with a variety of possible improvements. An excellent summary with new results on the acceleration of the ICP algorithm has been given by Rusinkiewicz and Levoy (2001), who also suggest that *iterative corresponding point* is a better expansion for the abbreviation ICP than the original *iterative closest point*. Among the many improvements of ICP we point to a paper by Sharp et al. (2002), where correspondences are determined in a hybrid space of point and feature coordinates. This leads to a remarkable gain on the side of global convergence. Planitz et al. (2005) proposed a general framework for analyzing, comparing, developing and implementing surface correspondence algorithms. Robust point set registration using a statistical rather than geometric approach has been proposed by Tsin and Kanade (2004). For an overview of further recent literature on registration we also refer to (Eggert et al. (1998); Gelfand et al. (2003); Ikemoto et al. (2003); Mian et al. (2004); Rodrigues et al. (2002) and the references therein.

A study of the geometry of the squared distance function of a geometric object led to the formulation of another type of registration algorithms (Pottmann et al., 2004), where one system (scan) 'flows' within the squared distance field of the other scan towards the latter. The present paper has been motivated by that approach.

Contributions and Outline of the Present Paper. Despite the large amount of work on registration, it seems that there is no thorough investigation of registration algorithms from the viewpoint of geometry and optimization. Filling this gap is the main purpose of the present contribution. The study of registration as a geometric optimization problem reveals important information on the behavior of known algorithms and provides the theoretical basis for empirical results which have been reported in earlier papers. Moreover, we will present an analysis and some extensions of the work in (Pottmann et al., 2004), which aim at registration algorithms with quadratic convergence.

This paper is organized as follows. Section 2 summarizes basic facts from kinematics and the geometry of the distance function to a surface. In Section 3, registration is formulated as a constrained nonlinear least squares problem. Gradients of the objective function in various norms and corresponding gradient descent schemes for registration are studied. These algorithms are of importance for the global convergence behavior; the local convergence is just linear. Registration with the ICP algorithm is discussed in Section 4. It is shown that ICP exhibits *linear convergence*. To obtain better local convergence, we devise and analyze in Section 5 algorithms with quadratic convergence; such algorithms are of the Newton type and require second order approximants of the objective function. Simplified versions are Gauss–Newton iteration and the Levenberg–Marquart method. These are addressed in Section 5.4. In fact, Gauss–Newton iteration turns out to be the algorithm of Chen and Medioni. Section 6 contains the experimental validation of the theory and a comparison of the various algorithms with respect to global convergence behavior, stability and running time.

2. Kinematical Geometry and the Squared Distance Function of a Surface

First Order Properties of One-Parameter Motions. Since registration requires the computation of an optimal rigid body motion, kinematical geometry plays an important role. We review here some basic facts; for proofs and more details we refer to the literature, e.g. (Bottema and Roth, 1990; Pottmann and Wallner, 2001). Consider a rigid body moving in Euclidean three-space $\mathbb{R}^3$. We think of two copies of $\mathbb{R}^3$: One copy associated with the moving body and called *moving system* Σ^0, and one copy called the *fixed system* Σ. We use Cartesian coordinates and denote points of the moving system Σ^0 by $\mathbf{x}^0, \mathbf{y}^0, \ldots,$ and points of the fixed system by $\mathbf{x}, \mathbf{y}$, and so on.

A *one-parameter motion* Σ^0/Σ is a smooth family of Euclidean congruence transformations depending on a parameter t which can be thought of as time. A point $\mathbf{x}^0$ of Σ^0 is, at time t, mapped to the point $\mathbf{x}(t) = A(t) \cdot \mathbf{x}^0 + \mathbf{a}_0(t)$ of Σ. All points of Σ^0 have a *path curve* $\mathbf{x}(t)$ in Σ. The path of the origin is $\mathbf{a}_0(t)$. $A(t)$

describes the rotational part; we have $A^T = A^{-1}$ and $\det(A) = 1$.

The first derivative $\dot{\mathbf{x}}(t) = \dot{A}(t) \cdot \mathbf{x}^0 + \dot{\mathbf{a}}(t)$ of the path of $\mathbf{x}^0$ is its *velocity vector* at time t. We write $\mathbf{v}(\mathbf{x})$ for the vector field of vectors $\dot{\mathbf{x}}(t)$ attached to the points $\mathbf{x}(t)$. It is well-known that the vector field $\mathbf{v}(\mathbf{x})$ is linear and has the special form

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}. \tag{1}$$

Of special interest are the *uniform* motions, whose velocity vector field is constant over time. Apart from the trivial uniform motion, where all velocities are zero, uniform motions are either *translations* ($\mathbf{c} = \mathbf{o}$, $\bar{\mathbf{c}} \neq \mathbf{o}$), *rotations about a fixed axis* ($\mathbf{c} \cdot \bar{\mathbf{c}} = 0$, $\mathbf{c} \neq \mathbf{o}$) or *helical motions* ($\mathbf{c} \cdot \bar{\mathbf{c}} \neq 0$). The latter are the superposition of a uniform rotation and a uniform translation parallel to the rotation's axis. If ω is the angular velocity of the rotation, and v the velocity of the translation, then $p = v/\omega$ is called the *pitch* of the helical motion. Up to the first differentiation order, any one-parameter motion agrees locally with one of these motions.

If $(\mathbf{c}, \bar{\mathbf{c}})$ represents the velocity vector field of the motion, then the Plücker coordinates $(\mathbf{g}, \bar{\mathbf{g}})$ of the axis, the angular velocity ω and the pitch p of the instantaneous helical motion (including special cases) are reconstructed by

$$p = (\mathbf{c} \cdot \bar{\mathbf{c}})/\mathbf{c}^2, \quad \omega = \|\mathbf{c}\|, \quad (\mathbf{g}, \bar{\mathbf{g}}) = (\mathbf{c}, \bar{\mathbf{c}} - p\mathbf{c}). \tag{2}$$

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

Second Order Taylor Approximant of Uniform Motions. So far we have seen that a first order approximation of a motion at a given position, say at time $t = 0$, is given by $\mathbf{x}_1(t) = \mathbf{x}(0) + t\dot{\mathbf{x}}(0) = \mathbf{x}_0 + t(\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0)$. Here, $\mathbf{x}_0$ is the position of $\mathbf{x}^0 \in \Sigma^0$ at $t = 0$ in Σ. A *second order approximant* of a uniform motion is (cf. (Bottema and Roth, 1990)),

$$\mathbf{x}_2(t) = \mathbf{x}_0 + t(\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0) + \frac{t^2}{2}\mathbf{c} \times (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0). \tag{3}$$

We will use (3) as a *local parameterization of the Euclidean motion group*, which is precise up to second order. There, it is sufficient to identify $(t\mathbf{c}, t\bar{\mathbf{c}})$ with $(\mathbf{c}, \bar{\mathbf{c}})$ and use the following parameterization with six scalar parameters $(\mathbf{c}, \bar{\mathbf{c}})$,

$$\begin{aligned}\mathbf{x}(\mathbf{c}, \bar{\mathbf{c}}) &= \mathbf{x}_0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0 + \frac{1}{2}\mathbf{c} \times (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0) \\ &= \mathbf{x}_0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0 + \frac{1}{2}[\mathbf{c} \times \bar{\mathbf{c}} + (\mathbf{c} \cdot \mathbf{x}_0)\mathbf{c} \\ &\quad - \mathbf{c}^2\mathbf{x}_0].\end{aligned} \tag{4}$$

Computing a Displacement from a Taylor Approximant. The first or second order approximations of uniform motions discussed above are in general not rigid body transformations. Later, it will be necessary to actually perform the rigid body transformation, whose first or second order approximant is known. In other words, we also have to add the higher order terms in the Taylor expansion. Fortunately, this turns out as a very simple task: In the unlikely case that there is no rotational part, i.e., $\mathbf{c} = 0$, we are done, since then we have a translation with the vector $\bar{\mathbf{c}}$, which of course is a rigid body motion. Otherwise we note that the velocity field of the instantaneous motion is uniquely associated with a uniform helical motion. Its axis A and pitch p can be computed with formula (2). The rotational angle is given by $\phi = \|\mathbf{c}\|$. Altogether, the desired motion is the superposition of a rotation about the axis A through an angle of $\phi = \|\mathbf{c}\|$ and a translation parallel to A by the distance of $p \cdot \phi$. For the explicit formulae we refer to the literature (Bottema and Roth, 1990; Pottmann and Wallner, 2001).

Euclidean Motions Embedded in the Affine Group. If we do not impose orthogonality on the matrix A, we get, for each t, an *affine map*. Viewing rigid body transformations as special affine maps will be very useful for the planned analysis of registration algorithms.

In the following, we use a kinematic mapping (see (Hofer et al., 2004)) that views affine maps as points in 12-dimensional affine space. For that, consider the affine map $\mathbf{x} = \alpha(\mathbf{x}^0) = \mathbf{a}_0 + A \cdot \mathbf{x}^0$. Let us denote the three column vectors of A as $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$. They describe the images of the basis vectors of Σ^0 in Σ. Of course, we have $\mathbf{x} = \mathbf{a}_0 + x_1^0\mathbf{a}_1 + x_2^0\mathbf{a}_2 + x_3^0\mathbf{a}_3$. Now we associate with the affine map α a *point in 12-dimensional affine space* $\mathbb{R}^{12}$, represented by the vector $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$. The images of Euclidean congruence transformations (rigid body motions) $\alpha \in SE(3)$ form a 6-dimensional manifold $M^6 \subset \mathbb{R}^{12}$. Its six equations are given by the

orthogonality conditions of A, i.e., $\mathbf{a}_i \cdot \mathbf{a}_j = \delta_{ij},\ i, j = 1, 2, 3$.

It will be necessary to introduce a meaningful *metric* in $\mathbb{R}^{12}$. Following (Hofer et al., 2004), this is done with help of a collection X of points $\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_N^0$ in the moving system (body), which shall be called *feature points*. The squared distance between two affine maps α and β is now defined as sum of squared distances of feature point positions after application of α and β, respectively,

$$\|\alpha - \beta\|^2 = \|\mathsf{A} - \mathsf{B}\|^2 := \sum_i [\alpha(\mathbf{x}_i^0) - \beta(\mathbf{x}_i^0)]^2. \quad (5)$$

With $\mathsf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$, $\mathsf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_3)$, $\mathsf{C} := \mathsf{A} - \mathsf{B} = (\mathbf{c}_0, \ldots, \mathbf{c}_3)$, and $\mathbf{x}_i^0 = (x_{i,1}^0, x_{i,2}^0, x_{i,3}^0)$ the distance becomes

$$\|\mathsf{A} - \mathsf{B}\|^2 = \|\mathsf{C}\|^2 = \sum_i [\mathbf{c}_0 + x_{i,1}^0 \mathbf{c}_1 + x_{i,2}^0 \mathbf{c}_2 + x_{i,3}^0 \mathbf{c}_3]^2 =: \mathsf{C}^T \cdot M \cdot \mathsf{C}. \quad (6)$$

This expression with help of a positive definite symmetric matrix M shows that the metric (5) in the space of affine maps is Euclidean. It only depends on the barycenter $\mathbf{s}_x = (1/N) \sum_i \mathbf{x}_i^0$ and on the inertia tensor $J := \sum_i \mathbf{x}_i^0 \cdot \mathbf{x}_i^{0T}$ of the set of feature points $\mathbf{x}_i^0$ in the moving system (Hofer et al., 2004).

The Squared Distance Function of a Surface. Here we will summarize a few basic facts on the squared distance function. For more details, we refer to (Pottmann and Hofer, 2003). Given a surface $\Phi \subset \mathbb{R}^3$, the squared distance function d^2 assigns to each point $\mathbf{x} \in \mathbb{R}^3$ the square of its shortest distance to Φ. The importance of this function for our algorithms lies in the fact that we want to compute a position of a data point cloud which minimizes the sum of squared distances to a given surface. Since several important optimization concepts require second order approximants of the objective function, we need second order approximants of d^2.

Let us fix the notation. We consider a surface Φ with unit normal vector field $\mathbf{n}(\mathbf{s}) = \mathbf{n}_3(\mathbf{s})$, attached to its points $\mathbf{s}$. At each point $\mathbf{s} \in \Phi$, we have a local Cartesian frame $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n})$, whose first two vectors $\mathbf{n}_1, \mathbf{n}_2$ determine the principal curvature directions. We will refer to this local frame as *principal frame* $\Pi(\mathbf{s})$. Let κ_i be the (signed) principal curvature to the principal curvature direction $\mathbf{n}_i$, $i = 1, 2$, and let $\rho_i = 1/\kappa_i$.

It is known that the second order Taylor approximant F_d of the function d^2 at a point $\mathbf{p} \in \mathbb{R}^3$ is expressed in the principal frame at $\mathbf{p}$'s closest point (normal foot point) $\mathbf{s} \in \Phi$ as

$$F_d(\mathbf{x}) = \frac{d}{d - \rho_1}(\mathbf{n}_1 \cdot \mathbf{x} + h_1)^2 + \frac{d}{d - \rho_2}(\mathbf{n}_2 \cdot \mathbf{x} + h_2)^2 + (\mathbf{n}_3 \cdot \mathbf{x} + h_3)^2. \quad (7)$$

Here, $\mathbf{n}_i \cdot \mathbf{x} + h_i = 0$, $i = 1, 2, 3$, are the equations of principal planes and tangent plane at $\mathbf{s}$, respectively.

In the important special case $d = 0$ (i.e., $\mathbf{p} = \mathbf{s}$), the approximant F_d equals the squared distance function to the tangent plane of Φ at $\mathbf{s}$. Thus, if $\mathbf{p}$ is close to Φ, the squared distance function to the tangent plane at $\mathbf{p}$'s closest point on Φ is a good approximant of d^2.

We may have an indefinite Taylor approximant, which might be undesirable for optimization. Then, we derive *nonnegative quadratic approximants* either by replacing a negative term $d/(d - \rho_j)$ by zero or by $|d|/(|d| + |\rho_j|)$; a motivation for the latter choice is given in (Pottmann and Hofer, 2003). In any case, a second order approximant F_d is with appropriate coefficients α_1, α_2 and $\alpha_3 = 1$ given by

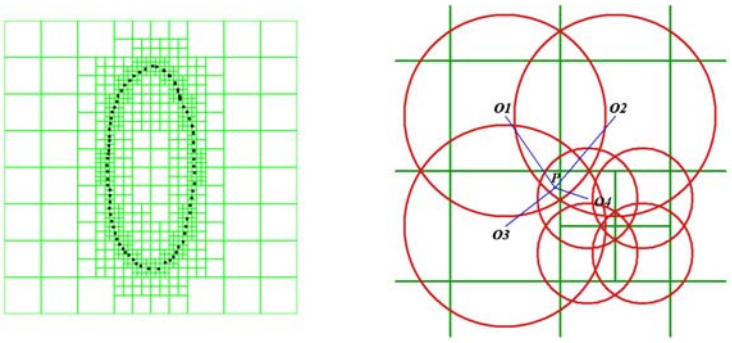
$$F_d(\mathbf{x}) = \sum_{j=1}^{3} \alpha_j (\mathbf{n}_j \cdot \mathbf{x} + h_j)^2. \quad (8)$$

Note that so far we tacitly assumed that $\mathbf{p}$ does not lie on the *cut locus* of Φ. There the distance function d and also its square are not differentiable, and it makes no sense to talk about a second order Taylor approximant.

For later use we finally note that the *gradient* ∇d^2 of the squared distance function at a given point $\mathbf{p}$ is

$$\nabla d^2 = 2(\mathbf{p} - \mathbf{s}) = 2d\,\mathbf{n}. \quad (9)$$

Remark 1. For the sake of brevity, we are discussing in this paper only the case of smooth surfaces. However, the change to the more practical case of piecewise smooth surfaces is straightforward. Such a surface exhibits sharp edges and vertices (intersection points of edges, singular points such as the vertex of a cone); its squared distance field is composed of squared distance fields of smooth surfaces, of curves (edges and eventual boundary curves) and of points (vertices). The squared distance field of a point is quadratic anyway. Quadratic approximants to squared distance fields of space curves have been studied in (Pottmann and Hofer, 2003). Intuitively, the simplicity of the extension to piecewise smooth objects is explained as follows: we attach small



(a) cells in a d2tree for two dimensional PCD

(b) Sketch to the modified d2tree in 2D; cells are bounded by circles

Figure 1. Modified d2tree structure in 2D.

smooth blending surfaces along edges and corners, with blending radius ε, and consider the limit for $\varepsilon \to 0$.

A Data Structure for Fast Distance Information Retrieval. Surface patches obtained from a 3D scanning device are usually defined by point cloud data (PCD) which do not contain any topology information. For the purpose of registration of PCD, one can still compute second order quadratic approximants in each iteration. A numerically stable algorithm in this case benefits from a globally smooth fitting surface and then computes closest points together with the curvature information at these points. This is not easy and time consuming. Therefore, we briefly address here a modified d2tree method (Leopoldseder et al., 2003) for computing quadratic approximants to the squared distance function. It involves least squares fitting of quadratic patches. The pre-computed quadratic patches are stored in a special data structure called d2tree. Figure 1(a) shows a d2tree for simple two dimensional ellipse-like PCD.

Simply put, the d2tree in 3D is an octree like data structure each cell of which stores a quadratic function that approximates the squared distance locally. Previous structures of d2tree compute these quadratic functions by least squares fitting to the squared distance function with the same error threshold. However, as different cells correspond to different approximants, these constructions can not preserve the continuity of quadratic approximants along the boundary of each cell. The modified d2tree structure solves this problem by borrowing the idea of 'partition of unity' (Ohtake et al., 2003), which is typically used to integrate locally defined approximants into a global approximation. In our approach, the quadratic patch in each cell C_i is associated with a C^2 compactly supported function $w_i(\cdot)$,

$$w_i(\mathbf{x}) = W\left(\frac{\|\mathbf{x} - \mathbf{o}_i\|^2}{\theta\, d_i^2}\right). \tag{10}$$

Here, $\mathbf{o}_i$ and d_i are respectively the center and the length of the diagonal of cell C_i, and $W(\cdot)$ is a C^2 function with support interval $[0, 1]$; the constant θ controls the overlap of the cells. In our implementation, we chose W to be a cubic B-spline basis function and $\theta = 1.7$.

The squared distance approximant at a point $\mathbf{x}$ is defined by blending of the squared distance approximants of its adjacent cells,

$$F_+(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x})(\mathbf{x}^T \cdot A_i \cdot \mathbf{x} + 2\mathbf{b}_i \cdot \mathbf{x} + c_i)}{\sum_i w_i(\mathbf{x})}. \tag{11}$$

The summation in Eq. (11) is taken over all cells. However, as all $w_i(\cdot)$ are compactly supported, for a fixed point x_0, only a few terms in (11) contribute to its squared distance approximation so that it can be efficiently computed. Figure 1(b) shows a sketch in 2D, where the squared distance approximation of $\mathbf{p}$ is a weighted combination of the squared distance approximations in cells with centers $\mathbf{o}_i$, $1 \le i \le 4$.

The construction of the modified d2tree is done in a top-down style based on fitting quadratic functions $F(\cdot)$ to samples of the squared distance field of the PCD. The details of the construction is similar to the method used in (Ohtake et al., 2003; Leopoldseder et al., 2003) and will not be described here. For our construction, the number of levels of the tree and the error threshold for the quadratic approximants are the required parameters.

Our application requires a squared distance approximant near a given point $\mathbf{p}$. Unlike the d2tree defined before, we use the second order Taylor approximant of $F_+(\mathbf{x})$ at $\mathbf{p}$,

$$F_2(\mathbf{x}) = F_+(\mathbf{p}) + \nabla F_+(\mathbf{p})^T \cdot (\mathbf{x} - \mathbf{p}) + \frac{1}{2}(\mathbf{x} - \mathbf{p})^T \cdot \nabla^2 F_+(\mathbf{p}) \cdot (\mathbf{x} - \mathbf{p}).$$

As $W(\cdot)$ has an analytic expression, both $\nabla F_+(\cdot)$ and $\nabla^2 F_+(\cdot)$ can be computed analytically.

Remark 2. Compared with the previous d2tree (Leopoldseder et al., 2003), the modified d2tree structure takes more time to supply the squared distance approximant at a given point, as it needs the computation of gradient and Hessian of $F_+(\cdot)$. To cut down the computation time in the present application, one can just apply the modified strategy when the cell is near the surface. However, the time needed for computing gradient and Hessian remains small compared to the time which would be necessary for computing closest points in each iteration of the following registration algorithms.

3. Problem Formulation and Gradient Descent

A set of points $X^0 = (\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots)$ is given in some coordinate system Σ^0. It shall be rigidly moved (registered, positioned) to be in best alignment with a given surface Φ, represented in system Σ. We view Σ^0 and Σ as moving and fixed system, respectively. A position of X^0 in Σ is denoted by $X = (\mathbf{x}_1, \ldots)$. It is the image of X^0 under some rigid body motion α. Since we identify positions with motions, the motions have to act on the same initial position. Thus, we always write $X = \alpha(X^0)$.

The point set X^0 may be a cloud of measurement points on the surface of a 3D object. The surface Φ may be the corresponding CAD model, another scan of the same object, a scan of a similar object, a mean shape in some class of shapes, etc. For our description, we will simply speak of a data point cloud and a surface Φ ('model shape'), but have in mind that Φ may also be given just as a point cloud. For computations with point cloud data, we refer to (Mitra et al., 2004).

The *registration problem* shall be formulated in a least squares sense as follows. Compute the rigid body transformation α^*, which minimizes

$$F(\alpha) = \sum_i d^2(\alpha(\mathbf{x}_i^0), \Phi). \tag{12}$$

Here, $d^2(\alpha(\mathbf{x}_i^0), \Phi)$ denotes the squared distance of $\alpha(\mathbf{x}_i^0)$ to Φ. If we view α as a special affine map, we have to compute its 12 parameters $(\mathbf{a}, A)$ under the constraint that A is an orthogonal matrix. Hence, the present problem is a *constrained nonlinear least squares problem* (Geiger and Kanzow, 2002; Fletcher, 1987; Kelley, 1999).

The following notation will be used throughout this paper. The current position of the data point cloud in some iterative procedure is called $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots) = \alpha(X^0)$; if necessary, we write more precisely $X_c = (\mathbf{x}_{1c}, \ldots) = \alpha_c(X^0)$. The next position in an iteration is indicated by $X_+ = (\mathbf{x}_{1+}, \ldots) = \alpha_+(X^0)$. The minimizer of F is $X^* = \alpha^*(X^0)$.

For practical reasons, in particular for dealing with outliers in the data set X, one may use a weighted sum. This is not a major difference and shall be neglected in the following.

3.1. Gradient Descent

In the following, we compute the gradient of the objective function F in (12). In view of (9), we multiply F by the factor 1/2, but call the function again F. A tangential direction in the Euclidean motion group is determined by an instantaneous velocity vector field $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. Let $\mathbf{y}_i$ be the closest points of the current data point positions $\mathbf{x}_i$ on Φ, and set $\mathbf{f}_i := \mathbf{x}_i - \mathbf{y}_i$. Then, by Eq. (9), the *directional derivative* of F in direction $\mathsf{C} = (\mathbf{c}, \bar{\mathbf{c}})$ reads

$$\frac{\partial F}{\partial \mathsf{C}} = \sum_i (\mathbf{x}_i - \mathbf{y}_i) \cdot \mathbf{v}(\mathbf{x}_i) = \sum_i \mathbf{f}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) = \sum_i (\mathbf{f}_i \cdot \bar{\mathbf{c}} + \bar{\mathbf{f}}_i \cdot \mathbf{c}).$$

Here $\bar{\mathbf{f}}_i = \mathbf{x}_i \times \mathbf{f}_i$ is the momentum vector of the surface normal through $\mathbf{x}_i$. Let us view the vectors $\mathbf{f}_i$ as forces acting along the corresponding surface normals. We call these forces the *repelling forces*. Then, $\bar{\mathbf{f}}_i$ are the moments of these forces. Altogether, we have a *repelling force system*, represented in terms of screw theory (Pottmann and Wallner, 2001), pp. 192) by the

screw

$$\mathsf{F} = (\mathbf{f}, \bar{\mathbf{f}}) = \left(\sum_i \mathbf{f}_i, \sum_i \bar{\mathbf{f}}_i \right). \tag{13}$$

We will call F the *repelling screw* and $-\mathsf{F}$ the *attracting screw*. Hence, the directional derivative appears as *virtual work done by the repelling force system on the instantaneously moving data shape*,

$$\frac{\partial F}{\partial \mathsf{C}} = \mathbf{f} \cdot \bar{\mathbf{c}} + \bar{\mathbf{f}} \cdot \mathbf{c}. \tag{14}$$

With known results from line geometry and screw theory (Pottmann and Wallner, 2001) we conclude: *An instantaneous motion with directional derivative zero corresponds to a screw which is reciprocal to* F. *In particular, the axes of instantaneous rotations, which yield vanishing directional derivative of F, lie in a linear line complex.*

A minimizer is characterized by vanishing derivative in all directions. This is only possible if the screw F vanishes. In terms of statics, the condition may be expressed as follows:

Proposition 3. *At a position, which is a local minimizer of the objective function F of the registration problem, the repelling force system (or equivalently the attracting force system) is in equilibrium.*

Remark 4. In the case of known correspondences, we have an analogous equilibrium property of the force system $\mathsf{F} = (\mathbf{f}, \bar{\mathbf{f}})$ defined by the vectors $\mathbf{x}_i - \mathbf{y}_i$ to pairs of corresponding points. In particular, this requires $\mathbf{f} = 0$, which expresses exactly the well-known correspondence of the barycenters of the two point sets X and Y see (Faugeras and Hebert, 1986; Horn, 1987).

To compute the gradient, we need a metric, since the direction C needs to be normalized. The simplest normalization via $\mathbf{c}^2 + \bar{\mathbf{c}}^2 = 1$ yields as gradient

$$\nabla F = (\bar{\mathbf{f}}, \mathbf{f}) =: \mathsf{F}^*. \tag{15}$$

It is more natural, however, to use the Euclidean metric (5) for normalization of the tangent vector to M^6, represented by C. This requires that we normalize according to $\sum_i (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2 = 1$. This normalization can be written as

$$\mathsf{C}^T \cdot M_e \cdot \mathsf{C} = 1. \tag{16}$$

Writing the directional derivative in the form $\partial F / \partial \mathsf{C} = \mathsf{F}^{*T} \cdot M_e^{-1} \cdot M_e \cdot \mathsf{C}$, we see that the gradient $\nabla_e F$ of F for the normalization via the metric (5) is

$$\nabla_e F = M_e^{-1} \cdot \nabla F = M_e^{-1} \cdot \mathsf{F}^*. \tag{17}$$

Both $-\nabla F$ and $-\nabla_e F$ are in a certain metric directions of steepest descent and can be employed in a *gradient descent algorithm*. One computes X_+ from X_c by application of a 'small' displacement, which is in first order given by the velocity field in direction of the steepest descent. One considers the helical motion defined by this velocity field $(\mathbf{c}, \bar{\mathbf{c}})$. Then, one applies to the current position X_c the helical motion according to Section 2, with an appropriate rotational angle ϕ. One can start with $\phi = \|\mathbf{c}\|$ and then check the validity of the corresponding step. If the decrease of the objective function is not sufficient, the rotational angle is reduced according to the Armijo rule (Kelley, 1999) or a more sophisticated step size prediction scheme of optimization (Fletcher, 1987; Kelley, 1999).

Remark 5. The gradient according to (17) possesses the following interpretation. We are looking for a velocity vector field $\mathbf{v}(\mathbf{x})$, determined by $\mathsf{C} = (\mathbf{c}, \bar{\mathbf{c}})$, such that the first order approximants of the displaced data points, namely the points $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i) = \mathbf{x}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i$, are as close as possible to the closest points $\mathbf{y}_i \in \Phi$ of $\mathbf{x}_i$, in a least squares sense. This requires the minimization of

$$F_1 = \sum (\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i) - \mathbf{y}_i)^2 = \sum_i (\mathbf{f}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2. \tag{18}$$

With the expression of $\sum_i (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2 = 1$ in the form (16), and with help of (15) and (13), function F_1 reads in matrix notation

$$F_1 = \mathsf{C}^T \cdot M_e \cdot \mathsf{C} + 2(\mathsf{F}^*)^T \cdot \mathsf{C} + \sum_i \mathbf{f}_i^2. \tag{19}$$

Therefore, the minimizer C_m is given by the negative gradient from Eq. (17),

$$\mathsf{C}_m = -M_e^{-1} \cdot \mathsf{F}^* = -\nabla_e F. \tag{20}$$

Thus, a gradient descent based on $\nabla_e F$ tries in each iteration to bring the new data points $\mathbf{x}_{i+}$ as close as possible to the foot points $\mathbf{y}_i$ of the current data points

$\mathbf{x}_{ic}$. This is similar to the ICP algorithm, which is discussed in more detail in Section 4. There, we show that ICP is linearly convergent. The same holds for a gradient descent, if one uses an appropriate step size (Kelley, 1999).

Although gradient descent is not a good method for the fine positioning, it may be very useful to reach the convergence area of an algorithm with quadratic convergence, described in Section 5.

4. The ICP Algorithm Revisited

The most widely used algorithm for the solution of the registration problem is the *iterative closest point (ICP) algorithm* of Besl and McKay (1992). We will briefly describe this algorithm and then take another point of view which immediately reveals its convergence properties.

The ICP algorithm performs in each iteration the following two steps.

(1) For each point $\mathbf{x}_i = \alpha(\mathbf{x}_i^0)$ in the current position of the data shape, the closest point $\mathbf{y}_i$ in the model shape is computed. This is the most time consuming part of the algorithm and can be implemented efficiently, e.g. by using an octree data structure. As 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.
(2) The rigid motion α_+ is computed such that the moved data points $\mathbf{x}_{i+} = \alpha_+(\mathbf{x}_i^0)$ are closest to their corresponding points $\mathbf{y}_i$, where the objective function to be minimized is

$$F_1 = \sum_i \|\mathbf{x}_{i+} - \mathbf{y}_i\|^2. \tag{21}$$

This least squares problem can be solved explicitly. The translational part of α_+ brings the barycenter $\mathbf{s}_x^0$ of X^0 to the barycenter $\mathbf{s}_y$ of Y (cf. Remark 4). The rotational part of α_+ can be obtained as the unit eigenvector that corresponds to the maximum eigenvalue of a certain symmetric 4×4 matrix (Faugeras and Hebert, 1986; Horn, 1987). The solution eigenvector is nothing but the unit quaternion description of the rotational part of α_+.

Now steps 1 and 2 are repeated, always using the updated data points, until the change in the mean-square error falls below a preset threshold. The ICP algorithm always converges monotonically to a local minimum, since the value of the objective function is decreasing in each iteration.

4.1. ICP Exhibits Linear Convergence

The ICP algorithm can be understood nicely if we embed the set of rigid body motions into the space D of continuous deformations. A distance measure in D can be introduced similarly as in $\mathbb{R}^{12}$, say with help of the measurement points in X. Clearly, this distance cannot distinguish between deformations that act identically on X. We could also restrict to special deformations that are uniquely determined by an image set Y of X and reproduce Euclidean congruences where possible.

The set of Euclidean congruences is some 6-dimensional manifold C^6 in D. The set of deformations α which map X onto points of Φ, i.e. $F(\alpha) = 0$, is some manifold $D_0 \subset D$. In case that there are no measurement errors and X fits exactly to Φ, a solution α^* of the registration problem is an intersection point of D_0 and C^6.

The two steps of ICP are interpreted in D as follows.

(1) To the point $\alpha_c \in C^6$ (representing the motion between initial and current position X_c of the data point cloud X), compute the closest point $\alpha_f \in D_0$ (the deformation towards the cloud of closest points on Φ).
(2) To $\alpha_f \in D_0$, compute the closest point $\alpha_+ \in C^6$.

Hence, each iteration consists of two orthogonal projections with respect to the chosen metric in D. At first, one projects from a point on C^6 orthogonally onto D_0, and then orthogonally back to C^6. We will show that this kind of double projection *converges linearly*. In case of a precise fit between data and model shape, we have convergence to an intersection point of D_0 and C^6. If there exists a deviation between data shape X and model shape, we have convergence towards a common normal of D_0 and C^6. In both cases, the algorithm converges to a minimizer of the objective function F. Depending on the initial position, this may just be a local minimizer, but not the global one.

Linear convergence means that the distance of the iterates to the solution α^* decreases according to

$$\|\alpha_+ - \alpha^*\| \leq C\|\alpha_c - \alpha^*\|, \tag{22}$$

for some constant $C \in (0, 1)$.

Let us now proceed with a proof of the error formula (22). In particular, we would like to compute the constant C, which determines the speed of convergence.

For our purposes it is sufficient to make the following simplification. We consider the sequence of iterates $(\ldots, \alpha_c, \alpha_+, \ldots)$ in C^6 as points of some curve $\mathbf{c} \subset C^6$. The intermediate foot points α_f lie in some curve $\mathbf{f} \subset D_0$. Each tangent of the curve $\mathbf{c}$ lies in the corresponding tangent space of C^6; hence a normal onto C^6 is also a normal onto $\mathbf{c}$. The same holds for the curve $\mathbf{f}$. The desired common normal of D_0 and C^6 is also a common normal of these two curves; the normal foot points shall be $\mathbf{c}^*$ $(= \alpha^*)$ and $\mathbf{f}^*$. Of course, in case of an intersection point we have $\mathbf{c}^* = \mathbf{f}^*$. Therefore, we consider the double projection algorithm for the computation of the common normal of two curves $\mathbf{c}$ and $\mathbf{f}$. It is sufficient to assume finite dimension m of the embedding space $D = \mathbb{R}^m$; since only the second order Taylor expansions of $\mathbf{c}$ and $\mathbf{f}$ around $\mathbf{c}^*$ and $\mathbf{f}^*$ enter the discussion, dimension $m = 5$ is actually sufficient. Moreover, it suffices to express orthogonality in $\mathbb{R}^m$ with help of the canonical inner product.

We consider arc length parameterizations $\mathbf{c}(u)$ and $\mathbf{f}(v)$ for the two curves, with $\mathbf{c}(0) = \mathbf{c}^*$, $\mathbf{f}(0) = \mathbf{f}^*$. Assuming bounded derivatives up to third order, the Taylor expansions read

$$\mathbf{c}(u) = \mathbf{c}^* + u\mathbf{c}_0' + \frac{u^2}{2}\mathbf{c}_0'' + O(u^3),$$
$$\mathbf{f}(v) = \mathbf{f}^* + v\mathbf{f}_0' + \frac{v^2}{2}\mathbf{f}_0'' + O(v^3).$$

The common normal property of $\mathbf{c}^*$, $\mathbf{f}^*$ is expressed as

$$(\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{c}_0' = 0, \quad (\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{f}_0{}' = 0. \tag{23}$$

Given a current position $\mathbf{c}(u_c)\mathbf{f}(u_c)$ for the common normal, which is orthogonal to $\mathbf{f}$ at $\mathbf{f}(u_c)$, the next position $\mathbf{c}(u_+)$, $\mathbf{f}(u_c)$ is orthogonal to $\mathbf{c}$. This is formulated in the equations

$$(\mathbf{c}(u_c) - \mathbf{f}(v_c)) \cdot \mathbf{f}'(v_c) = 0,$$
$$(\mathbf{c}(u_+) - \mathbf{f}(v_c)) \cdot \mathbf{c}'(u_+) = 0. \tag{24}$$

Now we insert the Taylor expansions into these two equations. The absolute terms cancel because of (23). Vanishing of the first order terms yields two linear equations in u_c, u_+, v_c, from which we eliminate v_c and finally get

$$u_+ = Cu_c,$$

with

$$C = \frac{(\mathbf{c}_0' \cdot \mathbf{f}_0')^2}{[\mathbf{c}_0'^2 + (\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{c}_0''][\mathbf{f}_0'^2 + (\mathbf{f}^* - \mathbf{c}^*) \cdot \mathbf{f}_0'']}.$$

C is the constant we are looking for, since $u = 0$ corresponds to the foot point $\mathbf{c}^*$. To express C in geometric quantities, we use the properties of arc length parameterizations, $\mathbf{c}_0'^2 = \mathbf{f}_0'^2 = 1$, and denote the angle between the tangents at the normal foot points by ϕ, i.e. $\cos\phi = \mathbf{c}_0' \cdot \mathbf{f}_0'$. With $d \geq 0$ as distance between the foot points $\mathbf{c}^*$ and $\mathbf{f}^*$, we have $\mathbf{f}^* - \mathbf{c}^* = d\mathbf{n}$, $\|\mathbf{n}\| = 1$. So far, this gives

$$C = \frac{\cos^2\phi}{(1 - d\mathbf{n} \cdot \mathbf{c}_0'')(1 + d\mathbf{n} \cdot \mathbf{f}_0'')}.$$

By the Frenet equations, we have

$$\mathbf{c}_0'' = \kappa_c \mathbf{n}_c, \quad \mathbf{f}_0'' = \kappa_f \mathbf{n}_f,$$

with κ_c, κ_f as curvatures and $\mathbf{n}_c, \mathbf{n}_f$ as unit principal normal vectors of the curves $\mathbf{c}$ and $\mathbf{f}$, respectively. Of course, these entities are taken at the normal foot points. $\mathbf{n}_c \cdot \mathbf{n}$ equals the cosine of the angle γ_c between the common normal and the osculating plane of $\mathbf{c}$ at $\mathbf{c}^*$. The quantity $\kappa_c \cos\gamma_c$ can be seen as *normal curvature of the curve* $\mathbf{c}$ *with respect to the normal vector* $\mathbf{n}$. Analogously we define the normal curvature κ_f^n, but to have symmetry, we use the normal $-\mathbf{n}$ there (so that it points from $\mathbf{f}^*$ to $\mathbf{c}^*$). This finally yields

$$C = \frac{\cos^2\phi}{(1 - d\kappa_c^n)(1 - d\kappa_f^n)}. \tag{25}$$

Remark 6. The normal curvature κ_c^n of $\mathbf{c}$ at $\mathbf{c}^*$, with respect to the normal $\mathbf{n}$, can be visualized as follows: Connecting the point $\mathbf{f}^*$ with the curve $\mathbf{c}$ yields a cone. By developing this cone into the plane, $\mathbf{c}$ is transformed into a planar curve $\tilde{\mathbf{c}}$, whose ordinary curvature at $\tilde{\mathbf{c}}^*$ (with the normal orientation given by $\tilde{\mathbf{n}}$) is precisely κ_c^n (Do Carmo, 1976; Spivak, 1975). The interpretation of κ_f^n is analogous.

If the curves intersect, i.e. $d = 0$, the convergence only depends on their intersection angle. The property

$C = \cos^2\phi$ is immediately clear for two intersecting straight lines. It is not surprising that it appears in first order also if $\mathbf{c}$ and $\mathbf{f}$ are not lines. For curves $\mathbf{c}$ and $\mathbf{f}$, which are tangent at some point $\mathbf{c}^* = \mathbf{f}^*$, we have $d = 0$ and $\phi = 0$, and thus $C = 1$. This gives a convergence which is below a linear rate!

It is much more subtle to analyze the case $d \neq 0$. Obviously, even curvature information enters the discussion. The situation can be easily understood if one takes two circles $\mathbf{c}$ and $\mathbf{f}$ in the plane. Clearly, we have $\phi = 0$. The speed of convergence is determined by the radii of the circles; it is an elementary exercise to verify the validity of (22) with the constant from (25).

4.2. *Conclusions on the Performance of ICP*

We will only discuss the case of a small residual problem (d small); there, the data point cloud X fits very well onto Φ. By Eq. (25), the speed of convergence is given by $C \approx \cos^2\phi$ and thus we have to find the angle ϕ, under which the minimizer is approached.

By Eq. (5) squared distances between two positions, say the current position X and the minimizer X^*, are computed as sum of squared distances of corresponding data point locations,

$$\|X - X^*\|^2 = \|\alpha - \alpha^*\|^2 = \sum_i (\mathbf{x}_i - \mathbf{x}_i^*)^2. \quad (26)$$

As approximants to the tangent vectors at the minimizer (vectors $\mathbf{c}_0'$ and $\mathbf{f}_0'$ of the previous subsection), we may use the normalized secant vectors $(X - X^*)/\|X - X^*\|$ and $(Y - Y^*)/\|Y - Y^*\|$, and thus we have

$$\cos\phi \approx \frac{\sum_i (\mathbf{x}_i - \mathbf{x}_i^*) \cdot (\mathbf{y}_i - \mathbf{y}_i^*)}{\sqrt{\sum_i (\mathbf{x}_i - \mathbf{x}_i^*)^2}\sqrt{\sum_i (\mathbf{y}_i - \mathbf{y}_i^*)^2}}. \quad (27)$$

During the computation, X^* is not yet known. An alternative is the estimation of ϕ from two successive iterates,

$$\cos\phi \approx \frac{\sum_i (\mathbf{x}_{ic} - \mathbf{x}_{i+}) \cdot (\mathbf{y}_{ic} - \mathbf{y}_{i+})}{\sqrt{\sum_i (\mathbf{x}_{ic} - \mathbf{x}_{i+})^2}\sqrt{\sum_i (\mathbf{y}_{ic} - \mathbf{y}_{i+})^2}}. \quad (28)$$

This confirms an intuitively obvious and experimentally verified phenomenon: *ICP is very slow, if tangential moves along the surface are needed. Then the angle ϕ is small and the constant C is close to 1*. Tangential moves belong to a velocity vector field of a rigid body motion which is nearly tangential to Φ.

We have run a large number of experiments to empirically test the accuracy of the estimate (28) of te constant in the linear convergence behavior of ICP. A representative example is the following one. The chosen surface Φ is a bi-cubic B-spline surface with 36 control points and uniform knots. The size of the object is approximately $0.352 \times 0.340 \times 0.354$. The data set X results from random sampling of $k = 500$ points on Φ and successive displacement of the point cloud as a rigid body system; thus we have a zero residual problem. Figure 2 shows the initial and the final position after 200 iterative steps of standard ICP. Table 1 presents for each given iteration the error $E(j) = \sqrt{[\sum_i (\mathbf{x}_i - \mathbf{x}_i^*)^2]/k}$ according to (26), the estimate $\cos^2\phi$ of the constant C using formula (28) and the quotient $E(j)/E(j-1)$, which represents the exact error reduction in each iteration. The last two quantities are graphed in Fig. 2, bottom. It reveals that the theoretical convergence result describes the exact behavior very well, except for a few initial iterations when the data point cloud is far from the fixed object. This is expected, since we have performed a local convergence analysis which does not capture the initial phase.

Surfaces, which possess a velocity vector field $\mathbf{v}(\mathbf{x})$, such that $\mathbf{v}(\mathbf{x})$ is exactly tangential to Φ for all $\mathbf{x} \in \Phi$, are invariant under a uniform motion. Such a surface must be a plane, sphere, cylinder, rotational or helical surface. Clearly, for such a surface, F has an infinite number of minimizers. An instability can also exist infinitesimally or approximately. A linear algorithm for the detection of such cases can be based on line geometry (Pottmann and Wallner, 2001); strategies for handling them in an ICP algorithm have been described by Gelfand et al. (2003), and Ikemoto et al. (2003).

Let us summarize the results on the convergence of ICP.

Proposition 7. *The ICP algorithm exhibits in general linear convergence with a decay constant C given by Eq. (25). For a zero residual problem, where the minimizer is approached tangentially, we have the worst case $C = 1$; a tangential approach occurs in an exact way only for surfaces which are invariant under a uniform motion.*

Without further discussion, we mention that the quadratically convergent algorithms in the next section exhibit a better convergence for small angles ϕ than

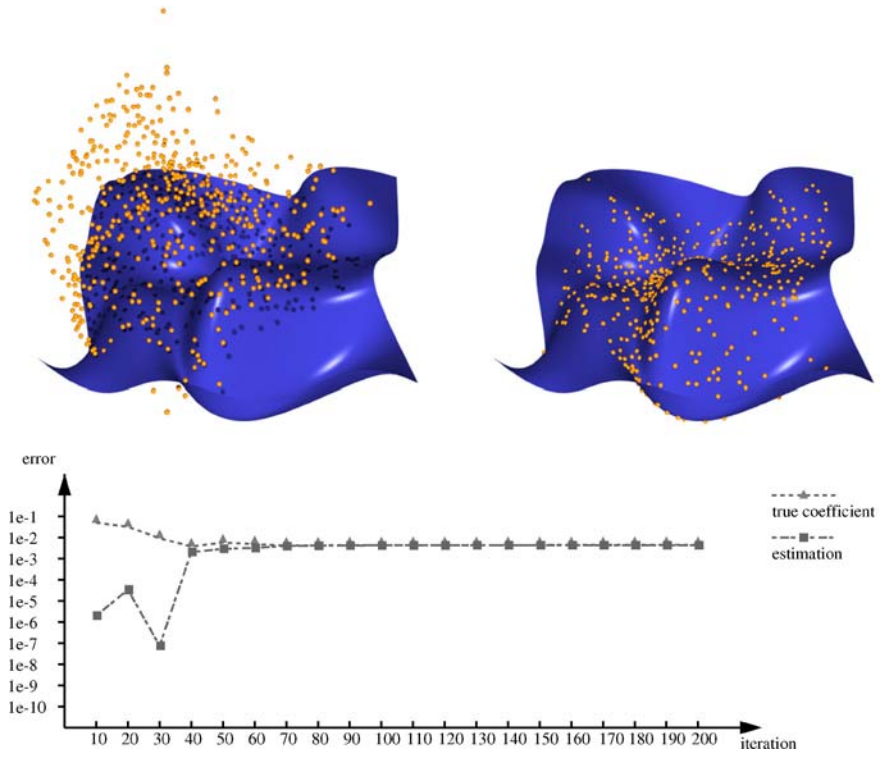


Figure 2. Local convergence behavior of the standard ICP algorithm: (upper left) initial position, (upper right) final position, (bottom) illustration of the constant C for the linear error reduction, and its estimate $\cos^2 \phi$ according to (28).

ICP does. However, they are no longer quadratically convergent for $\phi = 0$.

5. Registration Algorithms of the Newton-Type and Gauss–Newton Iteration

There are various possibilities to achieve quadratic convergence in registration algorithms. Recall that quadratic convergence means an error reduction of the form

$$\|\alpha_+ - \alpha^*\| \leq C\|\alpha_c - \alpha^*\|^2, \tag{29}$$

with some positive constant C. We will describe several algorithms, all of them based on a Newton type iteration or a simplification of it.

Algorithms which use quadratic approximants of the squared distance function according to (7) will be called *SDM (squared distance minimization)* algorithms. Those, which set $d = 0$ and thus use only squared tangent plane distances, are called *TDM* schemes. In this section, the following algorithms will be analyzed (in this order).

(1) *Affine SDM*. As a preparation for further discussions, we drop the rigidity constraint of the moving system and thus allow an affine distortion.
(2) *SDM 1*. This scheme, first proposed in (Pottmann et al., 2004), uses a linearization of the motion (rigidity constraint).
(3) *SDM 2* is an SDM scheme based on a second order motion approximant.
(4) *TDM* simplifications exist to all SDM algorithms mentioned above. We will show that they correspond to Gauss–Newton iteration and thus should be enhanced by regularization techniques such as the Levenberg–Marquardt method.

All these algorithms follow the same basic scheme and are quite easy to implement because of the careful study of the squared distance function and kinematical geometry; they can take advantage from preprocessing of the squared distance field (see (Mitra et al., 2004)). This geometric insight is lacking in a paper by Tucker and Kurfess (2003), which applies the Newton method to registration in a straightforward way and thus leads

Table 1. Error reduction in the standard ICP algorithm.

Iterative Closest Point (ICP)							
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\cos^2\phi$	j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\cos^2\phi$
0	8.607e-2			110	1.302e-7	0.8637	0.8633
10	5.291e-2	0.9696	0.5302	120	3.014e-8	0.8639	0.8635
20	3.453e-2	0.9510	0.6529	130	6.987e-9	0.8640	0.8636
30	1.631e-2	0.8968	0.3871	140	1.621e-9	0.8640	0.8637
40	3.670e-3	0.8573	0.8314	150	3.764e-10	0.8641	0.8638
50	8.478e-4	0.8754	0.8467	160	8.745e-11	0.8641	0.8639
60	1.957e-4	0.8685	0.8511	170	2.032e-11	0.8642	0.8639
70	4.720e-5	0.8616	0.8601	180	4.725e-12	0.8643	0.8639
80	1.071e-5	0.8625	0.8617	190	1.101e-12	0.8640	0.8639
90	2.451e-6	0.8631	0.8624	200	2.588e-13	0.8659	0.8638
100	5.640e-7	0.8635	0.8629				

to quite involved expressions and little possibilities for acceleration.

Newton Algorithms. Before we enter the discussion of registration, let us recall the most basic facts on Newton iteration (Kelley, 1999). A Newton method for the minimization of a function $F(\alpha)$ computes a second order Taylor approximant at the current position α_c and minimizes this quadratic function to obtain the next iterate α_+. Therefore, with the gradient $\nabla F(\alpha_c)$ and the Hessian $\nabla^2 F(\alpha_c)$, one has

$$\alpha_+ = \alpha_c - (\nabla^2 F(\alpha_c))^{-1} \cdot \nabla F(\alpha_c).$$

Under appropriate assumptions on F and on the initial iterate, a Newton iteration converges quadratically to a local minimizer. In order to obtain a globally convergent algorithm, i.e. an algorithm which converges from each initial position to a local minimizer, one has to make some improvements (Kelley, 1999). If the Hessian is not positive definite, the Newton direction may fail to be a descent direction; then one has to employ an approximate Hessian, which in our case will come from nonnegative quadratic approximants of the squared distance function. Moreover, one should use a *step size control* and compute a step λ such that

$$\alpha_+ = \alpha_c - \lambda(\nabla^2 F(\alpha_c))^{-1} \cdot \nabla F(\alpha_c),$$

has sufficient descent (Kelley, 1999). In the following, we will not explicitly point to this stabilization, but we are assuming it is done.

Gauss–Newton Iteration. Our objective function $F(\alpha)$ in Eq. (12) is a sum $(1/2)\sum_i d_i(\alpha)^2$ of squares. One speaks of a *nonlinear least squares problem* (Kelley, 1999). To avoid the costly computation of the full Hessian $\nabla^2 F$, one may use a *Gauss–Newton* iteration, which is equivalent to the solution of the linear least squares problem

$$\min \sum_{i=1}^{N} [d_i(\alpha_c) + \nabla d_i(\alpha_c) \cdot (\alpha - \alpha_c)]^2. \tag{30}$$

It is well-known (Kelley, 1999), pp. 24) that the distance $\|e_c\| = \|\alpha_c - \alpha^*\|$ of the current iterate to the minimizer α^* is related to the error $\|\alpha_+\|$ in the next iterate by

$$\|e_+\| \leq K(\|e_c\|^2 + \|R(\alpha^*)\|\,\|e_c\|). \tag{31}$$

Here, $R(\alpha^*) = (d_1, \ldots, d_N)(\alpha^*)$ is the residual at the minimizer, and K is an appropriate constant which involves the Jacobian of $R(\alpha)$. The error estimate is only true, if one is sufficiently close to the minimum. The well-known conclusions of (31) are: Gauss–Newton iteration converges quadratically for a zero residual problem. There, the data can be fitted exactly. Moreover, for good initial data and a small residual problem, convergence of Gauss–Newton is fast. For a large residual problem, the iteration may not converge at all.

Optimization theory provides several methods to achieve convergence of Gauss–Newton like iterations even for large residual problems (Kelley, 1999). A variant of the Gauss–Newton iteration does not apply the full step $\alpha_+ - \alpha_c$, but just a scalar multiple $\lambda(\alpha_+ - \alpha_c)$, usually with $\lambda < 1$, to the current iteration. Various methods for a line search along $\alpha_c + \lambda(\alpha_+ - \alpha_c)$ can be applied (see (Kelley, 1999)). This results in a so-called *damped Gauss–Newton algorithm*. Another way

to modify Gauss–Newton is a regularization with the *Levenberg–Marquardt method* (Kelley, 1999). Here, a scalar multiple of the unit matrix is added to the approximate Hessian.

5.1. *Affine SDM: A Newton Algorithm for Affine Registration*

As a preparation for further investigations let us first consider affine registration. In certain situations, affine registration may even be used for registration by a rigid body motion, namely if F possesses an isolated minimizer α^* within the affine group which is contained in (or very close to) M^6. A minimizer lies in M^6 if the deviations between data set and model shape are zero (up to Gaussian noise). This minimizer is isolated if there are no affine transformations of the model shape into itself.

Starting from an appropriate initial position α^0, we perform a Newton iteration in $\mathbb{R}^{12}$ for the minimization of F. This requires a second order approximation of the objective function F. Since F is the sum of squared distances of the data point positions $\mathbf{x}_i$ to the model shape Φ, a second order approximant is $F_2 = \sum_i F_{d,i}$, where $F_{d,i}$ is the second order approximant of the squared distance function to the model shape at $\mathbf{x}_i$. These approximants have been investigated in Section 2. Let $\mathbf{n}_{i,j} \cdot \mathbf{x} + h_{i,j} = 0$, $\|\mathbf{n}_{i,j}\| = 1$, for $j = 1, 2, 3$, be the equations of the coordinate planes of the principal frame at $\mathbf{x}_i$'s closest point $\mathbf{y}_i \in \Phi$. Then, by Eq. (8), a second order Taylor approximant of the squared distance function at $\mathbf{x}_i$ is written as

$$F_{d,i}(\mathbf{x}) = \sum_{j=1}^{3} \alpha_{i,j} (\mathbf{n}_{i,j} \cdot \mathbf{x} + h_{i,j})^2. \tag{32}$$

The same form holds for a nonnegative modification. Nonnegative approximants should be applied at least in initial steps of the iteration to ensure positive definiteness of the Hessian of the objective function.

We now insert an affine displacement of the data points,

$$\mathbf{x}_i' = \mathbf{x}_i + \mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3, \tag{33}$$

into F_2 and arrive at the local quadratic model of the objective function

$$F_2 = \sum_i \sum_{j=1}^{3} \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i + \mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3) + h_{i,j}]^2. \tag{34}$$

Since $\mathbf{n}_{i,j} \cdot \mathbf{x}_i + h_{i,j}$ is the distance of $\mathbf{x}_i$ to the jth coordinate plane of the principal frame, this value equals 0 for $j = 1, 2$; it equals the oriented distance d_i of $\mathbf{x}_i$ to Φ for $j = 3$. Therefore we may rewrite F_2 as

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3)]^2 + \tilde{F}_2. \tag{35}$$

Here, $\tilde{F}_2$ denotes the part arising from the squared distances to the tangent planes at the closest points $\mathbf{y}_i$,

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3) + d_i]^2. \tag{36}$$

The minimization of the quadratic function F_2 in the parameters $(\mathbf{c}_0, \ldots, \mathbf{c}_3)$ of the affine displacement requires the solution of a linear system. Applying this affine displacement to the data set, we obtain a new position. This procedure is iterated. We stop with an appropriate criterion, e.g. if the error or its decrease fall below a given threshold or a maximum number of iterations has been reached. To enforce that the final position of the data set is a Euclidean copy of the original one, we may register the original position to the final affine position, which is a well-known eigenvalue problem (the second step in each iteration of ICP).

Since the present method is a Newton algorithm, it converges *quadratically*.

Remark 8. Affine registration in the present formulation has an infinite number of singular solutions: These occur if the whole moving system shrinks to a single point of the model shape, which clearly results in a zero residual. Our experiments confirm that this shrinking effect may appear if the initial position is too far away from the model shape (cf. Table 3).

5.2. *SDM 1: A Newton Algorithm Based on a First Order Motion Approximant*

SDM 1 according to (Pottmann et al., 2004) keeps the rigidity constraint, i.e., the path in $\mathbb{R}^{12}$ towards the minimizer is restricted to M^6. Let us first explain the iterative procedure in $\mathbb{R}^{12}$. Here, each iteration from α_c to α_+ consists of the following two steps.

(1) Compute the tangent space T^6 of M^6 at α_c and minimize a local quadratic model F_2 of the objective function F within T^6. Let α_T^* denote the unique minimum in T^6.
(2) Project α_T^* onto M^6 to obtain α_+.

Such a projected Newton algorithm needs not even be convergent if the unconstrained minimizer α^u (in $\mathbb{R}^{12}$) is far away from M^6. However, *if $\alpha^u = \alpha^*$ lies in M^6, the algorithm can be shown to be quadratically convergent*. These results follow by a local quadratic approximation of the objective function at the minimizer and by the use of corresponding results on the constrained minimization of quadratic functions (see, e.g. (Hofer and Pottmann, 2004)).

The realization of the two steps in SDM 1 is as follows (Pottmann et al., 2004).

Step 1. The tangent space T^6 is defined by Euclidean velocity fields, i.e. $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. Equivalently, $\mathbf{c}_i$ of (33) are no longer arbitrary, but define a skew symmetric matrix. Therefore, minimization of the local quadratic model inside T^6 requires the minimization of the quadratic function F_2 in $(\mathbf{c}, \bar{\mathbf{c}})$,

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)]^2 + \tilde{F}_2. \quad (37)$$

As before, $\tilde{F}_2$ arises from squared tangent plane distances,

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) + d_i]^2 = \sum_i [\mathbf{n}_i \cdot \bar{\mathbf{c}} + \bar{\mathbf{n}}_i \cdot \mathbf{c} + d_i]^2, \quad (38)$$

and can be used instead of F_2 when we are already close to the model shape (see also Section 5.4). Note that $(\mathbf{n}_i, \bar{\mathbf{n}}_i)$ are the Plücker coordinates of the surface normal through $\mathbf{x}_i$. F_2 is a quadratic function in the unknowns $(\mathbf{c}, \bar{\mathbf{c}})$. The unique solution $(\mathbf{c}^*, \bar{\mathbf{c}}^*)$ can be given explicitly by solving a system of linear equations.

Step 2. The projection back to M^6 proceeds according to Section 2. We apply a helical motion which is determined by the velocity field $(\mathbf{c}^*, \bar{\mathbf{c}}^*)$.

The presented algorithm can be made convergent in any situation, even if the minimum within the affine group is not close to M^6. However, then the choice of the rotational angle cannot simply be $\|\mathbf{c}\|$. Especially, if a large rotational angle arises, it is better to use $\arctan \|\mathbf{c}\|$ or even a smaller value than that. A secure way is to employ the Armijo rule or a similar strategy from optimization (Kelley, 1999) for the determination of an appropriate step size, analogous to the procedure in a gradient descent algorithm. It is well known in optimization (Kelley, 1999) that this results in an algorithm with *linear convergence*.

5.3. *SDM 2: A Newton Algorithm with Second Order Motion Approximation*

To achieve quadratic convergence in any case, we can use a second order approximant for the motion from α_c to α_+ according to (4). This means that we estimate the displaced data point $\mathbf{x}_i$ by

$$\mathbf{x}_i' = \mathbf{x}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i + T_{i,2},$$

with the second order term

$$T_{i,2} = \frac{1}{2}[\mathbf{c} \times \bar{\mathbf{c}} + (\mathbf{c} \cdot \mathbf{x}_i)\mathbf{c} - \mathbf{c}^2 \mathbf{x}_i].$$

We insert this into $F_d(\mathbf{x}_i)$ and sum up,

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i + T_{i,2})]^2 + \tilde{F}_2. \quad (39)$$

We observe that the quadratic term $T_{i,2}$ in the first part produces just cubic or quartic contributions to F_2. Since we will minimize a local quadratic model at $(\mathbf{c}, \bar{\mathbf{c}}) = (0, 0)$, these terms do not matter at all. However, we have to look into $\tilde{F}_2$,

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i + T_{i,2}) + d_i]^2, \quad (40)$$

Skipping again the higher order terms, we get a local quadratic approximant, denoted by $\tilde{F}_2'$,

$$\tilde{F}_2' = \sum_i [\mathbf{n}_i \cdot \bar{\mathbf{c}} + \bar{\mathbf{n}}_i \cdot \mathbf{c} + d_i]^2 + 2 \sum_i d_i \mathbf{n}_i \cdot T_{i,2}. \quad (41)$$

Hence, the only relevant correction term compared to the use of a linearized motion as in Subsection 5.2 is

$$F_{2c} = \sum_i d_i \, [\det(\mathbf{n}_i, \mathbf{c}, \bar{\mathbf{c}}) + (\mathbf{c} \cdot \mathbf{x}_i)(\mathbf{c} \cdot \mathbf{n}_i) - \mathbf{c}^2 (\mathbf{x}_i \cdot \mathbf{n}_i)]. \quad (42)$$

Computationally, the second order motion approximation does not require much more effort. However, in

this refined version we can guarantee quadratic convergence provided that we have an initial position in the region of attraction of the minimizer.

5.4. *TDM: Registration via Gauss–Newton Iteration.*

At the final steps, the second order approximants $F_d(\mathbf{x}_i)$ will be close to squared distance functions of tangent planes at the closest points $\mathbf{y}_i$. This means that the influence of $\alpha_{i,1}$ and $\alpha_{i,2}$ will be negligible and thus these parameters can be set to zero. We may then simply use $\tilde{F}_2$ instead of F_2; this results in the TDM scheme associated with any of the SDM methods above. Minimization of squared tangent plane distances has been first proposed by (Chen and Medioni, 1991), and it has been observed in various papers that this results in faster convergence than ICP. We will give a deeper explanation by showing that TDM corresponds to Gauss–Newton iteration.

We have to formulate Eq. (30) for the present registration problem. It is better to start the discussion without the rigidity constraint, i.e., to consider affine registration. For this, we note that the gradient $\nabla d(\mathbf{x}_i)$ of the distance function to Φ at a point $\mathbf{x}_i$, taken with respect to the spatial coordinates $\mathbf{x}$, is given by the unit normal vector $\mathbf{n}_i$ at $\mathbf{x}_i$'s closest point $\mathbf{y}_i \in \Phi$, $\nabla d_i = (\mathbf{x}_i - \mathbf{y}_i)/\|\mathbf{x}_i - \mathbf{y}_i\| = \mathbf{n}_i$. The term $\nabla d_i(\alpha_c)\cdot(\alpha - \alpha_c)$ describes, in our case, the directional derivative of this distance for an affine displacement α of $\mathbf{x}_i$, which equals $\mathbf{n}_i \cdot (\bar{\mathbf{c}} + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3)$. Therefore, the minimization of squared tangent plane distances, which is described in (36), is identical to Gauss–Newton iteration (30).

Adding the rigidity constraint has been discussed in Subsections 5.2 and 5.3; in exactly the same way we can handle the constraint for Gauss–Newton iteration (TDM). The only difference is that we do not use full local quadratic approximants of the squared distance function at the current data points, but we only use squared tangent plane distances, described in function $\tilde{F}_2$. Let us summarize the conclusions one can make with known results from optimization, which have been mentioned at the beginning of this section.

Proposition 9. *Registration algorithms which are based on the approach by (Chen and Medioni, 1991) and iteratively minimize the sum of squared distances to tangent planes at the closest points $\mathbf{y}_i \in \Phi$ of the current data point locations $\mathbf{x}_i$, correspond to a Gauss–Newton iteration. Therefore, these algorithms converge quadratically for a sufficiently good initial position and a zero residual problem (i.e., the data shape fits exactly onto the model shape).*

Moreover, one has to expect that the Chen & Medioni method works well for small residual problems. To achieve convergence even for a large residual problem, one should employ the modifications addressed above. For instance, *Levenberg–Marquardt (L-M) regularization* applied to Gauss–Newton iteration with a first order motion approximant requires iterative minimization of

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) + d_i]^2 + \nu(\bar{\mathbf{c}}^2 + \mathbf{c}^2). \quad (43)$$

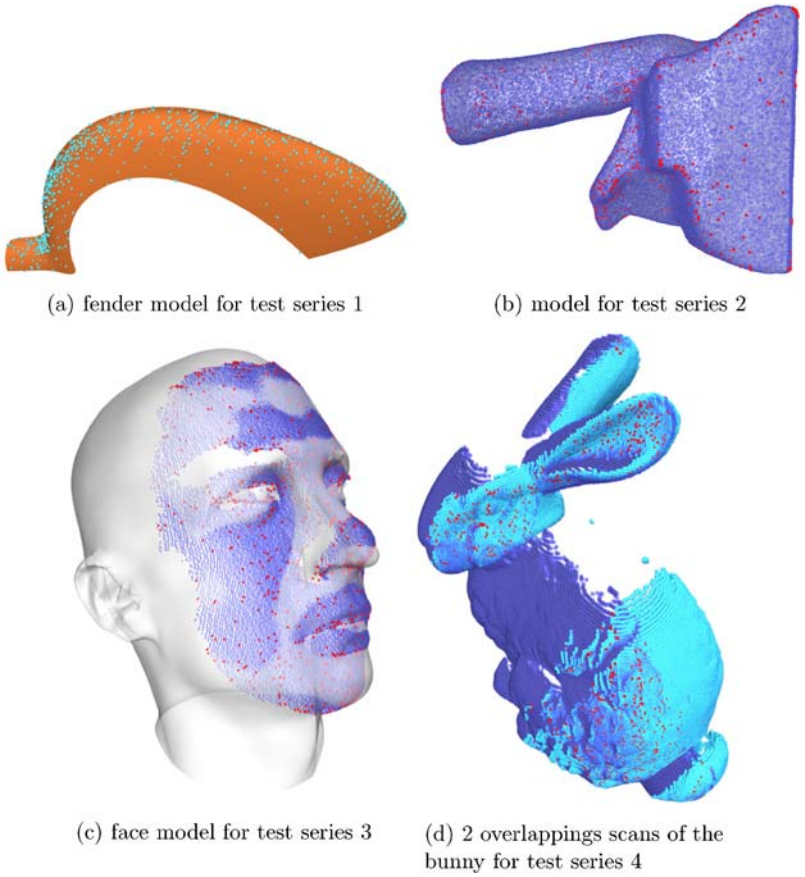
For the choice of the parameter ν, we refer to (Kelley, 1999).

6. Experimental Validation

In this section, we present an experimental verification of the theoretical results. We compare the standard ICP algorithm and the four algorithms presented in Subsections 5.1–5.4 with respect to local convergence and global stability. The experiments have been performed on a Pentium IV 2.8G with 1G RAM.

We present the results of test series run on four different models and corresponding data sets (Fig. 3). We test both local convergence and global stability and do this by an evaluation of the algorithms for a large number of initial positions; examples for such positions are shown in Fig. 4.

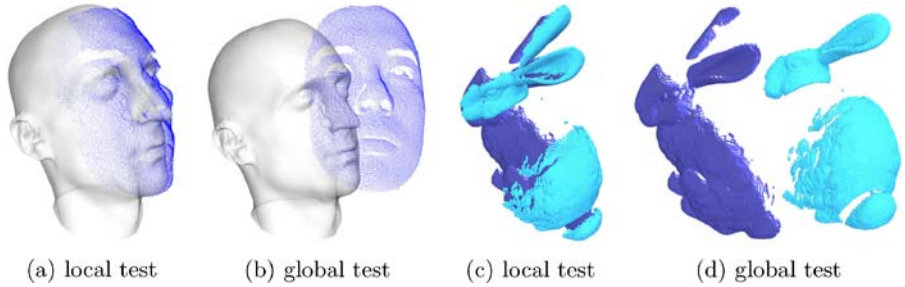
(1) *Test 1* uses a fender model represented as a bicubic B-spline surface (Fig. 3(a)). The size of the object is approximately $2.023 \times 0.750 \times 0.367$. The data set is obtained by sampling the model at $k = 500$ points with small Gaussian noise; this is done by normal-uniform sampling (Rusinkiewicz and Levoy, 2001). Newton's method is employed to compute on demand the closest point of each data point.
(2) In *Test 2* the original model is a triangle mesh (obtained from real measurement data; see Fig. 3(b)); it has a size of $0.2183 \times 0.2727 \times 0.2155$. Having an application in industrial inspection in mind, where the original model would be used many times, we use preprocessing by the modified d2tree



(a) fender model for test series 1 (b) model for test series 2

(c) face model for test series 3 (d) 2 overlappings scans of the bunny for test series 4

Figure 3. Models and data sets in registered position; the sample points used for registration in (b), (c) and (d) are shown in red.



(a) local test (b) global test (c) local test (d) global test

Figure 4. Some initial positions for local and global convergence tests; in all cases the shown positions yield convergence.

from Section 2. Data point clouds are obtained by random sampling at $k = 1000$ points with small Gaussian noise.

(3) *Test 3* concerns a large residual problem (Fig. 3(c)), where model shape and data point clouds come from different face models. Evaluation of the squared distance function is done via preprocessing.

(4) *Test 4* shows the process of aligning two partially overlapping scans of the bunny model (Figs. 3(d) and 4). During registration, the dark point cloud S is fixed and the other point cloud is active. We sampled the active part at $k = 500$ points and use preprocessing of S with the modified d2tree. Three parameters T_1, T_2, τ are used to exclude sample points that do not belong to the overlapping region.

In the objective function, the term of each sample $\mathbf{x}_i$ is multiplied by a weight w_i according to

$$w_i := \begin{cases} 1 & \text{if } d^2(\mathbf{x}_i, S) \le T_1, \\ \exp(-\tau(d^2(\mathbf{x}_i, S) - T_1)^2) & \text{if } T_1 < d^2(\mathbf{x}_i, S) < T_2, \\ 0 & \text{if } T_2 \le d^2(\mathbf{x}_i, S). \end{cases} \tag{44}$$

In general, T_1, T_2 need to be reduced during the optimization. The constant τ controls the decay of influence and can be increased in later steps of the iterative procedure. In the final few steps, we set $T_1 = T_2$.

Computation of *initial positions* of the data sets is performed as follows. We apply helical motions with parameters $(\mathbf{c}, \bar{\mathbf{c}})$ (cf. Section 2) to the optimally aligned position α^* of the data set. Thus, the latter belongs to parameters $(0, \dots, 0)$. The $(\mathbf{c}, \bar{\mathbf{c}})$–parameter domain is uniformly sampled as $(\beta i_x, \beta i_y, \beta i_z, Lj_x, Lj_y, Lj_z)$, where $-3 \le j_x, j_y, j_z \le 3$. For the local convergence tests we use $-3 \le i_x, i_y, i_z \le 3$, $\beta = \pi/90$ and L is 1/50 of the diagonal D_b of the bounding box of the model shape. Initial positions for global convergence tests belong to $-4 \le i_x, i_y, i_z \le 4$, $\beta = \pi/9$ and $L = D_b/8$.

6.1. *Local Convergence.*

The results of the four test series run on the initial positions as explained above are shown via 3D graphs in Figs. 5–8. Along the three axes we graph iteration number j, logarithm $\log(E(j))$ of the error (with $E(j)$ defined as in Section 4.2), and the number of occurrences in the test series. This shows that the 2D convergence graphs to the individual initial positions are pretty close to each other; we have a strong concentration at a certain mean plot.

The error decay in the experiments is in agreement with the theory. Affine SDM and SDM 2 are quadratically convergent, SDM 1, TDM and standard ICP are linearly convergent; the latter exhibits the clearly worst local convergence behavior. We also see that TDM does not work well for large residual problems. Basically the same picture is found for the total computation time (Table 2).

Let us discuss in more detail the time required per iteration. In all algorithms, the dominant cost lies in computing various approximants of the squared distance function. We analyze the cost t_{total} for one data point. Essentially, $t_{\text{total}} = t_{\text{query}} + t_a$ sums up the time t_{query} required for finding the closest point, and the time t_a for computing the squared distance approximant at the closest point. As we use the same data structure for computing the closest point, t_{query} remains fixed along all tested algorithms. TDM requires less information than SDM, and this yields in practice $t_a^{TDM} \approx 0.5 * t_a^{SDM}$. Thus, if t_{query} is dominant (as for test 1), t_{total} for SDM and TDM are nearly the same. If t_a is dominant (d2tree preprocessing), we have $t_{\text{total}}^{\text{TDM}} \approx 0.5 \cdot t_{\text{total}}^{\text{SDM}}$. However, faster convergence of SDM can still give lower total computation time of the registration process (Table 2).

6.2. *Global Stability.*

After examining the local convergence, we come to the global stability issue. We are interested here in the size of the funnel of attraction of the minimizer. Since pure Newton and Gauss Newton methods would not do well at all, we have to enhance them by step size control and/or L-M regularization.

For each algorithm, we let the moving system start from the initial positions as outlined above and count the percentage of those initial positions that result in the global minimizer α^*; the results are collected in Table 3.

We summarize the results as follows. SDM 1 and standard ICP show similar global behavior, but they are only a little better than SDM 2. TDM has a worse global behavior. As expected, affine SDM is the worst one, since it is prone to exhibit a shrinking effect when the initial position is far from the shape model.

Table 2. Average computational time of each algorithm.

	Test 1	Test 2	Face registration	Partial alignment
Preprocessing	0s	19.616s	21.571s	15.421s
Samples	1000	1000	1000	500
Standard ICP	11.12s	0.515s	0.637s	0.582s
TDM	1.623s	0.101s	0.153s	0.074s
SDM 1	0.523s	0.118s	0.204s	0.042s
SDM 2	0.409s	0.085s	0.104s	0.034s
affine SDM	0.334s	0.056s	0.087s	0.026s

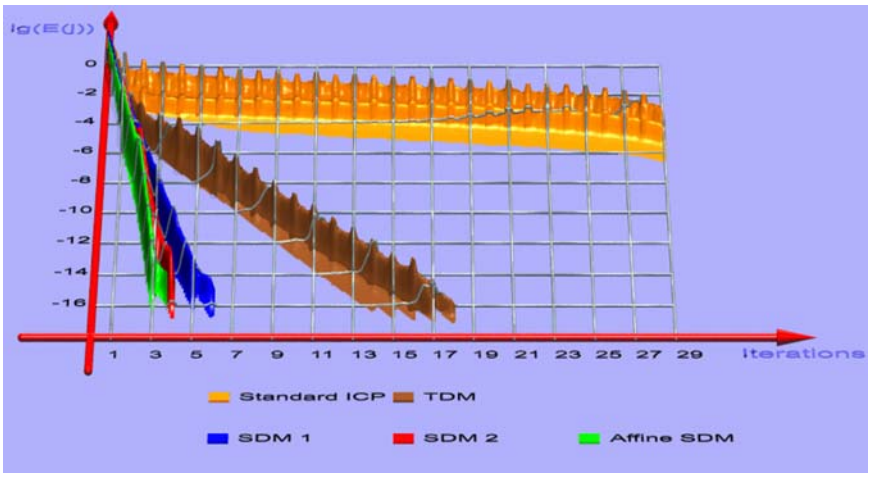


Figure 5. Error reduction for local convergence test 1 (small residual).

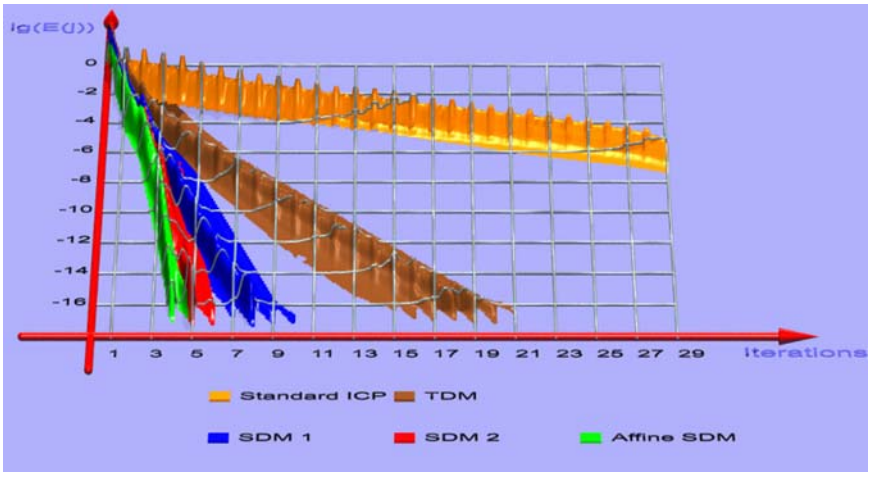


Figure 6. Error reduction for local convergence test 2 (small residual).

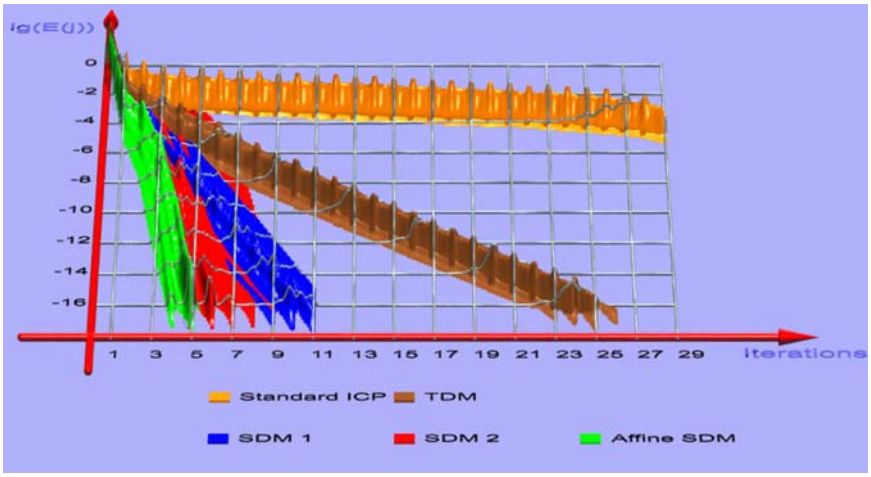


Figure 7. Error reduction for local convergence test 3 (large residual).

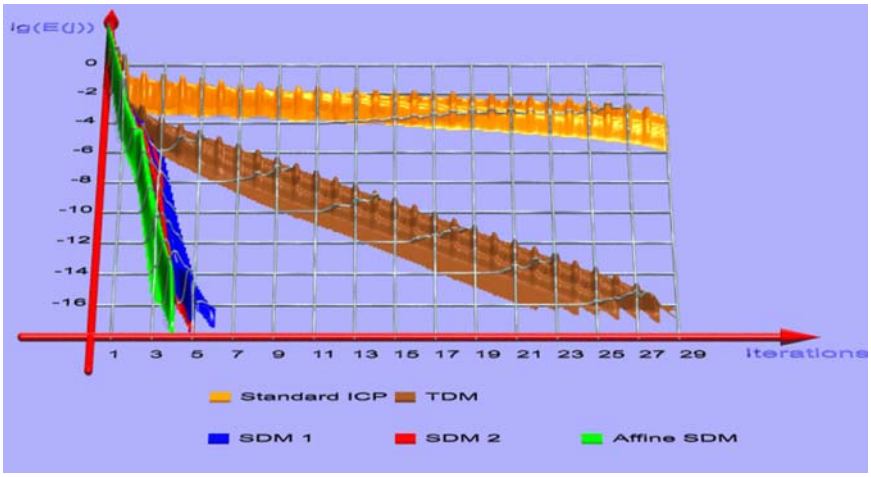


Figure 8. Error reduction for local convergence test 4 (partial matching).

Table 3. Percentage of initial positions which belong to the convergence funnel of the global minimizer.

	Test 1	Test 2	Face registration	Partial alignment
Standard ICP	30.902%	29.921%	6.721%	12.874%
TDM + Armijo rule	10.209%	5.726%	2.978%	3.656%
TDM + L-M method	19.743%	12.354%	4.567%	6.493%
SDM 1 + Armijo rule	37.434%	28.545%	8.721%	14.218%
SDM 1+ L–M method	40.211%	30.671%	9.326%	15.951%
SDM 2 + Armijo rule	20.825%	16.768%	4.822%	7.869%
SDM 2 + L–M method	22.579%	18.546%	6.542%	9.928%
Affine SDM + Armijo rule	0.207%	0.157%	0.056%	0.282%
Affine SDM + L–M method	0.588%	0.352%	0.122	0.633%%

7. Conclusion and Future Research

Exploiting the geometry of the squared distance function and using known facts from kinematical geometry and optimization, we have been able to analyze and improve the *local convergence behavior* of registration algorithms. In particular, we have proposed algorithms with local quadratic convergence. Moreover, an experimental validation of the theoretical results has been given. Both from the theoretical study and the experiments we conclude that the most widely used ICP algorithm (Besl and McKay, 1992) is the slowest. The algorithm of (Chen and Medioni, 1991) (essentially a Gauss–Newton iteration) shows faster convergence, but it is still behind the other schemes (SDM1, SDM2, affine SDM) with respect to convergence rate and computation time.

It has also been shown that the quadratically convergent algorithms (with regularization and step size control) do not require a very close but still a reasonable initial position. Finding such a position requires different techniques; a promising direction is the approach by (Sharp et al., 2002).

The presented optimization algorithms can be extended to the *simultaneous registration of multiple views*. Also there, we can achieve local quadratic convergence and verify it by experiments. For registration of multiple views, the choice of a good initial position is even more critical, and thus we will discuss multiview registration in a separate paper, both from the perspective of local and global convergence.

Acknowledgements

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 (FWF) under grant P16002-N05, by the innovative project '3D Technology' of Vienna University of Technology, and by the Natural Science Foundation of China under grants 60225016 and 60321002.

References

Bernardini, F. and Rushmeier, H. 2002. The 3D model acquisition pipeline. Computer Graphics Forum, 21:149–172.

Besl, P.J. and McKay, N.D. 1992. A method for registration of 3D shapes. IEEE Trans. Pattern Anal. and Machine Intell., 14:239–256.

Bottema, O. and Roth, B. 1990. *Theoretical Kinematics*. Dover: New York.

Chen, Y. and Medioni, G. 1991. Object modeling by registration of multiple range images. Proc. IEEE Conf. on Robotics and Automation.

Do Carmo, M.P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice Hall.

Eggert, D.W., Fitzgibbon, A.W., and Fisher, R.B. 1998. Simultaneous registration of multiple range views for use in reverse engineering of CAD models. Computer Vision and Image Understanding, 69:253–272.

Eggert, D.W., Lorusso, A., and Fisher, R.B. 1997. Estimating 3-D rigid body transformations: a comparison of four major algorithms. Machine Vision and Applications, 9:272–290.

Gelfand, N., Ikemoto, L., Rusinkiewicz, S., and Levoy, M. 2003. Geometrically stable sampling for the ICP algorithm, Proc. Intl. Conf. on 3D Digital Imaging and Modeling.

Faugeras, O.D. and Hebert, M. 1986. The representation, recognition, and locating of 3-D objects. Int. J. Robotic Res., 5:27–52.

Fletcher, R. 1987. *Practical Methods of Optimization*. Wiley: New York.

Geiger, C. and Kanzow, C. 2002. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer: Heidelberg.

Hofer, M., Pottmann, H., and Ravani, B. 2004. From curve design algorithms to motion design. Visual Computer, 20:279–297.

Hofer, M. and Pottmann, H. 2004. Algorithms for constrained minimization of quadratic functions—geometry and convergence analysis. Tech. Rep. 121, TU Wien, Geometry Preprint Series. http://www.geometrie.tuwien.ac.at/ig/papers/foot_tr121.pdf.

Horn, B.K.P. 1987. Closed form solution of absolute orientation using unit quaternions. Journal of the Optical Society A, 4:629–642.

Huber, D. and Hebert, M. 2003. Fully Automatic Registration of Mutiple 3D Data Sets. Image and Vision Computing, 21:637–650.

Ikemoto, L., Gelfand, N., and Levoy, M. 2003. A hierarchical method for aligning warped meshes. Proc. Intl. Conf. on 3D Digital Imaging and Modeling.

Kelley, C.T. 1999. *Iterative Methods for Optimization*. SIAM: Philadelphia.

Leopoldseder, S., Pottmann, H., and Zhao, H. 2003. The d2-tree: A hierarchical representation of the squared distance function, Tech. Rep. 101, Institute of Geometry, Vienna University of Technology.

Mian, A.S., Bennamoun, M., and Owens, R. 2004. Matching tensors for automatic correspondence and registration. Proceedings of ECCV'04, Springer LNCS 3022, pp. 495–505.

Mitra, N., Gelfand, N., Pottmann, H., and Guibas, L. 2004. Registration of point cloud data from a geometric optimization perspective. Proc. Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 23–32.

Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H.P. 2003. Multi-level partition of unity implicits. ACM Trans. Graphics, 22:153–161.

Planitz, B.M., Maeder, A.J, and Williams, J.A. 2005. The correspondence framework for 3D surface matching algorithms. Computer Vision and Image Understanding, 97:347–383.

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

Pottmann, H., Leopoldseder, S., and Hofer, M. 2004. Registration without ICP. Computer Vision and Image Understanding, 95:54–71.

Pottmann, H. and Wallner, J. 2001. *Computational Line Geometry*. Springer-Verlag.

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

Rusinkiewicz, S. and Levoy, M. 2001. Efficient variants of the ICP algorithm. Proc. 3rd Int. Conf. on 3D Digital Imaging and Modeling, Quebec.

Sharp, G.C., Lee, S.W., and Wehe, D.K. 2002. ICP registration using invariant features, IEEE Trans. Pattern Analysis and Machine Intelligence, 24:90–102.

Spivak, M. 1975. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish.

Tsin, Y. and Kanade, T. 2004. A correlation-based approach to robust point set registration. Proceedings of ECCV'04, Springer LNCS 3023, pp. 558–569.

Tucker, T.M. and Kurfess, T.R. 2003. Newton methods for parametric surface registration, Part 1: Theory. Computer-Aided Design, 35:107–114.