SIMULTANEOUS REGISTRATION OF MULTIPLE VIEWS OF A 3D OBJECT

Helmut Pottmann [a], **Stefan Leopoldseder** [a], **Michael Hofer** [a]

[a] Institute of Geometry, Vienna University of Technology, Wiedner Hauptstr. 8–10, A–1040 Wien, Austria - pottmann@geometrie.tuwien.ac.at, leopoldseder@geometrie.tuwien.ac.at, hofer@geometrie.tuwien.ac.at

KEY WORDS: registration, matching, inspection, three-dimensional data, CAD, geometry.

ABSTRACT

In the reconstruction process of geometric objects from several three-dimensional images (clouds of measurement points) it is crucial to align the point sets of the different views, such that errors in the overlapping regions are minimized. We present an iterative algorithm which simultaneously registers all 3D image views. It can also be used for the solution of related positioning problems such as the registration of one or several measurement point clouds of an object to a CAD model of that object. Our method is based on a first order kinematical analysis and on local quadratic approximants of the squared distance function to geometric objects.

1 INTRODUCTION

In the reconstruction process of surfaces with help of stereo photogrammetry one often obtains several point clouds arising from different views of the object. By evaluation of the surface texture in the different images, correspondences between points of overlapping point clouds can be determined with some confidence value. Now, the different point clouds have to be combined into one consistent representation. There, we may get errors in the overlapping regions. Minimization of those errors is the goal of the present algorithm.

A more challenging task is the simultaneous registration of several moving systems where no point-to-point correspondences are known. One application where only two systems are involved is the following:

Suppose that we are given a large number of 3D data points that have been obtained by some 3D measurement device (laser scan, light sectioning, ...) from the surface of a technical object. Furthermore, let us assume that we also have got the CAD model of this workpiece. This CAD model shall describe the 'ideal' shape of the object and will be available in a coordinate system that is different to that of the 3D data point set. For the goal of shape inspection it is of interest to find the optimal Euclidean motion (translation and rotation) that aligns, or registers, the point cloud to the CAD model. This makes it possible to check the given workpiece for manufacturing errors and to classify the deviations.

Another application with more than two systems is the multiple matching of different 3D laser scanner images of some 3D object. The 3D point sets of different views will be given in different coordinate systems, their position in a common 'object' coordinate system may be known only approximately. Now the key task is to simultaneously match, or register, the different point sets such that they optimally fit in their overlapping regions.

In the following we show how to solve these optimization problems with an iterative algorithm. In each iteration step all N systems are registered simultaneously. An arbitrary number of systems (at least one) is kept fixed. The algorithm uses a kinematical analysis of first order and solves a linear system of equations, which comes from a least squares problem. In the case that we do not have correspondences, the algorithm is based on local quadratic approximants of the squared distance function to geometric objects. The novelty in our method concerns the following apects: We use local quadratic approximants of the squared distance function instead of pure point-point or point-plane distances. By an approach which relies on instantaneous kinematics we just solve a linear system in each iteration step, even in case of simultaneously registering more than two systems.

In Sec. 2 we briefly review contributions in the literature which are closely related to our algorithm. In Sec. 3 some basic facts of spatial kinematics are collected. Sec. 4 is devoted to the mathematical description of our algorithm which simultaneously aligns multiple point clouds in the case that point-to-point correspondences are known. In Sec. 5 we describe how to treat the more difficult case where no correspondences are given. Finally, in Sec. 6 topics of further research are addressed.

2 CURRENT REGISTRATION ALGORITHMS

Let us first focus on registration problems where only two systems are involved ($N = 2$). One system moves relative to the second system which is kept fixed. If point-to-point correspondences are known, the optimal motion that minimizes the Euclidean distances between corresponding points can be explicitly given. The use of quaternions for determining this motion can already be found in (Faugeras Hebert, 1986, Horn, 1987). In many applications, however, no point-to-point correspondences are given. One example is the alignment of a single 3D point cloud to a geometric entity which could be a CAD model or another 3D point set. Here a well-known standard algorithm is the iterative closest point (ICP) algorithm of Besl and McKay (Besl McKay, 1992). In Sec. 2.1 we will briefly summarize this algorithm which establishes point-to-point correspondences in each iteration step and uses the representation of 3D Euclidean motions by unit quaternions. For an overview on the recent literature on this topic we refer to (Eggert et al., 1998). A summary with new results on the acceleration of the ICP algorithm has been given by Rusinkiewicz and Levoy (Rusinkiewicz Levoy, 2001).

There are two major restrictions of the ICP algorithm. First, it is implicitly assumed that one of the data sets is a subset of the other. The presence of points that have no corresponding point in the other set leads to incorrect assignments. Several different approaches to threshold outliers have been presented, see the literature cited in (Eggert et al., 1998). Secondly, the ICP algorithm is a two set approach and is not directly extendable to multiple data sets. It is not sufficient to apply registration to consecutive pairs of 3D point sets, since alignment errors accumulate and certain point sets will be very poorly adjusted. There have been several approaches to the simultaneous registration of all data sets, see e.g. the spring force model of (Eggert et al., 1998). (Bergevin et al., 1996) apply incremental transformations to all the moving systems with respect to a fixed system and use point-to-tangent plane correspondences in the overlapping regions.

2.1 The ICP algorithm

A point set ('data' shape) is rigidly moved (registered, positioned) to be in best alignment with the corresponding CAD model ('model' shape) in the following iterative way.

In the first step of each iteration, for each point of the data shape, the closest point 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.

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

$$\sum_i \|\mathbf{y}_i - M(\mathbf{x}_i)\|^2.$$

This least squares problem can be solved explicitly, see e.g. (Besl McKay, 1992, Horn, 1987). The translational part of M brings the center of mass of X to the center of mass of Y. The rotational part of M can be obtained as the unit eigenvector that corresponds to the maximum eigenvalue of a symmetric 4×4 matrix. The solution eigenvector is nothing but the unit quaternion description of the rotational part of M.

After this second step the positions of the data points are updated via $X_{\text{new}} = M(X_{\text{old}})$. Now step 1 and step 2 are repeated, always using the updated data points, as long as 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 both in steps 1 and 2.

3 KINEMATICS

An important part of the algorithm uses instantaneous kinematics. Thus, we briefly describe the most important facts before going into the details of the algorithm.

Consider a continuous one-parameter motion of a rigid body in space. If $\mathbf{x}$ is a point in Euclidean three-space, the symbol $\mathbf{v}(\mathbf{x})$ denotes the velocity vector of that point of the moving body which is at this moment at position $\mathbf{x}$. Thus $\mathbf{v}(\mathbf{x})$ is a time-dependent vector attached to the point $\mathbf{x}$. It is well known that at some instant t, a smooth motion has a velocity vector field of the form

$$\mathbf{v}(\mathbf{x}) = \overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}, \tag{1}$$

with vectors $\mathbf{c}, \overline{\mathbf{c}} \in \mathbb{R}^3$. Thus the velocity vector field (or the *infinitesimal motion*) at some instant t is uniquely determined by the pair $(\mathbf{c}, \overline{\mathbf{c}})$.

Of special interest are the *uniform* motions, whose velocity vector field is constant over time. It is well known that 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 $\overline{\mathbf{c}}$. The paths of a uniform translation are straight lines parallel to $\mathbf{c}$.

2. *Uniform rotations* with nonzero angular velocity about a fixed axis. We have $\mathbf{c} \cdot \overline{\mathbf{c}} = 0$, but $\mathbf{c} \neq \mathbf{o}$. The point trajectories of a uniform rotation are circles with the rotational axis as common axis.

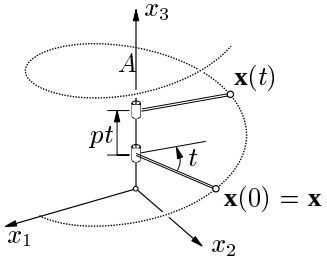

Figure 1: Helical motion.

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 the point $\mathbf{x}$ is situated on the axis, its path coincides with the axis. The trajectories of the other points are *helices*.

   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. We use the convention that ω is nonnegative, that $p > 0$ for right-handed helical motions, and that $p < 0$ for left-handed ones.

   To further clarify the concept of a uniform helical motion, we assume the x_3-axis of a Cartesian system (x_1, x_2, x_3) to be the helical axis (see Fig. 1). A uniform helical motion may then be written as $\mathbf{x} \mapsto \mathbf{x}(t)$ with

$$\mathbf{x}(t) = \begin{pmatrix} \cos t & -\sin t & 0 \\ \sin t & \cos t & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ p \cdot t \end{pmatrix}, \tag{2}$$

   $p = 0$ means a uniform rotation, and for $p \to \infty$ the motion tends to a uniform translation.

Since all possible pairs $(\mathbf{c}, \overline{\mathbf{c}})$ actually occur, we can use these three cases to classify the type of velocity vector field at one instant of an arbitrary smooth motion: *Infinitesimal translations* are characterized by $\mathbf{c} = \mathbf{o}$, and *infinitesimal rotations* by $\mathbf{c} \cdot \overline{\mathbf{c}} = 0$. The remaining velocity vector fields are said to belong to *infinitesimal helical motions*. At all instants, if the velocity vector field of a smooth motion is nonzero, it belongs to one of the three cases.

If $(\mathbf{c}, \overline{\mathbf{c}})$ represents the velocity vector field of a uniform rotation or helical motion, then the Plücker coordinates $(\mathbf{g}, \overline{\mathbf{g}})$ of the axis A, the pitch p and the angular velocity ω are reconstructed by

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

see e.g. (Pottmann Wallner, 2001).

The Plücker coordinates $(\mathbf{g}, \overline{\mathbf{g}})$ of a straight line A consist of a direction vector $\mathbf{g}$ and the moment vector $\overline{\mathbf{g}}$ about the origin. From the moment vector, we can easily compute a point $\mathbf{p}$ of the line A, since for all points $\mathbf{p}$ on A we have the relation $\overline{\mathbf{g}} = \mathbf{p} \times \mathbf{g}$.

The above results about infinitesimal motions are a limit case of the following fundamental result of 3-dimensional kinematics: Any two positions of a rigid body in 3-space can be transformed onto each other by a (discrete) helical motion (consisting of a rotation about an axis and a translation along that axis), including the special cases of a pure rotation and a pure translation.

Our algorithm actually iteratively computes the velocity vector field of a discrete helical motion. Those underlying helical motions are then used for the displacement.

4 SIMULTANEOUS REGISTRATION WITH KNOWN CORRESPONDENCES

The first application we have in mind is the simultaneous registration of N point clouds which have been obtained by stereo photogrammetry. The point clouds partially overlap, and in these regions correspondences (plus confidence values) between points of different point clouds are known from surface texture analysis.

The N point clouds can be viewed as rigid systems and are denoted by Σ_i. An arbitrary number (at least one) of the given systems remains fixed. The others shall be moved such that after application of the motions the distances of corresponding points, weighted with their confidence value, are as small as possible. Since in our case we have $N > 2$, only an iterative procedure is possible. We use a geometric method that involves instantaneous kinematics, and thus it is similar to the approach in (Bourdet Clément, 1988).

Only those points in a cloud are used for the alignment process which belong to an overlapping region with a neighboring point cloud. For such a given data point pair $(\mathbf{x}_i, \mathbf{y}_i)$ we know the index j of the system Σ_j to which $\mathbf{x}_i$ belongs,

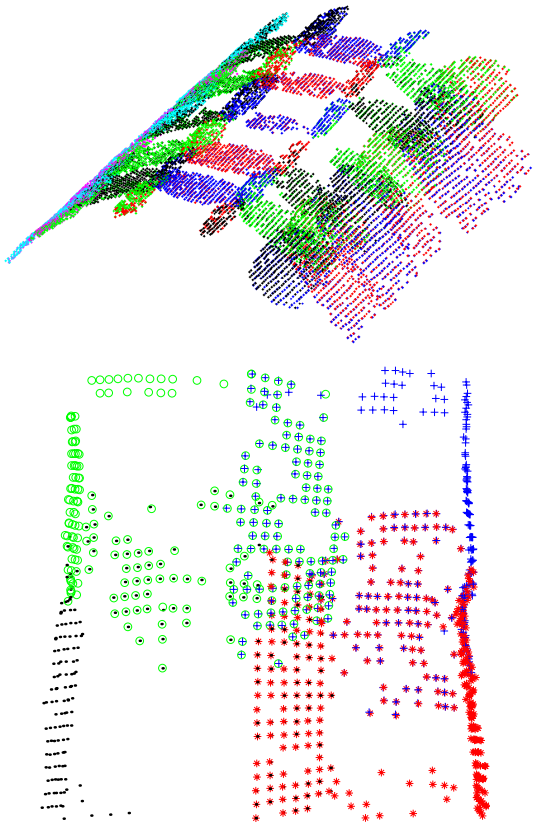

Figure 2: Example of multiple registration with known correspondences: All given 30 point clouds (top) and detail of 4 clouds showing the overlapping areas (bottom). Data by courtesy of Gerhard Paar, Joanneum Research.

and the index k indicating the system Σ_k of the point $\mathbf{y}_i$. The point pairs have found to be in correspondence with a confidence value $w_i \in (0, 1)$.

Our goal is to move each system Σ_l by a motion M_l in a way, such that after application of all these motions M_l, the new positions of corresponding points are as close as possible to each other in a least squares sense. Thereby we have to keep in mind the confidence values of correspondences.

4.1 Displacement estimation via instantaneous kinematics

Since the expected motions are small displacements anyway, we replace them by instantaneous motions. The instantaneous motion of system Σ_l against one fixed system (called Σ_0 henceforth) possesses a velocity vector field. It is characterized by two vectors $\mathbf{c}_l, \overline{\mathbf{c}}_l \in \mathbb{R}^3$, and analogously to Eq. 1, the velocity vector $\mathbf{v}_{l0}$ of a point $\mathbf{x}_i \in \Sigma_l$ is then given by

$$\mathbf{v}_{l0}(\mathbf{x}_i) = \overline{\mathbf{c}}_l + \mathbf{c}_l \times \mathbf{x}_i. \quad (4)$$

For a pair of corresponding points $(\mathbf{x}_i, \mathbf{y}_i)$ we would like to estimate their distance after the motions have been applied to their systems Σ_j and Σ_k, respectively. In first order, we

can use the velocity vectors, and thus the squared distance of the displaced points is given by

$$Q_1(\mathbf{x}_i, \mathbf{y}_i) = (\mathbf{x}_i + \mathbf{v}_{j0}(\mathbf{x}_i) - \mathbf{y}_i - \mathbf{v}_{k0}(\mathbf{y}_i))^2 = (\mathbf{x}_i - \mathbf{y}_i + (\overline{\mathbf{c}}_j + \mathbf{c}_j \times \mathbf{x}_i) - (\overline{\mathbf{c}}_k + \mathbf{c}_k \times \mathbf{y}_i))^2. \quad (5)$$

This term $Q_1(\mathbf{x}_i, \mathbf{y}_i)$ is a *quadratic* function in the unknowns $\mathbf{c}_j, \overline{\mathbf{c}}_j, \mathbf{c}_k, \overline{\mathbf{c}}_k$ of the instantaneous motions applied to the involved systems Σ_j and Σ_k.

There is an alternative to Eq. (5): Instead of linearizing the motion of Σ_j against Σ_0 and the motion of Σ_k against Σ_0, one can linearize the relative motion of Σ_j against Σ_k. The velocity vector $\mathbf{v}_{jk}$ of a point $\mathbf{x}_i \in \Sigma_j$ for this relative motion is given by

$$\mathbf{v}_{jk}(\mathbf{x}_i) = \mathbf{v}_{j0}(\mathbf{x}_i) - \mathbf{v}_{k0}(\mathbf{x}_i). \quad (6)$$

Here, the distance of interest is between the point $\mathbf{x}_i + \mathbf{v}_{jk}(\mathbf{x}_i)$ and $\mathbf{y}_i$ (i.e., $\mathbf{x}_i$ is interpreted to be moving with system Σ_j relative to the point $\mathbf{y}_i$ in system Σ_k). The squared distance of these two points of interest is given by

$$Q_2(\mathbf{x}_i, \mathbf{y}_i) = (\mathbf{x}_i + \mathbf{v}_{jk}(\mathbf{x}_i) - \mathbf{y}_i)^2 = (\mathbf{x}_i - \mathbf{y}_i + (\overline{\mathbf{c}}_j + \mathbf{c}_j \times \mathbf{x}_i) - (\overline{\mathbf{c}}_k + \mathbf{c}_k \times \mathbf{x}_i))^2. \quad (7)$$

The term $Q_2(\mathbf{x}_i, \mathbf{y}_i)$ is again a quadratic function in the unknowns $\mathbf{c}_j, \overline{\mathbf{c}}_j, \mathbf{c}_k, \overline{\mathbf{c}}_k$.

We see that any pair of corresponding points gives rise to such a quadratic term Q_1 or Q_2 in the involved unknown motion parameters. That term is a first order estimate of the squared distance (error) after application of the motions. Hence, to perform the error minimization, we will minimize the following weighted sum

$$F = \sum_i w_i Q_2(\mathbf{x}_i, \mathbf{y}_i). \quad (8)$$

The weight w_i is the known confidence value of the pair $(\mathbf{x}_i, \mathbf{y}_i)$. Note that since point-to-point correspondences are known, both Q_1 and Q_2 can be used. Without known correspondences, however, it is necessary to use the velocity vectors $\mathbf{v}_{jk}$ for the relative motion of Σ_j against Σ_k, see Sec. 5.

The minimization of F is mathematically simple, because F is a quadratic function in the unknown motion parameters $\mathbf{c}_l, \overline{\mathbf{c}}_l$. Collecting all unknowns in the vector $\mathcal{C} = (\mathbf{c}_1, \overline{\mathbf{c}}_1, \mathbf{c}_2, \overline{\mathbf{c}}_2, \ldots, \mathbf{c}_N, \overline{\mathbf{c}}_N)^T$, we may write F in the form

$$F = \mathcal{C}^T \cdot \mathcal{B} \cdot \mathcal{C} + 2\mathcal{A} \cdot \mathcal{C} + \sum_i w_i (\mathbf{x}_i - \mathbf{y}_i)^2. \quad (9)$$

Hence, the minimizer $\mathcal{C}$ of F solves the following linear system,

$$\mathcal{B} \cdot \mathcal{C} + \mathcal{A}^T = 0, \quad (10)$$

where $\mathcal{B}$ is a $6 \times 6N$ matrix and $\mathcal{A}$ is a $1N \times 6N$ matrix.

Note that it is very easy to fix more than one system. Fixing Σ_l just requires to set both vectors $\mathbf{c}_l$ and $\overline{\mathbf{c}}_l$ equal to zero.

4.2 Computing the actual displacements from velocities

In the previous subsection we have estimated the displacement vector of a point (i.e., the vector pointing from the old to the new position) with help of the velocity vector of an instantaneous motion. However, displacing points in this way would result in an affine mapping of the corresponding system Σ_l and not in a rigid body motion. Although such affine transformations are actually used in the literature (Bourdet Clément, 1988), we prefer to compute exact rigid body motions in the following way.

It is sufficient to explain this for one moving system, which we denote by Σ, and whose instantaneous displacement is given by the vectors $\mathbf{c}, \overline{\mathbf{c}}$. 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 idea now is to move points via that helical motion approximately as far as indicated by the velocity vectors (points are now moved along helical paths of that motion). Note that $\|\mathbf{c}\|$ gives the angular velocity of the rotational part. We apply a motion to Σ which is the superposition of a rotation about the axis A through an angle of $\alpha = \arctan(\|\mathbf{c}\|)$ and a translation parallel to A by the distance of $p \cdot \alpha$.

A rotation through an angle of α about an axis (with unit direction vector $\mathbf{a} = (a_x, a_y, a_z)$) through the origin is known to be given by $\mathbf{x}' = \mathcal{R} \cdot \mathbf{x}$ with orthogonal matrix

$$\mathcal{R} = \frac{1}{m_{00}} \cdot \begin{bmatrix} m_{11} & 2(b_1b_2 + b_0b_3) & 2(b_1b_3 - b_0b_2) \\ 2(b_1b_2 - b_0b_3) & m_{22} & 2(b_2b_3 + b_0b_1) \\ 2(b_1b_3 + b_0b_2) & 2(b_2b_3 - b_0b_1) & m_{33} \end{bmatrix},$$
$$m_{00} = b_0^2 + b_1^2 + b_2^2 + b_3^2, m_{11} = b_0^2 + b_1^2 - b_2^2 - b_3^2,$$
$$m_{22} = b_0^2 - b_1^2 + b_2^2 - b_3^2, m_{33} = b_0^2 - b_1^2 - b_2^2 + b_3^2, \quad (11)$$

where $b_0 = \cos(\alpha/2)$, $b_1 = a_x \sin(\alpha/2)$, $b_2 = a_y \sin(\alpha/2)$, $b_3 = a_z \sin(\alpha/2)$.

The superposition of the rotation about the axis A with Plücker coordinates $(\mathbf{a}, \overline{\mathbf{a}})$, cf. Eq. (3), through an angle α and the translation parallel to A by $p \cdot \alpha$ is then given by

$$\mathbf{x}' = \mathcal{R}(\mathbf{x} - \mathbf{p}) + (p \cdot \alpha)\mathbf{a} + \mathbf{p}, \quad (12)$$

where $\mathcal{R}$ is the matrix given above and $\mathbf{p}$ is an arbitrary point on the helical axis (e.g. $\mathbf{p} = \mathbf{a} \times \overline{\mathbf{a}}$).

4.3 Iteration and termination criteria

With the methods from 4.1 the algorithm iteratively computes instantaneous motions of the moving systems, whose actual displacements are then computed as in 4.2. This iterative procedure is terminated if one of the two following conditions is satisfied.

1. The improvement in the objective function (8), after some iteration step, is below a chosen value.

2. The number of iterations exceeds a chosen constant.

5 SIMULTANEOUS REGISTRATION WITHOUT CORRESPONDENCES

Given are N clouds of 3D data points (systems $\Sigma_1, \ldots, \Sigma_N$) that partially overlap, but now we have neither correspondences between point pairs, nor confidence values, as in Sec. 4. But we still assume that a good initial position of these N 3D point sets in a global coordinate system is known. Furthermore we know which of the pairs of systems Σ_j, Σ_k actually overlap.

In Sec. 4 each data point $\mathbf{x}_i \in \Sigma_j$ that corresponds to $\mathbf{y}_i \in \Sigma_k$ contributes to the functional F in Eq. (8) with the term $Q_2(\mathbf{x}_i, \mathbf{y}_i)$ which is quadratic in the unknowns $\mathbf{v}_{jk}(\mathbf{x}_i) = (\overline{\mathbf{c}}_j - \overline{\mathbf{c}}_k) + (\mathbf{c}_j - \mathbf{c}_k) \times \mathbf{x}_i$. Now we also want to set up such a quadratic functional for each data point $\mathbf{x}_i$ in an overlapping region. We present a strategy based on a quadratic approximation of the squared distance function, cf. (Pottmann Hofer, 2002).

If two systems Σ_j, Σ_k overlap we triangulate the point cloud in Σ_k and refer to the triangulated point cloud as $T(\Sigma_k)$. First we determine those points $\mathbf{x}_i \in \Sigma_j$ that actually lie in the overlapping region, i.e., their distance to $T(\Sigma_k)$ is below a certain threshold. For literature on these thresholding techniques, see e.g. (Blais Levine, 1995, Eggert et al., 1998, Zhang, 1994).

Now, the goal is to bring the points $\mathbf{x}_i$ closer to the geometric shape $T(\Sigma_k)$, i.e., to move the points $\mathbf{x}_i$ to lower levels of the squared distance function to the triangulated surface $T(\Sigma_k)$. In (Pottmann Hofer, 2002) it is described how to compute for any point $\mathbf{x}_i \in \mathbb{R}^3$ a local quadratic approximant $F_{d,\mathbf{x}_i} =: F_d^i$ to a *smooth* surface. For our applications this has to be a nonnegative quadratic function, $F_d^i(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathbb{R}^3$. Here we do not have a smooth surface but a triangulated point cloud $T(\Sigma_k)$. We can apply the results of (Pottmann Hofer, 2002), if we are able to locally approximate the triangulated surface by a smooth surface. For this we can use local quadric fits to $T(\Sigma_k)$, see e.g. (Yang Lee, 1999). In case that the triangulation is too coarse, we may first apply mesh refinement techniques (e.g. interpolatory subdivision) to get a sufficiently dense triangulation (Dyn et al., 1990, Zorin, 1997).

Now the registration of two such overlapping point clouds is found by iteratively minimizing the functional

$$\sum_i F_d^i(\mathbf{x}_i + \mathbf{v}_{jk}(\mathbf{x}_i)), \tag{13}$$

which is quadratic in the unknowns $\mathbf{c}_j, \overline{\mathbf{c}}_j, \mathbf{c}_k, \overline{\mathbf{c}}_k$. If we do not consider points $\mathbf{x}_i$ from *one* overlapping region (Σ_j, Σ_k) only, but points from *all* overlapping regions simultaneously, then we have to minimize the functional (13) for the unknowns $\mathbf{c}_1, \overline{\mathbf{c}}_1, \ldots, \mathbf{c}_N, \overline{\mathbf{c}}_N$. In order to fix certain systems Σ_l, one simply has to set the vectors $\mathbf{c}_l, \overline{\mathbf{c}}_l$ equal to zero.

As a simple example for a quadratic approximant F_d^i of the squared distance function of $T(\Sigma_k)$ at the point $\mathbf{x}_i$, we would like to present a squared point-tangent plane distance: Let $\mathbf{y}_i$ denote the closest point of $T(\Sigma_k)$ to $\mathbf{x}_i$. We estimate the unit normal vector $\mathbf{n}_i$ to $T(\Sigma_k)$ in $\mathbf{y}_i$, e.g. by computing a regression plane using the points in a certain neighborhood of $\mathbf{y}_i$. In the following we refer to the plane with normal vector $\mathbf{n}_i$ through $\mathbf{y}_i$ as the tangent plane of $T(\Sigma_k)$ in $\mathbf{y}_i$. Let d_i denote the oriented distance of $\mathbf{x}_i$ to this plane. If $\mathbf{x}_i$ is already close to $T(\Sigma_k)$ it is better to first refine the triangulation and then compute the nearest point and proceed as mentioned above.

We want to minimize the squared distance of the displaced point $\mathbf{x}_i^* = \mathbf{x}_i + \mathbf{v}_{jk}(\mathbf{x}_i)$ to the tangent plane at $\mathbf{y}_i$. The distance is given by $d_i + \mathbf{n}_i \mathbf{v}_{jk}(\mathbf{x}_i)$ and thus we get the functional

$$\begin{aligned} &Q_3(\mathbf{x}_i) = F_d^i(\mathbf{x}_i^*) = (d_i + \mathbf{v}_{jk}(\mathbf{x}_i) \cdot \mathbf{n}_i)^2 \\ &= (d_i + (\overline{\mathbf{c}}_j - \overline{\mathbf{c}}_k) \cdot \mathbf{n}_i + \det(\mathbf{c}_j - \mathbf{c}_k, \mathbf{x}_i, \mathbf{n}_i))^2, \end{aligned} \tag{14}$$

which is quadratic in the unknowns $\mathbf{c}_j, \overline{\mathbf{c}}_j, \mathbf{c}_k, \overline{\mathbf{c}}_k$. In this way we can compute a quadratic term $Q_3(\mathbf{x}_i)$ for all points $\mathbf{x}_i$ that have been found to lie in an overlapping region. Hence, to simultaneously register all N point clouds (where at least one is fixed) we have to minimize the following functional

$$F = \sum_i Q_3(\mathbf{x}_i). \tag{15}$$

As mentioned in Sec. 4 one has to observe the fact that the map $\mathbf{x}_i \mapsto \mathbf{x}_i + \mathbf{v}_{j0}(\mathbf{x}_i)$ is no Euclidean rigid body motion. See Sec. 4.2 for the computation of the rigid motion which brings $\mathbf{x}_i$ close to tips of the vectors $\mathbf{x}_i + \mathbf{v}_{j0}(\mathbf{x}_i)$.

The squared distance function to the tangent plane has already been used in several variants to the ICP algorithm, cf. (Bergevin et al., 1996, Chen Medioni, 1992). Our approach of a local quadratic approximant of the squared distance function is more general and includes the squared distance function to the tangent plane as a special case. Fig. 3 gives an example for our algorithm in the case $N = 2$, namely the registration of a point cloud to a CAD model. In Fig. 4 the mean squared error of the data points to the CAD surface after each iteration is given, both for our algorithm and for the standard ICP algorithm. Our algorithm converges much faster than ICP, and with respect to the number of iterations it is comparable to Chen and Medioni's method. However, we solve just a linear system in each iteration step, whereas Chen and Medioni in each step run a numerical optimization algorithm. We are currently working on an efficient organization of the local quadratic approximants in a spatial data structure in order to speed up the algorithm. Then, industrial inspection tasks are expected to come close to real time performance, since the computations on the approximation of the squared distance function to the CAD model can be done in a preprocessing step. Further ideas for acceleration and stabilization of the algorithm can be taken from (Rusinkiewicz Levoy, 2001).

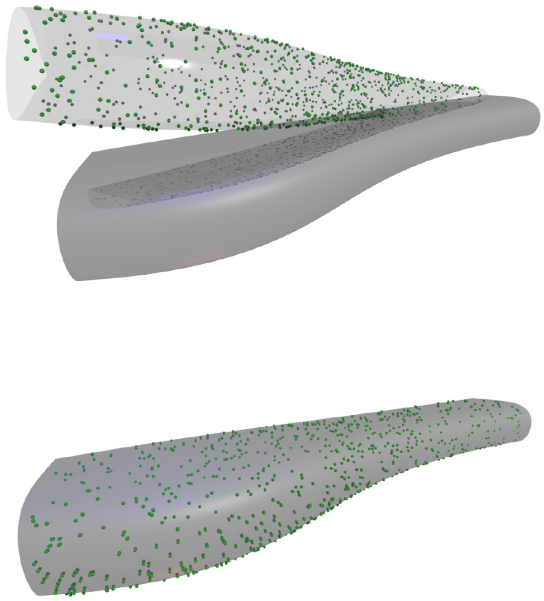

Figure 3: Registration of a point cloud to a CAD model: Initial and final position of the point cloud

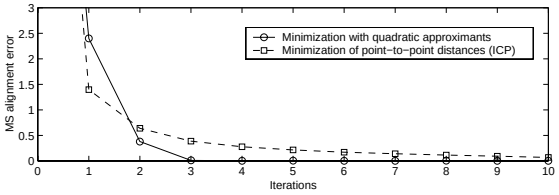

Figure 4: Comparison of the convergence rate for our method vs. minimization of point-to-point distances in each step (ICP).

6 EXTENSIONS AND FUTURE RESEARCH

Here we did not deal with those systematic and random errors in the 3D point clouds that arise in the data capturing process. Our algorithm for the simultaneous registration of multiple point clouds iteratively minimizes a function $F = \sum_i \omega_i Q_k(\mathbf{x}_i), k = 1, 2, 3$. The weights ω_i can be used to successively downweight outliers. Appropriate weighting schemes may be found in (Rousseeuw Leroy, 1987).

In our contribution we have assumed that a rough, initial alignment of the point clouds is given and that small displacements of the point clouds are sufficient to bring them in optimal alignment. A very ambitious task for future research is to derive a stable algorithm to find these rough initial alignments. Such an algorithm should be applicable to multiple point clouds and should exploit as much information on the involved geometric entities as possible.

Finally, for special geometries like 3D objects composed of simple surfaces one may have additional or more precise information on the squared distance function (Kverh Leonardis, 2002). Additional work has to be done to include such information in the basic algorithm.

ACKNOWLEDGEMENTS

This work has been carried out within the K plus Competence Center *Advanced Computer Vision* and was funded from the K plus program.

REFERENCES

Bergevin R., Laurendeau D., Poussart, D., 1996. Registering range views of multipart objects. *Comput. Vision Image Understanding*, 61, pp. 540–547.

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

Blais, G., Levine, D., 1995. Registering multiview range data to create 3D computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17, pp. 820–824.

Bourdet, P., Clément, A., 1988. A study of optimal-criteria identification based on the small-displacement screw model. *Annals of the CIRP*, 37, pp. 503–506.

Chen, Y., Medioni, G., 1992. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10, pp. 145–155.

Dyn, N., Levin, D., Gregory, J. A., 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2), pp. 160–169.

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

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

Horn, B. K. P., 1987. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4, pp. 629–642.

Kvehr, B., Leonardis, A., 2002. A new refinement method for registration of range images based on segmented data. *Computing*, 68(1), pp. 81-96.

Nikolaidis, N., Pitas, I., 2001. *3-D Image Processing Algorithms*. Wiley.

Pottmann, H., Hofer, M., 2002. Geometry of the squared distance function to curves and surfaces. Proceedings *Mathematics and Visualization*, Springer, to appear.

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

Rousseeuw, P. J., Leroy, A. M., 1987. *Robust Regression and Outlier Detection*, Wiley.

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

Yang, M., Lee, E., 1999. Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design*, 31, pp. 449–457.

Zhang, Z., 1994. Iterative point matching for registration of smooth surfaces using differential properties. In: *Proceedings of the 3rd European Conference on Computer Vision*, Stockholm, pp. 397–406.

Zorin, D., Schröder, P., Sweldens, W., 1997. Interactive multiresolution mesh editing. In: *SIGGRAPH 97 Conference Proceedings*, pp. 259–268.