Geometry and convergence analysis of algorithms for registration of 3D shapes

Helmut Pottmann

Geometric Modeling and Industrial Geometry Group
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

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 known registration algorithms such as ICP and introduce new algorithms with an improved local and global convergence behavior. Most of our work deals with the fundamental problem of registering two views (scans, surfaces) with unknown correspondences. It is then shown how to extend the concepts to the simultaneous registration of an arbitrary number of views.

Key words: 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 [1]. 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 before we address the simultaneous registration of more than two systems.

Previous work

A well-known standard algorithm to solve the present registration problem is the *iterative closest point (ICP) algorithm* of Besl and McKay [2]. Independently, Chen and Medioni [4] 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 [29], who also suggest that *iterative corresponding point* is a better expansion for the abbreviation ICP than the original *iterative closest point*. For an overview of the recent literature on registration we also refer to [6,8,16,22,28] and the references therein.

Contributions 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 it leads to new algorithms with improved local or global convergence properties.

This paper is organized as follows. Section 2 summarizes basic facts from kinematical geometry which are important for the present investigation. Further essential background concerns the geometry of the distance function to a surface, which is reviewed in Sec. 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 in Sec. 4. These algorithms are of importance for the global convergence behavior; the local convergence is just linear. Registration with the ICP algorithm is discussed in Sec. 5. It is shown that ICP exhibits *linear convergence*. The constant, which determines the convergence speed, is

closely related to the surface geometry and the direction from which the minimum is approached. To obtain better local convergence, we devise and analyze in Sec. 6 algorithms with quadratic convergence; such algorithms are of the Newton type and require second order approximants of the objective function. Simplified versions, which are frequently used optimization algorithms for certain types of nonlinear least squares problems, are Gauss–Newton iteration and the Levenberg–Marquart method. These are addressed in Sec. 7. In fact, Gauss–Newton iteration turns out to be precisely the algorithm of Chen and Medioni. Therefore, this algorithm exhibits quadratic convergence for a good initial position and a zero residual problem (the data point cloud fits precisely onto the model surface). Finally, in Sec. 8 we show how to use the previous results for the simultaneous registration of more than two views.

The present investigation provides the theoretical basis for empirical results which have been reported in earlier papers. It also brings some order into the variety of registration algorithms and presents new concepts, partially in continuation of the work based on the geometry of the squared distance function [25,26].

2 Spatial Kinematics

Since registration requires the computation of an optimal rigid body motion, it is not surprising that kinematical geometry plays an important role. We review here some basic facts. Proofs are omitted in most cases; they may be found in the literature, e.g. [3,27].

2.1 First order properties of one-parameter motions

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 space* or *moving system* Σ^0, and one copy called the *fixed space* or *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

$$\mathbf{x}(t) = A(t) \cdot \mathbf{x}^0 + \mathbf{a}_0(t) \tag{1}$$

of Σ, where $A(t) \in \mathrm{SO}_3$ and $\mathbf{a}_0(t) \in \mathbb{R}^3$. In this way all points of Σ^0 have a

path curve or *trajectory* $\mathbf{x}(t)$ in Σ. The trajectory of the origin is $\mathbf{a}_0(t)$. $A(t)$ describes the rotational part of the motion; 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

$$\mathbf{v}(\mathbf{x}) = \overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}. \tag{2}$$

The vector $\mathbf{c}$ is called *Darboux vector* or *vector of angular velocity.*

Of special interest are the *uniform* motions, whose velocity vector field is constant over time. Apart from the trivial uniform motion, where nothing moves at all and all velocities are zero, there are the following three cases:

(1) Uniform translations have $\mathbf{c} = \mathbf{o}$, but $\overline{\mathbf{c}} \neq \mathbf{o}$, i.e., all velocity vectors equal $\overline{\mathbf{c}}$.
(2) Uniform rotations with nonzero angular velocity about a fixed axis. We have $\mathbf{c} \cdot \overline{\mathbf{c}} = 0$, but $\mathbf{c} \neq \mathbf{o}$.
(3) Uniform helical motions are the superposition of a uniform rotation and a uniform translation parallel to the rotation's axis. They are characterized by $\mathbf{c} \cdot \overline{\mathbf{c}} \neq 0$. 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. Formally, $p = 0$ means a uniform rotation and $p = \infty$ is a translation.

Up to the first differentiation order, any one-parameter motion agrees locally with one of these motions.

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

$$p = (\mathbf{c} \cdot \overline{\mathbf{c}})/\mathbf{c}^2, \quad \omega = \|\mathbf{c}\|, \quad (\mathbf{g}, \overline{\mathbf{g}}) = (\mathbf{c}, \overline{\mathbf{c}} - p\mathbf{c}). \tag{3}$$

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

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

$$\mathbf{x}_1(t) = \mathbf{x}(0) + t\dot{\mathbf{x}}(0) = \mathbf{x}_0 + t(\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0). \tag{4}$$

Here, $\mathbf{x}_0$ is the position of $\mathbf{x}^0 \in \Sigma^0$ at $t = 0$ in Σ. We are now going to derive a *second order approximant*. For our purposes, it is sufficient to consider uniform motions, since we are only interested in a local parameterization of the motion group which is precise up to second order. Hence, we have to compute $\ddot{\mathbf{x}}(0)$, the acceleration vector of $\mathbf{x}^0$ at time $t = 0$. By the subgroup property of uniform motions, the velocity vector field is time independent. Considering the motion of a point, its velocity vector is just transformed by the linear (rotational) part of the motion. Hence, its derivative equals $\dot{\mathbf{v}} = \mathbf{c} \times \mathbf{v}$. Therefore, we have $\ddot{\mathbf{x}}(0) = \mathbf{c} \times (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0)$, and a second order Taylor approximation of a uniform motion is given by

$$\mathbf{x}_2(t) = \mathbf{x}_0 + t(\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0) + \frac{t^2}{2}\mathbf{c} \times (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0). \tag{5}$$

This formula follows also immediately from the representation of uniform motions via the exponential mapping [3].

We will later use equation (5) 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 $(\mathbf{c}, \bar{\mathbf{c}})$,

$$\begin{aligned}\mathbf{x}(\mathbf{c}, \bar{\mathbf{c}}) &= \mathbf{x}_0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0 + \frac{1}{2}\mathbf{c} \times (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0) \\ &= \mathbf{x}_0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_0 + \frac{1}{2}[\mathbf{c} \times \bar{\mathbf{c}} + (\mathbf{c} \cdot \mathbf{x}_0)\mathbf{c} - \mathbf{c}^2\mathbf{x}_0].\end{aligned} \tag{6}$$

2.3 Relative Motions

We consider three copies Σ^i, Σ^j, Σ^k of Euclidean space. Σ^i performs a one-parameter motion with respect to Σ^j and Σ^j does the same with respect to Σ^k. This results in a composite motion of Σ^i with respect to Σ^k. Then, the relative velocities of these three motions, $\mathbf{v}_{ij}$, $\mathbf{v}_{jk}$, $\mathbf{v}_{ik}$, at some point, say $\mathbf{x}^k \in \Sigma^k$,

satisfy the relation

$$\mathbf{v}_{ik}(\mathbf{x}^k) = \mathbf{v}_{ij}(\mathbf{x}^k) + \mathbf{v}_{jk}(\mathbf{x}^k). \tag{7}$$

For simpler notation, we view Σ^k as fixed system Σ and write just single indices. The velocity fields $\mathbf{v}_{ik} = \mathbf{v}_i$ and $\mathbf{v}_{jk} = \mathbf{v}_j$ shall be described according to (2) by vector pairs $(\mathbf{c}_i, \overline{\mathbf{c}}_i)$ and $(\mathbf{c}_j, \overline{\mathbf{c}}_j)$, respectively. Then, the velocity field $\mathbf{v}_{ij}$ of the relative motion Σ^i/Σ^j is represented by $(\mathbf{c}_i - \mathbf{c}_j, \overline{\mathbf{c}}_i - \overline{\mathbf{c}}_j)$.

If both Σ^i/Σ and Σ^j/Σ are uniform motions with a Taylor expansion of the form (6), then the motion Σ^i/Σ^j is no longer uniform. Composition of these Taylor approximants and skipping the cubic and quartic terms results in the following second order approximant,

$$\begin{aligned}\mathbf{x}_{ij} = \mathbf{x} + \overline{\mathbf{c}}_i - \overline{\mathbf{c}}_j + (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{x} + \frac{1}{2}[(\mathbf{c}_i - \mathbf{c}_j) \times \overline{\mathbf{c}}_i + \mathbf{c}_j \times (\overline{\mathbf{c}}_j - \overline{\mathbf{c}}_i) \\ + \mathbf{c}_i \times (\mathbf{c}_i \times \mathbf{x}) + \mathbf{c}_j \times (\mathbf{c}_j \times \mathbf{x}) - 2\mathbf{c}_j \times (\mathbf{c}_i \times \mathbf{x})].\end{aligned} \tag{8}$$

2.4 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 $\overline{\mathbf{c}}$, 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 (3). 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 [3,27].

2.5 The group of Euclidean motions embedded in the affine group

If we do not impose orthogonality on the matrix A in equation (1), 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. Hence, we now describe a few simple facts that have been successfully used in various places, for example in a method for the transfer of curve design algorithms to the design of smooth rigid body motions [12].

In the following, we use a kinematic mapping 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 [12], 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* henceforth. 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,

$$\|\alpha - \beta\|^2 = \|\mathsf{A} - \mathsf{B}\|^2 := \sum_i [\alpha(\mathbf{x}_i^0) - \beta(\mathbf{x}_i^0)]^2. \tag{9}$$

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

$$\|\mathsf{A} - \mathsf{B}\|^2 = \|\mathsf{C}\|^2 = \sum_i [\mathbf{c}_0 + x_{i,1}^0\mathbf{c}_1 + x_{i,2}^0\mathbf{c}_2 + x_{i,3}^0\mathbf{c}_3]^2 =: \mathsf{C}^T \cdot M \cdot \mathsf{C}. \tag{10}$$

This expression with help of a positive definite symmetric matrix M immediately reveals the following facts [12]: The metric (9) 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.

By a well-known result from mechanics, we can replace the points $\mathbf{x}_1^0, \ldots, \mathbf{x}_N^0$ by the six vertices of the inertia ellipsoid without changing the barycenter and inertia tensor of X. To do so, we choose the barycenter as origin and the eigenvectors of J as coordinate axes in the moving system. Then, the six points have coordinates $(\pm f_1, 0, 0)$, $(0, \pm f_2, 0)$, $(0, 0, \pm f_3)$, where $2f_i^2$ are the eigenvalues of J. Now, the norm in $\mathbb{R}^{12}$ becomes

$$\|\mathsf{C}\|^2 = 6\mathbf{c}_0^2 + 2\sum_{i=1}^{3} f_i^2\mathbf{c}_i^2. \tag{11}$$

3 The squared distance function of a surface

Here we will summarize a few basic facts on the squared distance function. For more details and the derivation of these results, we refer to [25].

Given a surface $\Phi \subset \mathbb{R}^3$, we are interested in the squared distance function d^2, which assigns to each point $\mathbf{x} \in \mathbb{R}^3$ the square of its shortest distance to Φ. The importance of this function for an analysis of registration algorithms lies in the fact that we want to compute an optimal position of a data shape (usually a point cloud), which minimizes the sum of squared distances to a model shape that represents a surface Φ. Several important optimization concepts require second order approximants of the objective function. Thus, we have to study these approximants for the squared distance function d^2.

Consider a surface Φ with a 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 right-handed Cartesian system whose first two vectors $\mathbf{n}_1, \mathbf{n}_2$ determine the principal curvature directions. The latter are not uniquely determined at an umbilical point. There, we can take any two orthogonal tangent vectors $\mathbf{n}_1, \mathbf{n}_2$. We will refer to the thereby defined 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 $\rho_i = 1/\kappa_i$. Then, the two principal curvature centers at the considered surface point $\mathbf{s}$ are expressed in Π as $\mathbf{k}_i = (0, 0, \rho_i)$. The second order Taylor approximant F_d to the squared distance function d^2 at the point $\mathbf{p} = (0, 0, d)$ is the following [25].

Proposition 1 *The second order Taylor approximant of the squared distance function of a surface Φ at a point $\mathbf{p} \in \mathbb{R}^3$ is expressed in the principal frame at its normal foot point $\mathbf{s} \in \Phi$ via*

$$F_d(x_1, x_2, x_3) = \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2. \tag{12}$$

Let us look at two important special cases.

- For $d = 0$ we obtain
$$F_d(x_1, x_2, x_3) = x_3^2.$$
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 to the tangent plane at the closest point to the surface is a very good approximant.
- For $d = \infty$ we obtain
$$F_\infty(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2.$$

This is the squared distance to the foot point on the surface.

We see that distances to normal foot points are just good if we are in the 'far field' of the surface Φ. In the near field it is much better to use other local quadratic approximants. The simplest one is the squared distance to the tangent plane at the normal foot point.

For an implementation which employs the discussed approximants, it is better to express them in the same coordinate system as the surface itself. This is done by viewing F_d as a weighted sum of x_1^2 and x_2^2, the squared distances to the principal planes, and x_3^2, the squared distance to the tangent plane at the foot point $\mathbf{s}$. Thus, with $\mathbf{n}_i \cdot \mathbf{x} + d_i = 0$, $i = 1, 2, 3$, as Hesse normal forms of principal planes and tangent plane at $\mathbf{s}$, respectively, the quadratic approximant reads

$$F_d(\mathbf{x}) = \frac{d}{d-\rho_1}(\mathbf{n}_1 \cdot \mathbf{x} + d_1)^2 + \frac{d}{d-\rho_2}(\mathbf{n}_2 \cdot \mathbf{x} + d_2)^2 + (\mathbf{n}_3 \cdot \mathbf{x} + d_3)^2. \quad (13)$$

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 [25]. In any case, a second order approximant F_d is with appropriate coefficients α_1, α_2 and $\alpha_3 = 1$ given by

$$F_d(\mathbf{x}) = \sum_{j=1}^{3} \alpha_j (\mathbf{n}_j \cdot \mathbf{x} + d_j)^2. \quad (14)$$

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 given in the principal frame of the foot point as $\nabla d^2 = (0, 0, 2d)$. This follows e.g. from (12). In a global system the gradient is, up to the factor 2, the vector from the foot point $\mathbf{f}$ to $\mathbf{p}$. Hence, the gradient of the function $g = d^2/2$ is

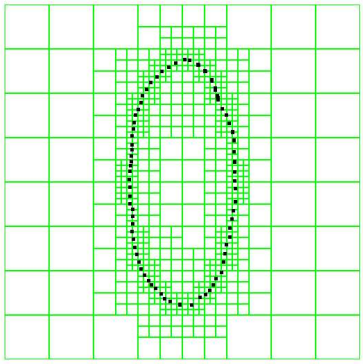
$$\nabla g = \mathbf{p} - \mathbf{f} = d\mathbf{n}. \quad (15)$$

Remark 2 *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*

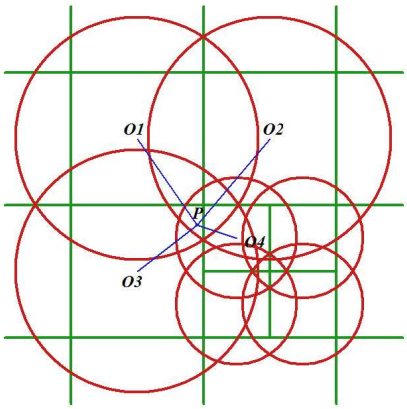
of points (vertices). The squared distance field of a point is quadratic anyway. Quadratic approximants to squared distance fields of space curves have been studied in [25]. Intuitively, the simplicity of the extension to piecewise smooth objects is explained as follows: we attach small smooth blending surfaces along edges and corners, with blending radius ε, and consider the limit for $\varepsilon \to 0$.

3.1 A data structure for fast distance information retrieval

In Computer Vision and 3D Photography, 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 foot 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 [20] 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.



(a) cells in a d2tree for two dimensional PCD



(b) Sketch to the construction of a modified d2tree in 2D; cells in red

Fig. 1. Modified d2tree structure

Simply put, the d2tree is a quadtree 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' [24], which is typically used to integrate locally defined approximants into a global approximation which preserves important properties, such as the maximum error and convergence order. In our approach, the quadratic patch in each cell C_i is associated with a C^2 compactly supported function $w_i(\cdot)$,

$$w_i(\mathbf{x}) = W(\frac{\|\mathbf{x} - \mathbf{o}_i\|^2}{1.7 * d_i^2}). \tag{16}$$

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 $[0, 1]$. In this article, we choose W to be a cubic B-spline basis function.

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

$$F_+(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x})(\mathbf{x}^T \cdot A_i \cdot \mathbf{x} + 2\mathbf{b}_i \cdot \mathbf{x} + c_i)}{\sum_i w_i(\mathbf{x})}. \tag{17}$$

The summation in equation (17) 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 (17) contribute to its squared distance approximation so that it can be fast 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 $\mathbf{o}_i, 1 \leq i \leq 4$.

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 [24,20] 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 $\mathbf{p}$,

$$F_2(\mathbf{x}) = F_+(\mathbf{p}) + \nabla F_+(\mathbf{p})^T \cdot (\mathbf{x} - \mathbf{p}) + \frac{1}{2}(\mathbf{x} - \mathbf{p})^T \cdot \nabla^2 F_+(\mathbf{p}) \cdot (\mathbf{x} - \mathbf{p}). \tag{18}$$

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

Remark 3 *Compared with the previous d2tree [20], 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 foot points in each iteration of the following registration algorithms.

4 Problem formulation and gradient descent

4.1 Formulation of registration as a constrained optimization problem

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 details on working with point cloud data, we refer to [23]. Additional issues which come up when only a part of the data shape agrees with a part of the model shape are handled in Sec. 8.

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

$$F(\alpha) = \sum_i d^2(\alpha(\mathbf{x}_i^0), \Phi). \tag{19}$$

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* [11,10,18].

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.

4.2 Gradient descent

In the following, we compute the gradient of the objective function F in (19). In view of (15), 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. With two vectors $(\mathbf{c}, \overline{\mathbf{c}})$ it is written as $\mathbf{v}(\mathbf{x}) = \overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. Let $\mathbf{y}_i$ be the foot 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 equation (15), the *directional derivative* of F in direction $\mathsf{C} = (\mathbf{c}, \overline{\mathbf{c}})$ reads

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

Here $\overline{\mathbf{f}}_i = \mathbf{x}_i \times \mathbf{f}_i$ is the moment vector of the surface normal through $\mathbf{x}_i$. It is appropriate to view the vectors $\mathbf{f}_i$ as forces acting along the corresponding surface normals. We call these forces the *repelling forces*. Then, $\overline{\mathbf{f}}_i$ are the moments of these forces. Altogether, we have a *repelling force system*, represented in terms of screw theory ([27], pp. 192) by the screw

$$\mathsf{F} = (\mathbf{f}, \overline{\mathbf{f}}) = (\sum_i \mathbf{f}_i, \sum_i \overline{\mathbf{f}}_i). \tag{20}$$

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*,

$$\frac{\partial F}{\partial \mathsf{C}} = \mathbf{f} \cdot \overline{\mathbf{c}} + \overline{\mathbf{f}} \cdot \mathbf{c}. \tag{21}$$

With known results from line geometry and screw theory [27] we conclude: *An instantaneous motion with directional derivative zero corresponds to a screw which is reciprocal to the screw* 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 4 *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 5 *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 [9,14]).*

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

$$\nabla F = (\bar{\mathbf{f}}, \mathbf{f}) =: \mathsf{F}^*. \tag{22}$$

It is more natural, however, to use the Euclidean metric (9) for normalization of the tangent vector to M^6, represented by C. This requires that we normalize according to

$$\sum_i (\overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2 = 1. \tag{23}$$

On the left hand side we have a positive definite quadratic form, so that the normalization can be written as

$$\mathsf{C}^T \cdot M_e \cdot \mathsf{C} = 1. \tag{24}$$

Writing the directional derivative in the form $\partial F / \partial \mathsf{C} = \mathsf{F}^{*T} \cdot M_e^{-1} \cdot M_e \cdot \mathsf{C}$, the gradient $\nabla_e F$ of F for the normalization (23) induced by the Euclidean metric (9) is deduced as

$$\nabla_e F = M_e^{-1} \cdot \nabla F = M_e^{-1} \cdot \mathsf{F}^*. \tag{25}$$

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 subsection 2.4, 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 [18] or a more sophisticated step size prediction scheme of optimization [10,18].

Remark 6 *The gradient according to (25) 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*

$$F_1 = \sum (\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i) - \mathbf{y}_i)^2 = \sum_i (\mathbf{f}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)^2. \tag{26}$$

With the expression of (23) in the form (24), and with help of (22) and (20), function F_1 *reads in matrix notation*

$$F_1 = \mathsf{C}^T \cdot M_e \cdot \mathsf{C} + 2(\mathsf{F}^*)^T \cdot \mathsf{C} + \sum_i \mathbf{f}_i^2. \tag{27}$$

Therefore, the minimizer C_m *is given by the negative gradient from equation (25),*

$$\mathsf{C}_m = -M_e^{-1} \cdot \mathsf{F}^* = -\nabla_e F. \tag{28}$$

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 Sec. 5. There, we show that ICP is linearly convergent. The same holds for a gradient descent, if one uses an appropriate step size [18].*

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 Sec. 6.

5 The ICP algorithm revisited

The most widely used algorithm for the solution of the registration problem is the *iterative closest point (ICP) algorithm* of P. Besl and N.D. McKay [2]. 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

$$F_1 = \sum_i \|\mathbf{x}_{i+} - \mathbf{y}_i\|^2. \tag{29}$$

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 5). 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 [9,14]. The solution eigenvector is nothing but the unit quaternion description of the rotational part of α_+.

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. The ICP algorithm always converges monotonically to a local minimum, since the value of the objective function is decreasing in each iteration.

5.1 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

$$\|\alpha_+ - \alpha^*\| \leq C\|\alpha_c - \alpha^*\|, \tag{30}$$

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

Let us now proceed with a proof of the error formula (30). 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 d of the embedding space $D = \mathbb{R}^d$; since only the second order Taylor expansions of $\mathbf{c}$ and $\mathbf{f}$ around $\mathbf{c}^*$ and $\mathbf{f}^*$ enter the discussion, dimension $d = 5$ is actually sufficient. Moreover, is suffices to express orthogonality in $\mathbb{R}^d$ 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

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

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

$$(\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{c}_0' = 0, \quad (\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{f}_0' = 0. \tag{31}$$

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

$$(\mathbf{c}(u_c) - \mathbf{f}(v_c)) \cdot \mathbf{f}'(v_c) = 0, \quad (\mathbf{c}(u_+) - \mathbf{f}(v_c)) \cdot \mathbf{c}'(u_+) = 0. \tag{32}$$

Now we insert the Taylor expansions into these two equations. The absolute terms cancel because of (31). 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

$$u_+ = Cu_c, \quad \text{with} \quad C = \frac{(\mathbf{c}_0' \cdot \mathbf{f}_0')^2}{[\mathbf{c}_0'^{\,2} + (\mathbf{c}^* - \mathbf{f}^*) \cdot \mathbf{c}_0''][\mathbf{f}_0'^{\,2} + (\mathbf{f}^* - \mathbf{c}^*) \cdot \mathbf{f}_0'']}.$$

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,

$$\cos\phi = \mathbf{c}_0' \cdot \mathbf{f}_0'.$$

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,

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

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

$$C = \frac{\cos^2\phi}{(1 - d\kappa_c^n)(1 - d\kappa_f^n)}. \tag{33}$$

Remark 7 *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 [5,30]. 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 (30) with the constant from (33).

5.2 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 equation (33), 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 equation (9) 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,

$$\|X - X^*\|^2 = \|\alpha - \alpha^*\|^2 = \sum_i (\mathbf{x}_i - \mathbf{x}_i^*)^2. \tag{34}$$

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

$$\cos\phi \approx \frac{\sum_i (\mathbf{x}_i - \mathbf{x}_i^*) \cdot (\mathbf{y}_i - \mathbf{y}_i^*)}{\sqrt{\sum_i (\mathbf{x}_i - \mathbf{x}_i^*)^2} \sqrt{\sum_i (\mathbf{y}_i - \mathbf{y}_i^*)^2}}. \tag{35}$$

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

$$\cos\phi \approx \frac{\sum_i (\mathbf{x}_{ic} - \mathbf{x}_{i+}) \cdot (\mathbf{y}_{ic} - \mathbf{y}_{i+})}{\sqrt{\sum_i (\mathbf{x}_{ic} - \mathbf{x}_{i+})^2} \sqrt{\sum_i (\mathbf{y}_{ic} - \mathbf{y}_{i+})^2}}. \tag{36}$$

This confirms an intuitively obvious and experimentally verified phenomenon: *ICP is very slow, if tangential moves along the surface are needed. Then the*

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				

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 present here an example to empirically test the accuracy of the estimate (36) of the constant in the linear convergence behavior of ICP. 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 (34), the estimate $\cos^2\phi$ of the constant C using formula (36) and the quotient $E(j)/E(j-1)$, which represents the exact error reduction in each iteration. The last two quantities are graphed in Figure 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 [27]; strategies for handling them in an ICP algorithm have been described by Gelfand et al. [8,16].

Let us summarize the results on the convergence of ICP.

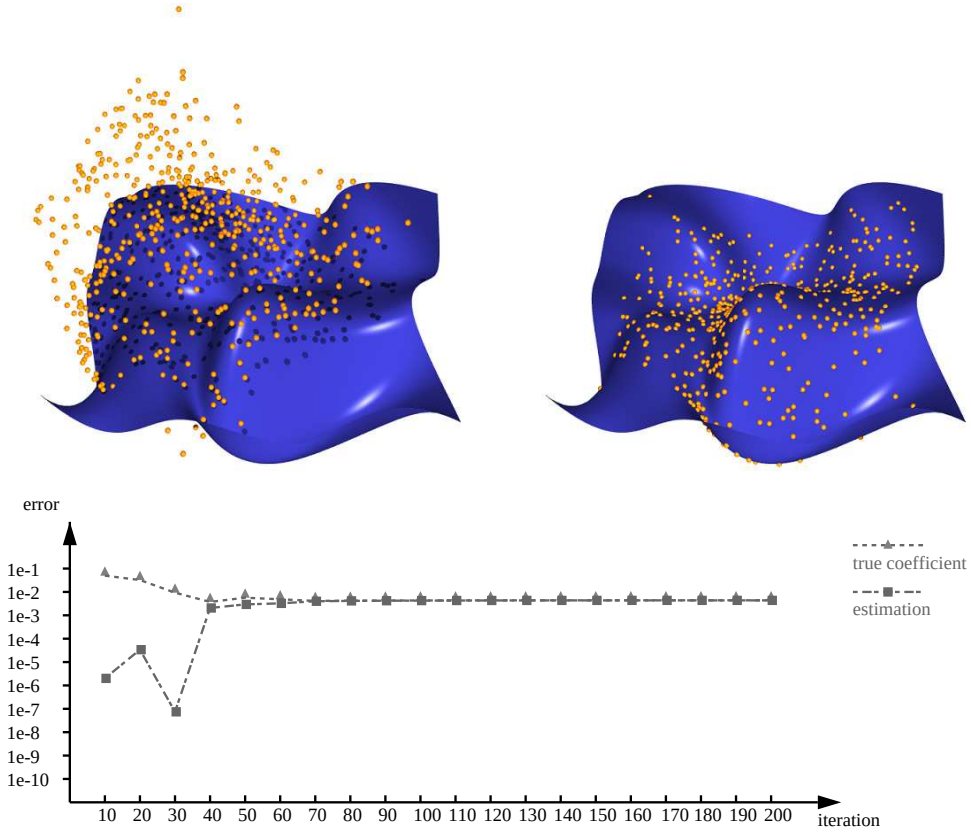


Fig. 2. Local convergence behavior of the standard ICP algorithm: (upper left) initial position, (upper right) final position, (bottom) illustration of the constant C for the linear error reduction, and its estimate $\cos^2 \phi$ according to (36)

Proposition 8 *The ICP algorithm exhibits in general linear convergence with a decay constant C given by equation (33). 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 ICP does. However, they are no longer quadratically convergent for $\phi = 0$.

6 Quadratically convergent registration algorithms

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

form

$$\|\alpha_+ - \alpha^*\| \leq C\|\alpha_c - \alpha^*\|^2, \tag{37}$$

with some positive constant C. We will describe several quadratically convergent algorithms, all of them based on a Newton type iteration:

(1) The most universally usable algorithm carefully computes second order approximants to the squared distance function of Φ and to the constraint set, given by the rigidity of the moving system.
(2) A simplified version of the former algorithm just uses a linearization of the motion. This is justified by the fact that the minimizer we are looking for may also be a minimizer within the affine group. Especially if the deviations between data point set and model shape are very small, this is true. In such cases, the algorithm still converges quadratically.
(3) If the model shape does not allow affine transformations in itself and the deviation is close to zero, the desired minimizer is also an isolated minimizer within the affine group. This means that we can remove the rigidity constraint and just work with a Newton algorithm in the affine group.

We start the discussion with the simplest version, namely (3), in subsection 6.1. Method (2) in studied in subsection 6.2, and finally the most general, but computationally also most intensive algorithm (1) is described in subsection 6.3. All our 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 [23]). This geometric insight is lacking in a paper by Tucker and Kurfess [31], which applies the Newton method to registration in a straightforward way and thus leads to quite involved expressions and little possibilities for acceleration.

Before we enter the discussion of registration, let us recall the most basic facts on Newton iteration [18]. A Newton method for the minimization of a function $F(x)$ solves $\nabla F = 0$ iteratively via linearization. Equivalently, it computes a second order Taylor approximant at the current position x_c and minimizes this quadratic function to obtain the next iterate x_+. Therefore, with the gradient $\nabla F(x_c)$ and the Hessian $\nabla^2 F(x_c)$, one has

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

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 [18].

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

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

has sufficient descent [18]. In the following, we will not explicitly point to this stabilization, but we are assuming it is done.

6.1 Registration with a Newton algorithm in the affine group

The algorithm we are dealing with assumes that 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 self-motions of the model shape. This is not always the case: for example, in case of an ellipsoid as model shape we have even a 3-parameter group of affine automorphisms. However, in many practical situations, such affine self-transformations will not be possible and for such cases we propose to proceed as follows.

Starting from an appropriate initial position α^0, we perform a Newton iteration in $\mathbb{R}^{12}$ for the minimization of F. A Newton method requires a second order approximation of the objective function F. Since F is the sum of squared distances of the data points $\mathbf{x}_i$ to the model shape Φ, a second order approximant is

$$F_2 = \sum_i F_{d,i}, \tag{38}$$

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 3. Let $\mathbf{n}_{i,j} \cdot \mathbf{x} + d_{i,j} = 0$, $j = 1, 2, 3$, be the Hesse normal forms of the coordinate planes of the principal frame at the foot point $\mathbf{y}_i$. Then, by equation (14), a second order Taylor approximant of the squared distance function at $\mathbf{x}_i$ is written as

$$F_{d,i}(\mathbf{x}) = \sum_{j=1}^{3} \alpha_{i,j} (\mathbf{n}_{i,j} \cdot \mathbf{x} + d_{i,j})^2. \tag{39}$$

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,

$$\mathbf{x}_i' = \mathbf{x}_i + \mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3, \tag{40}$$

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

$$F_2 = \sum_i \sum_{j=1}^{3} \alpha_{i,j}[\mathbf{n}_{i,j} \cdot (\mathbf{x}_i + \mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3) + d_{i,j}]^2. \tag{41}$$

Since $\mathbf{n}_{i,j} \cdot \mathbf{x}_i + d_{i,j}$ is the distance of $\mathbf{x}_i$ to the j-th 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

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j}[\mathbf{n}_{i,j} \cdot (\mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3)]^2 + \tilde{F}_2. \tag{42}$$

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

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\mathbf{c}_0 + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3) + d_i]^2. \tag{43}$$

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 go sure 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 one, 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 9 *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.*

We are now keeping the rigidity constraint. In other words, our path in $\mathbb{R}^{12}$ towards the minimum is restricted to M^6. Let us first explain our iterative procedure in $\mathbb{R}^{12}$. Here, each iteration from α^k to α^{k+1} consists of the following two steps.

(1) Compute the tangent space T^6 of M^6 at α^k 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^{k+1}.

Such a procedure needs not even be convergent if the unconstrained minimum (in $\mathbb{R}^{12}$) is far away from M^6. However, *if the minimum 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. [13]).

The realization of the two steps in the algorithm outlined above is as follows.

Step 1. The tangent space T^6 is defined by Euclidean velocity fields, i.e. $\mathbf{v}(\mathbf{x}) = \overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. Equivalently, $\mathbf{c}_i$ of (40) 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 following quadratic function in $(\mathbf{c}, \overline{\mathbf{c}})$,

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i)]^2 + \tilde{F}_2. \tag{44}$$

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

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) + d_i]^2 = \sum_i [\mathbf{n}_i \cdot \overline{\mathbf{c}} + \overline{\mathbf{n}}_i \cdot \mathbf{c} + d_i]^2, \tag{45}$$

and can be used instead of F_2 when we are already close to the model shape (see also Sec. 7). Note that $(\mathbf{n}_i, \overline{\mathbf{n}}_i)$) are the Plücker coordinates of the surface normal through $\mathbf{x}_i$. F_2 is a quadratic function in the unknowns $(\mathbf{c}, \overline{\mathbf{c}})$. The unique solution $(\mathbf{c}^*, \overline{\mathbf{c}}^*)$ can be given explicitly by solving a system of linear equations.

Step 2. The projection back to M^6 proceeds according to subsection 2.4. We apply a helical motion which is determined by the velocity field $(\mathbf{c}^*, \overline{\mathbf{c}}^*)$.

Let us remark that 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 [18] for the determination of an appropriate step size, analogous to the procedure in a gradient descent algorithm. It is well known in optimization [18] that this results in an algorithm with *linear convergence*.

6.3 A Newton algorithm based on a second order motion approximant

To achieve quadratic convergence in any case, we can use a second order approximant for the motion from α^k to α^{k+1} according to (6). This means that we estimate the displaced data point $\mathbf{x}_i$ by

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

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,

$$F_2 = \sum_i \sum_{j=1}^{2} \alpha_{i,j}[\mathbf{n}_{i,j} \cdot (\overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i + T_{i,2})]^2 + \tilde{F}_2. \tag{46}$$

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 the current position, i.e. $(\mathbf{c}, \overline{\mathbf{c}}) = (0, 0)$, these terms do not matter at all. However, we have to look into $\tilde{F}_2$,

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i + T_{i,2}) + d_i]^2, \tag{47}$$

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

$$\tilde{F}_2' = \sum_i [\mathbf{n}_i \cdot \overline{\mathbf{c}} + \overline{\mathbf{n}}_i \cdot \mathbf{c} + d_i]^2 + 2\sum_i d_i \mathbf{n}_i \cdot T_{i,2}. \tag{48}$$

Hence, the only relevant correction term compared to the use of a linearized motion as in subsection 6.2 is

$$F_{2c} = \sum_i d_i[\det(\mathbf{n}_i, \mathbf{c}, \overline{\mathbf{c}}) + (\mathbf{c} \cdot \mathbf{x}_i)(\mathbf{c} \cdot \mathbf{n}_i) - \mathbf{c}^2(\mathbf{x}_i \cdot \mathbf{n}_i)]. \tag{49}$$

Computationally, the second order motion approximation does not require much more effort. However, in this refined version we can guarantee quadratic convergence, if we have an initial position in the region of attraction of the minimum.

6.4 Examples and comparison

In this subsection, we want to compare the standard ICP algorithm and the three algorithms presented in subsections 6.1, 6.2 and 6.3 from the perspective of global stability and local convergence. Two examples are performed, corresponding to problems with a zero and a large residual, respectively.

In our first example, the object is a fender model, which is represented as a bi-cubic B-spline surface. The size of the object is approximately $2.023 \times 0.750 \times 0.367$. We sampled the model at $k = 500$ points, without adding noise.

The first test is devoted to analyze the *local convergence behavior*. Figure 3 shows the initial position and final position with respect to the model.

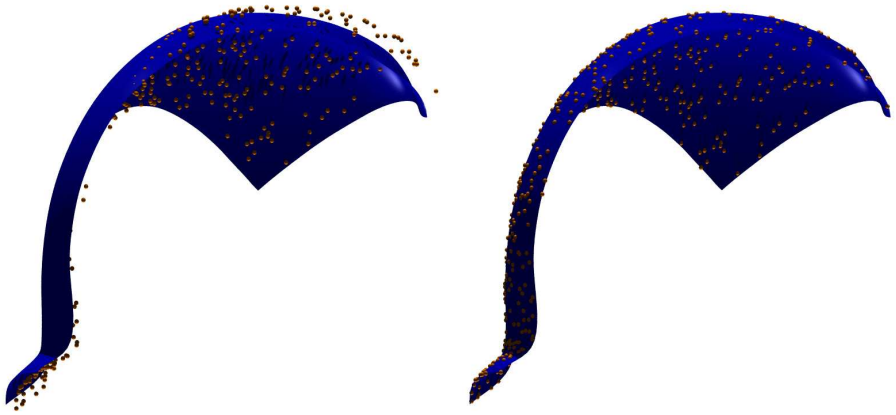


Fig. 3. Local convergence test: (left) initial position of data set and model shape, (right) final position.

In Table 2, the errors $E(j)$ of our algorithms and of the standard ICP algorithm are given. Among the four tested algorithms, affine registration is the fastest. Moreover, our algorithms stop after several iterations with the error below $1e-13$, whereas ICP still has $E(200) = 1.996e-11$ after 200 iterations. Furthermore the columns identified by $E(j)/E(j-1)^2$ reveal that

Table 2: Error reduction for a zero residual problem; local convergence test

	Newton, 1st order motion			**Newton, 2nd order motion**		
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$
0	2.942e-2	—	—	2.942e-2	—	—
1	3.303e-3	1.122e-1	3.816	3.671e-3	0.1249	4.241
2	8.899e-5	2.694e-2	8.158	8.863e-5	8.761e-2	6.577
3	4.935e-8	5.546e-4	6.232	2.258e-8	5.317e-2	2.875
4	1.794e-14	3.635e-7	7.367	1.570e-15	1.680e-3	3.079
5	<1.0e-16	—	—	<1.0e-16	—	—

	affine registration			**standard ICP**		
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$
0	2.942e-2	—	—	2.942e-2	—	—
1	4.540e-3	0.1543	5.245	1.109e-2	0.3769	—
2	9.067e-5	1.997e-2	4.398	7.569e-3	0.6825	—
3	2.324e-8	2.563e-4	2.827	5.335e-3	0.7084	—
4	6.200e-16	2.668e-8	1.148	3.904e-3	0.7318	—
⋮						
200	—	—	—	1.996e-11	0.9131	—

our algorithms are quadratically convergent, while ICP and shows only linear convergence.

After examining the local convergence, we come to the second test which focuses on *global stability*. In this case, affine registration tends shrink the point cloud converge to a point and thus we do not include it in this test. To ensure global convergence (i.e., convergence to a local minimizer from any initial position), all three tested algorithms are regularized by the Armijo rule. Figure 4 shows the initial position of the point cloud X and the progress made by each tested algorithm. The latter is illustrated with help of the positions of the barycenter and the vertices of the inertia ellipsoid of X. Moveover, each figure shows on the left hand side the panoramic view; on the right hand side it zooms into the situation near the minimizer.

In Table 3, the errors $E(j)$ of our algorithms and the standard ICP algorithm are given. Our algorithms stop after several iterations with the error below $1e-16$, whereas ICP still has $E(200) = 2.996e-10$ after 200 iterations. Moveover, compared with the second order Newton method, the first order Newton method appears to be more stable.

Now we can summarize the *behavior of our algorithms for zero residual problems*. We find that affine registration exhibits the fastest local convergence behavior. However, in other examples we verified an expected phenomenon: affine registration can lead to severe shrinking effects, since there are many undesirable local minima corresponding to positions, where the entire data

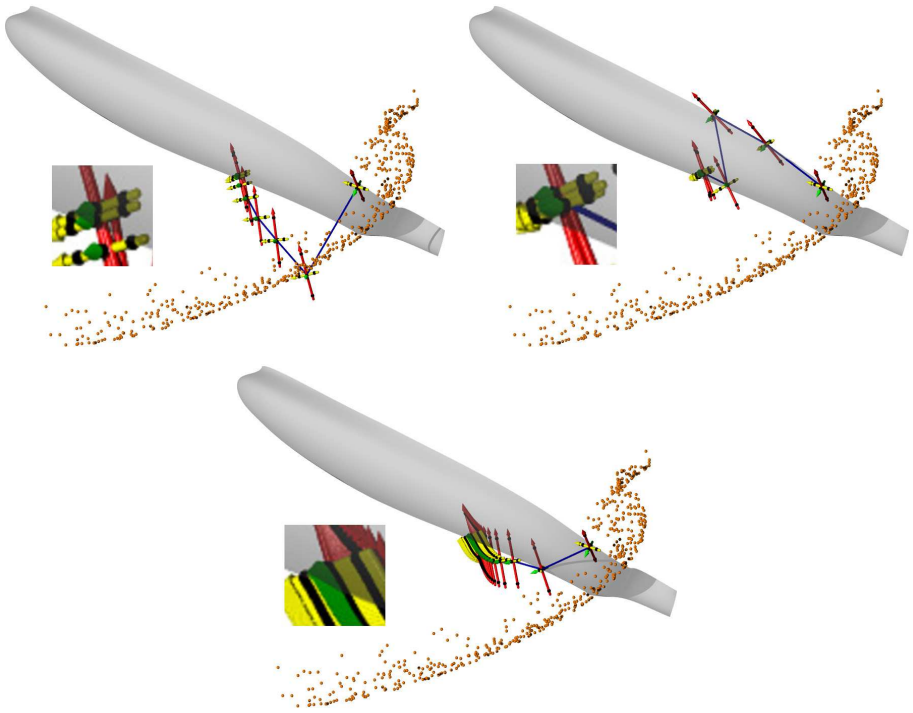


Fig. 4. Comparison of our algorithms with standard ICP, visualized by the motion of barycenter and inertia axes of the data point set in a panoramic view and a zoom into the region near the final position: Newton algorithm with (upper left) first order motion approximation, and (upper right) second order motion approximation, respectively; (bottom) standard ICP.

Table 3: Error reduction for a zero residual problem; global stability test

	Newton, 1st order			**Newton, 2nd order**			**ICP**	
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$
0	8.832e-1	—	—	8.832e-1	—	—	8.832e-1	—
1	6.013e-1	0.6908	0.7709	4.065e-1	0.4602	0.5211	4.207e-1	0.4763
2	3.397e-1	0.5549	0.9395	4.028e-1	0.9908	2.438	2.434e-1	0.5785
3	2.232e-1	0.6570	1.934	1.834e-1	0.4553	1.130	1.846e-1	0.7584
4	1.265e-1	0.5567	2.539	6.437e-2	0.3509	1.914	1.564e-1	0.8472
5	6.078e-2	0.4804	3.798	3.283e-2	0.5100	7.923	1.397e-1	0.8932
6	1.749e-2	0.2877	4.734	9.475e-3	0.2886	8.791	1.298e-1	0.8854
7	1.181e-3	6.752e-2	3.861	7.792e-4	8.223e-2	8.679	1.112e-1	0.8989
8	4.468e-5	3.783e-2	32.03	1.392e-5	1.786e-2	22.92	9.876e-2	0.8881
9	1.058e-8	2.368e-4	5.298	8.267e-10	1.629e-5	4.266	8.789e-2	0.8899
10	6.300e-16	5.955e-8	5.628	<1e-16	—	—	7.867e-2	0.8950
⋮								
200	—	—	—	—	—	—	2.996e-10	0.9131

set shrinks to a point on the model surface. From a stability perspective, the Newton algorithm with a first order motion approximation and a step size control seems to be superior to other algorithms.

After testing and analyzing a zero-residual problem, we come to a *large residual problem*. With an application in 3D face recognition in mind (see also [26]), we randomly choose 1000 noisy points from a head model. The point data should be aligned with the model shape, a 'generic' face given as a triangular mesh. The goal of this application is to bring the point data (measurement data) of the face into a standard position such that further processing steps can be applied to the data. In this application of face registration, we use the modified d2tree structure to compute the distance information of the model shape a priori. It is appropriate since many different face data sets might be registered with one fixed model shape.

One should actually take the variation in head size into account and register by a similarity transformation, composed of uniform scaling and a rigid body transformation. This requires only minor modifications in the algorithms presented above (see [26]). In our test, we did not include the scaling, since we wanted to test a large residual problem anyway.

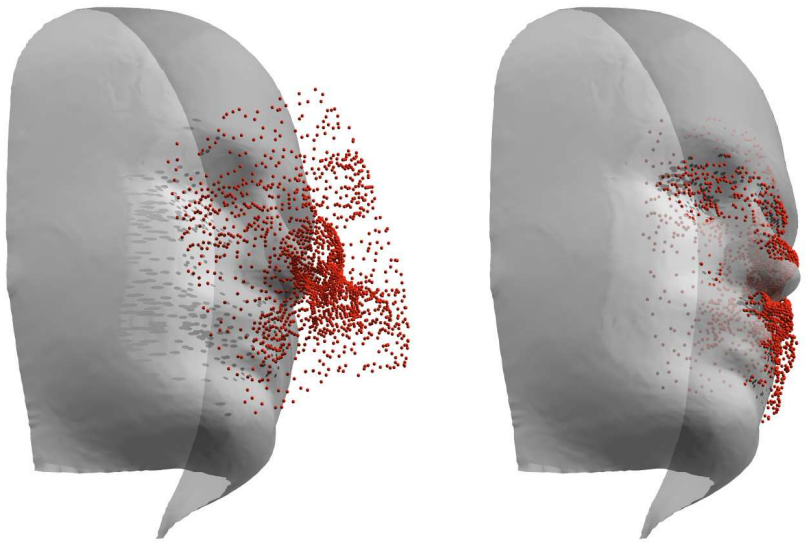


Fig. 5. A registration problem with a large residual: (left) initial position of the data set with respect to the head model, (right) final position.

We test the Newton algorithms with first and second order motion approximants, respectively, as well as the standard ICP. Figure 5 shows initial and final position of the data set in view of the 'generic' face. Besides, to make our experimental data clear, we scale coordinates of our system such that the data set is approximately $0.334 \times 0.516 \times 0.246$ units in size.

In Table 4, the errors $E(j)$ of our algorithms and of the standard ICP al-

Table 4: Error reduction in a large residual problem

	Newton, 2nd order			Newton, 1st order			ICP	
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$
0	7.238e-2	—	—	7.238e-2	—	—	7.238e-2	—
1	1.964e-2	2.713e-1	3.749	3.039e-2	0.4198	6.860	4.505e-2	0.6224
2	7.817e-3	3.980e-1	20.265	1.335e-2	0.4393	1.160e+1	3.462e-2	0.7685
3	2.195e-3	2.807e-1	35.921	2.926e-3	0.2198	1.517e+1	3.078e-2	0.8870
4	9.238e-4	4.208e-1	191.738	4.574e-4	0.1563	5.192e+1	2.736e-2	0.8832
5	4.286e-5	4.639e-2	50.222	7.837e-5	0.1713	3.161e+2	2.434e-2	0.8874
6	1.514e-7	3.532e-2	82.418	1.319e-5	0.1683	1.992e+3	2.160e-2	0.8872
7	2.718e-13	1.795e-6	11.858	2.077e-6	0.1575	1.232e+4	1.922e-2	0.8880
8	—	—	—	3.272e-7	0.1578	7.585e+4	1.708e-2	0.8890
⋮								
13	—	—	—	3.071e-11	0.1579	8.078e+8	9.490e-3	0.8890
14	—	—	—	4.907e-12	0.1598	5.204e+9	8.436e-3	0.889
15	—	—	—	7.842e-13	0.1597	3.252e+10	7.499e-3	0.8897
⋮								
200	—	—	—	—	—	—	3.102e-12	0.8894

gorithm are given. Our algorithm stops after several iterations with an error below $E = 1.0e-12$, whereas ICP still has $E(200) = 3.102e-12$ after 200 iterations. Moreover, when comparing our algorithms, we see that the quotient $E(j)/E(j-1)^2$ for the second order algorithm is bounded, while the corresponding quotient Newton algorithm with first order motion approximation tends to infinity as j increases; besides, the quotient $E(j)/E(j-1)$ is approximately constant in the first order Newton algorithm. All of this numerical evidence reveals that the Newton algorithm with second order motion approximant is quadratically convergent for large residual problems, whereas first order motion approximation in the Newton algorithm exhibits linear convergence only. Last but not least, the same quotient $E(j)/E(j-1)$ is much larger for the standard ICP algorithm.

7 Gauss–Newton iteration and the Levenberg–Marquardt regularization

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 foot 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. Minimization of squared tangent plane distances has been first proposed by Chen and Medioni [4], and it has been observed in various papers that this results in faster convergence than ICP. From a geometric point of view this is explained as follows (see [25]): Squared tangent plane distances are in the near field of the model shape close to second order approximants of the squared distance function. However, squared distances to foot points as employed in ICP are far away from second order precision; in fact they are second order approximants at

infinity. Hence, the original formulation of registration by Chen and Medioni performs much better for fine registration than ICP.

7.1 Some basics on Gauss–Newton iteration

We would like to understand registration based on squared tangent plane distances from the viewpoint of optimization, and for that, we recall a few basic facts [18]. Our objective function F in equation (19) is a sum $(1/2)\sum_i d_i^2$ of squares. One speaks of a *nonlinear least squares problem* [18]. For a nonlinear least squares problem, the Hessian is $\nabla^2 F = \sum_i \nabla d_i \cdot (\nabla d_i)^T + \sum_i d_i \nabla^2 d_i$. Often, the computation of the Hessians $\nabla^2 d_i$ is too costly and thus one works with the first part only, considered as an approximation to the true Hessian. A step in the resulting *Gauss-Newton* iteration is equivalent to the solution of the linear least squares problem

$$\min \sum_{i=1}^{N} [d_i(x_c) + \nabla d_i(x_c) \cdot (x - x_c)]^2. \tag{50}$$

It is well-known ([18],pp. 24) that the distance $\|e_c\| = \|x_c - x^*\|$ of the current iterate to the minimizer x^* is related to the error $\|e_+\|$ in the next iterate by

$$\|e_+\| \leq K(\|e_c\|^2 + \|R(x^*)\| \, \|e_c\|). \tag{51}$$

Here, $R(x^*) = (d_1, \ldots, d_N)(x^*)$ is the residual at the minimizer, and K is an appropriate constant which involves the Jacobian of $R(x)$. The error estimate is only true, if one is sufficiently close to the minimum. The well-known conclusions of (51) 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.

7.2 Registration with a Gauss-Newton algorithm

We have to formulate equation (50) 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 ∇d_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 its foot 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 $\nabla d_i(x_c) \cdot (x - x_c)$ describes, in our case, the directional derivative of this distance for an affine displacement of $\mathbf{x}_i$, which equals

$$\mathbf{n}_i \cdot (\overline{\mathbf{c}} + x_{i,1}\mathbf{c}_1 + x_{i,2}\mathbf{c}_2 + x_{i,3}\mathbf{c}_3).$$

Therefore, the minimization of squared tangent plane distances, which is described in (43), is identical to a Gauss–Newton iteration (50).

Adding the rigidity constraint has been discussed in Sec. 6; in exactly the same way we can handle the constraint for Gauss–Newton iteration. 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 in 7.1.

Proposition 10 *Registration algorithms, which are based on the Chen & Medioni approach [4] and iteratively minimize the sum of squared distances to tangent planes at the normal foot points $\mathbf{y}_i$ 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, we see that the Chen & Medioni method works well for small residual problems. However, there is no reason to expect convergence for a large residual problem.

7.3 Regularization according to the Levenberg–Marquardt method

Optimization theory provides several methods to achieve convergence of Gauss–Newton like iterations even for large residual problems [18].

A variant of the Gauss–Newton iteration does not apply the full step $x_+ - x_c$, but just a scalar multiple $\lambda(x_+ - x_c)$, usually with $\lambda < 1$, to the current iteration. Various methods for a line search along $x_c + \lambda(x_+ - x_c)$ can be applied (see [18]). This results in a so-called *damped Gauss–Newton algorithm*.

Another way to modify Gauss–Newton is a regularization with the *Levenberg–Marquardt method* [18]. Here, a scalar multiple of the unit matrix is added to the approximate Hessian. This method, applied to the Gauss-Newton iteration with a first order motion approximant, requires iterative minimization of

$$\tilde{F}_2 = \sum_i [\mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) + d_i]^2 + \nu(\bar{\mathbf{c}}^2 + \mathbf{c}^2). \tag{52}$$

For the choice of the parameter ν, we refer to [18].

We present here an example to test the effect of Levenberg–Marquardt regularization on registration with a Gauss-Newton algorithm and first order motion approximation. The chosen object is a mechanical model, which is represented as a triangular mesh (see Fig. 6). The size of the object is approximately $0.858 \times 1.542 \times 2$ units. Data points are obtained by sampling, without adding Gaussian noise. We compare the first order Newton algorithm with Armijo rule and Levenberg-Marquardt regularization, respectively. The behavior of the two regularization methods is graphed in Figure 6. One should keep in mind that both the Armijo rule and Levenberg–Marquardt regularization are aiming at global stabilization, and thus we only present the data from a few initial steps.

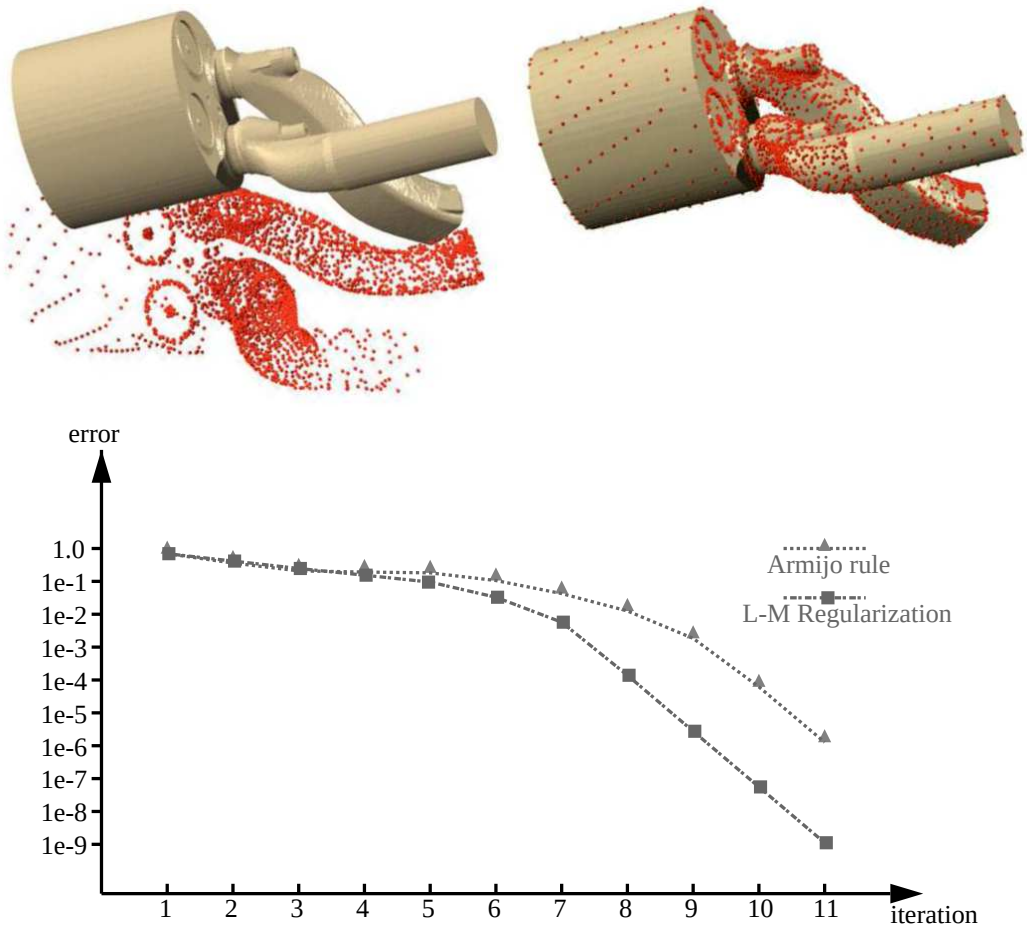


Fig. 6. Comparing Levenberg-Marquardt regularization with the Armijo rule: (upper left) initial position, (upper right) final position, (bottom) error decay

In our example, Levenberg-Marquardt regularization is faster than the Armijo rule, except for a few initial steps. We have also tested different initial positions. In some cases, the Armjio rule method converges to a local minimizer while Levenberg-Marquardt still reaches the global minimizer. Comparing the

two methods, we find that Levenberg-Marquardt regularization exhibits a behavior of global stability.

8 Simultaneous registration of multiple views without correspondences

We are now addressing the following registration problem: Surface patches S^i, $i = 1, \ldots, N$, which define rigid body systems Σ^i, shall be assembled to a single surface assuming that each patch has some overlap with at least one adjacent patch. Although we speak of patches, these could be point cloud data; for details on handling point cloud data, we refer to [23]. Let us point out that we are interested here in the *simultaneous registration* of the given N systems in contrast to the major part of the literature, which is solving this problem by pairwise registration (see e.g. [22] and the references therein).

We assume that we have some roughly correct initial positions S^i with information on a possible overlap. The confidence value of an overlap between patches S^i and S^j shall be $\gamma_{ij} \in [0, 1]$. The objective function we are going to minimize is

$$F(\alpha_1, \ldots, \alpha_N) = \sum_{i,j=1}^{N} \gamma_{ij} d^2(\alpha_i(S^i), \alpha_j(S^j)). \tag{53}$$

Here, $d^2(.,.)$ denotes a distance measure between two patch positions which will be defined in the following subsection. Of course, α_i are the rigid body transforms which map the initial patch positions S^i to their final location. One of the patches, say S^1 needs to be fixed. Thus, α_1 is the identity.

8.1 Distance of two patches in the overlapping region

We define a distance measure between two patch positions $S_i := \alpha_i(S^i)$ and S_j, which is active only in the overlapping region. Our distance will in general not be symmetric, $d^2(S_i, S_j) \neq d^2(S_j, S_i)$; however, both distances appear in the objective function (53), which finally gives symmetry. Let us randomly choose M points $\mathbf{x}_{ijk}$ on S_i, and let $\mathbf{y}_{ijk}$ be their closest points on S_j. Each pair gets a weight

$$w_{ijk} := \begin{cases} 1 & \text{if } \|\mathbf{x}_{ijk} - \mathbf{y}_{ijk}\|^2 \leq T \\ \exp[-\tau(\|\mathbf{x}_{ijk} - \mathbf{y}_{ijk}\|^2 - T)^2] & \text{if } \|\mathbf{x}_{ijk} - \mathbf{y}_{ijk}\|^2 > T \end{cases} \tag{54}$$

Thus, overlap detection is accomplished by setting some threshold T on the squared distance. It helps to confine the overlap region. T needs 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. For each patch pair (S_i, S_j), we can use another number M_{ij} of data points, depending on the size of the overlapping region. To keep the notation simple, we will disregard this possibility in the following.

The patch distance can now be defined as

$$d^2(S_i, S_j) := \sum_{k=1}^{M} w_{ijk} d^2(\mathbf{x}_{ijk}, S_j). \tag{55}$$

This is very similar to (19) and therefore the basic results can be carried over to the present setting. We use quadratic approximants of the squared distance fields of the patches S_j and can formulate Newton-type algorithms as before. Even the convergence results are the same. In order to show how relative kinematics enters the considerations, we outline this at hand of a Newton algorithm with linearized motions.

8.2 A Newton algorithm with first order motion approximants for the simultaneous registration of N systems

For the kinematical analysis, we consider $\Sigma^1 =: \Sigma$ fixed and view the motions of the other systems Σ^i against Σ. The transition from the current position of a system Σ^i to the next position will be estimated by the velocity field $\mathbf{v}_i$ described by $(\mathbf{c}_i, \overline{\mathbf{c}}_i)$. In particular, we have $(\mathbf{c}_1, \overline{\mathbf{c}}_1) = (0, 0)$. Thus, in each iteration, we will solve for $6(N-1)$ unknowns $(\mathbf{c}_i, \overline{\mathbf{c}}_i)$, $i = 2, \ldots, N$ in the local quadratic model

$$\begin{aligned} F_2 = \sum_{i,j=1}^{N} \sum_{k=1}^{M} \gamma_{ij} w_{ijk} [\alpha^1_{ijk} [\mathbf{n}^1_{ijk} \cdot (\overline{\mathbf{c}}_i - \overline{\mathbf{c}}_j + (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{x}_{ijk})]^2 + \\ \alpha^2_{ijk} [\mathbf{n}^2_{ijk} \cdot (\overline{\mathbf{c}}_i - \overline{\mathbf{c}}_j + (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{x}_{ijk})]^2] + \tilde{F}_2. \end{aligned} \tag{56}$$

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

$$\tilde{F}_2 = \sum_{i,j=1}^{N} \sum_{k=1}^{M} \gamma_{ij} w_{ijk} [\mathbf{n}_{ijk} \cdot (\overline{\mathbf{c}}_i - \overline{\mathbf{c}}_j + (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{x}_{ijk}) + d_{ijk}]^2, \tag{57}$$

and can be used instead of F_2 when we are already close to the final surface.

After solving the linear system arising from the minimization of (56), we move each individual patch S_i by a helical motion computed from $(\mathbf{c}_i, \overline{\mathbf{c}}_i)$ according to subsection 2.4. Step size control is recommended, and has to be performed with the same factor for all systems.

For a second order motion approximant, we would have to apply formula (8), which results in very similar additions as observed in subsection 6.3.

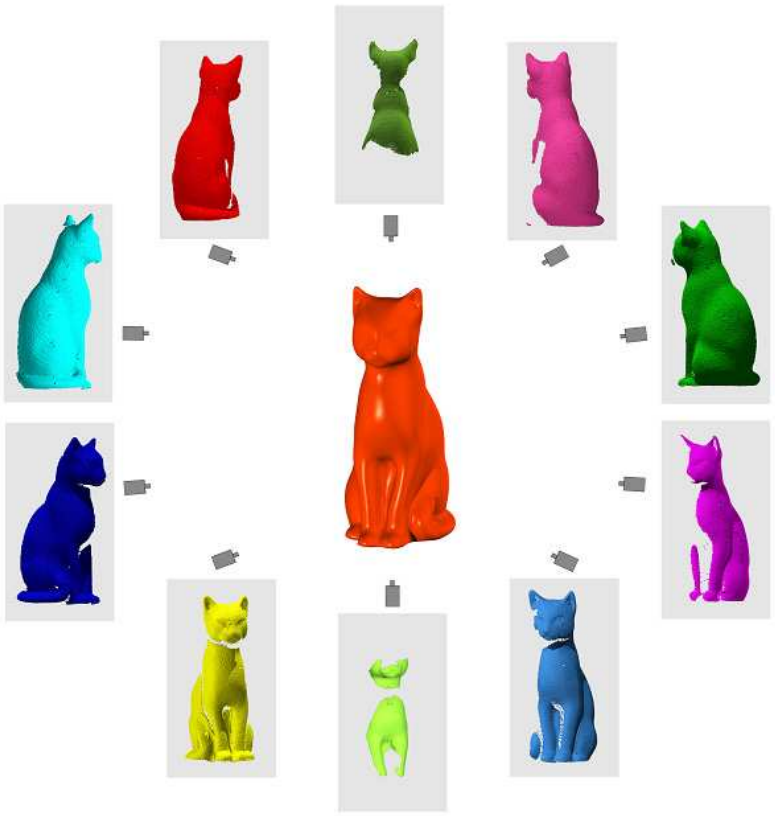
8.3 Examples

The main goal we have in mind is the numerical verification of the local convergence behavior of our algorithms in the environment of N systems. We use manually specified confidence values of surface patches and manually defined initial positions to simplify our discussion. An automatic determination could be obtained by methods proposed in [15,17,22].

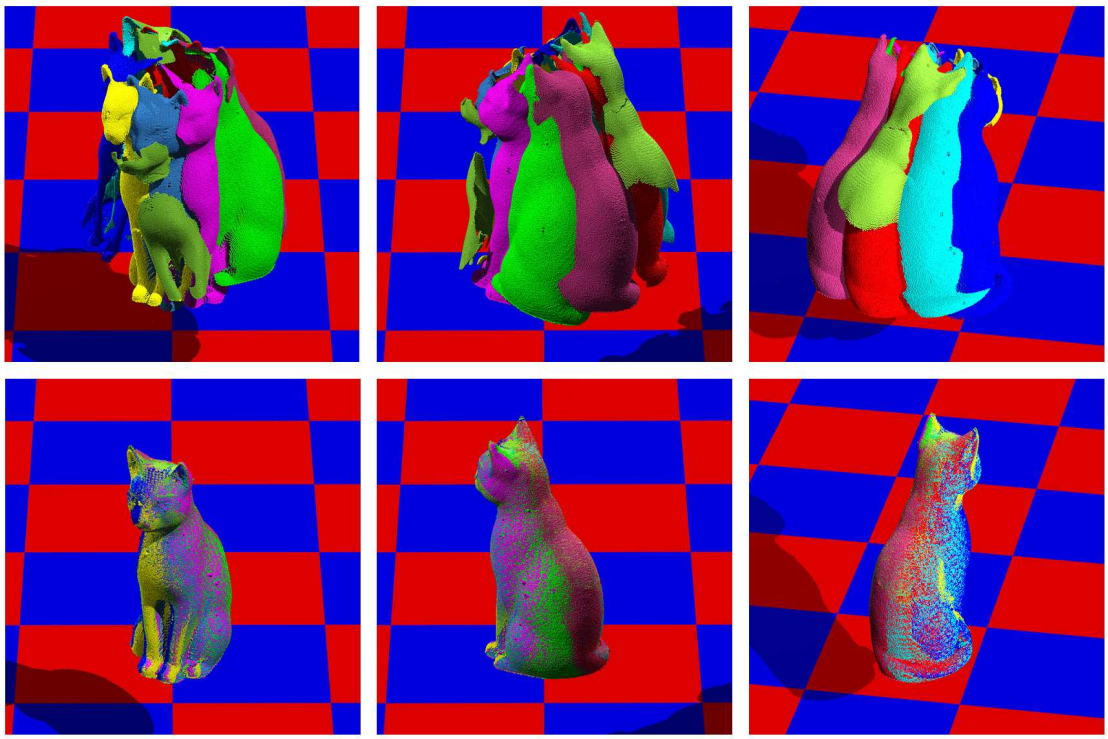
Our example has 10 systems, which are partially overlapping scans of a cat model, acquired with a 3D laser scanning device. This process is illustrated in Fig. 7(a). Each point cloud contains 40000–60000 points, but only 1000 randomly chosen points are taken in each iteration step. However, this data point selection is not applied when we are computing distance functions among different point clouds.

We test the Newton algorithms with first and second order motion approximants, respectively, and the standard ICP as well. One patch is fixed during the whole process for all three algorithms. The squared distance fields are represented via the modified d2tree structure introduced in section 3.1. Figure 7(b) shows initial and final positions of the ten point clouds from three different views. The initial positions for all three algorithms are chosen to be the same and their final positions are nearly the same, so that we only illustrate the final position of the second order Newton method. To ensure global convergence, the second order Newton algorithm is regularized by the Levenberg-Marquardt method. The convergence behavior of the algorithms is shown in Table 5.

Comparing the tested algorithms, we find that the second order Newton algorithm exhibits the fastest local convergence, while the first order Newton algorithm is superior to the other algorithms with respect to global stability.



(a) Simultaneous registration of ten scans of a cat model



(b) (Top) Initial positions and (Bottom) registered model

Fig. 7. Simultaneous registration of 10 views

Table 5: Error reduction for the simultaneous registration of 10 views

	Newton, 2nd order			Newton, 1st order			ICP	
j	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$	$\frac{E(j)}{E(j-1)^2}$	$E(j)$	$\frac{E(j)}{E(j-1)}$
0	1.322e-1	—	—	1.322e-1	—	—	1.322e-1	—
1	4.363e-2	3.301e-1	2.496	4.363e-2	3.301e-1	2.496	4.601e-2	0.3480
2	1.509e-2	3.458e-1	7.927	1.291e-2	2.959e-1	6.782	3.215e-2	0.6987
3	4.279e-3	2.835e-1	18.792	2.924e-3	2.265e-1	1.754e+1	2.347e-2	0.7300
4	4.874e-4	1.139e-1	26.620	3.283e-4	1.123e-1	3.841e+1	1.938e-2	0.8257
5	9.037e-6	1.845e-2	38.041	1.617e-5	4.925e-2	1.501e+2	1.608e-2	0.8297
6	2.897e-8	2.763e-3	305.75	7.210e-7	4.459e-2	2.757e+3	1.335e-2	0.8302
7	1.784e-13	7.144e-6	286.12	3.528e-8	4.893e-2	6.787e+4	1.108e-2	0.8299
⋮								
10	—	—	—	4.455e-12	5.135e-2	7.194e+8	6.336e-3	0.8304
11	—	—	—	2.228e-13	5.023e-2	1.122e+10	5.259e-3	0.8305
⋮								
100	—	—	—	—	—	—	4.979e-10	0.8326

9 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 improve the local convergence behavior of registration algorithms. In particular, we have proposed algorithms with local quadratic convergence. The theoretical results could be nicely verified at hand of computed examples.

We have also observed (see [23]) that one may get a larger funnel of attraction for the minimizer with the proposed schemes compared to the standard ICP algorithm, but we still need a reasonable initial position. This holds in particular, if we want to simultaneously register several overlapping parts (scans) of the same object.

Our future work will concentrate on the early phase of registration, in particular on the choice of the initial position. Currently, we are investigating algorithms which rely on a *distance measure between local surface regions*. We want to realize this idea by the use of image manifolds [19], which are formed over the given geometry data with help of integral invariants [21]. The latter can be computed in a stable way even in the presence of noise. Thus, for the present application, they are superior to differential invariants, especially if we have to handle point cloud data. The new algorithms shall not only use distances in 3D object space, like the concepts presented in the present paper, but employ the distance function to the image manifolds in a higher dimensional space. There, surface parts with similar local shape appear to be close, even if they are still far away in object space. This is expected to give a significant improvement of the global convergence behavior.

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

[1] F. Bernardini, H. Rushmeier. The 3D model acquisition pipeline. Computer Graphics Forum **21** (2002), 149–172.

[2] P. J. Besl, N. D. McKay. A method for registration of 3D shapes. IEEE Trans. Pattern Anal. and Machine Intell. **14** (1992), 239-256.

[3] O. Bottema, B. Roth. *Theoretical Kinematics*, Dover, New York, 1990.

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

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

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

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

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

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

[10] R. Fletcher. *Practical Methods of Optimization*. Wiley, New York, 1987.

[11] C. Geiger, C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, Heidelberg, 2002.

[12] M. Hofer, H. Pottmann, B. Ravani. From curve design algorithms to motion design. Visual Computer **20** (2004), 279–297.

[13] M. Hofer, H. Pottmann. 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.

[14] B. K. P. Horn. Closed form solution of absolute orientation using unit quaternions. Journal of the Optical Society A **4** (1987), 629–642.

[15] D. Huber, M. Hebert. Fully Automatic Registration of Mutiple 3D Data Sets. Image and Vision Computing **21** (2003), 637–650.

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

[17] A. E. Johnson. Spin Images: A Representation for 3D Surface Matching. PhD thesis, Carnegie Mellon Univ., 1997.

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

[19] R. Kimmel, R. Malladi, N. Sochen. Images as embedded maps and minimal surfaces: movies, color, texture and volumetric medical images. Intl. J. Computer Vision **39** (2000), 111–129.

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

[21] S. Manay, B.-W. Hong, A. J. Yezzi, S. Soatto. Integral invariant signatures. Proceedings of ECCV'04, Springer LNCS 3024, 2004, pp. 87–99.

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

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

[24] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.P Seidel. Multi-level partition of unity implicits. ACM Trans. Graphics **22** (SIGGRAPH 2003), 153–161.

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

[26] H. Pottmann, S. Leopoldseder, M. Hofer. Registration without ICP. Computer Vision and Image Understanding **95** (2004), 54–71.

[27] H. Pottmann, J. Wallner. *Computational Line Geometry*. Springer-Verlag, 2001.

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

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

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

[31] T. M. Tucker, T. R. Kurfess. Newton methods for parametric surface registration, Part 1: Theory. Computer-Aided Design **35** (2003), 107–114.