Kinematical methods for the classification, reconstruction, and inspection of surfaces

H. Pottmann
University of Technology, Vienna

J. Wallner
University of Technology, Vienna

S. Leopoldseder
University of Technology, Vienna

Mots clefs : surface reconstruction, surface inspection, surface registration, kinematics, line geometry, Computer Aided Design, Computer Vision

Introduction

This contribution discusses recent progress in our investigation of reconstruction of geometric objects from point clouds such as laser scanner data.

We first focus on a class of simple surfaces, which includes surfaces of revolution, cylindrical surfaces, and helical surfaces. The basic idea behind the reconstruction process is to find a motion which generates this surface. The algorithm works by fitting a velocity vector field to a given point cloud. This procedure is of a line-geometric nature and can be formulated as an approximation problem in line space. It basically amounts to solving a general eigenvalue problem. By locally applying this method we can reconstruct other types of surfaces as well — composite surfaces, and pipe or profile surfaces.

The second part of this addresses a geometric positioning problem, namely the optimal matching of a cloud of points (obtained by discrete measurements) to a CAD model. We further discuss related problems in Computer Vision and Robotics.

1 Kinematics

Consider the 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 $\mathbf{c}, \overline{\mathbf{c}}$. 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 $\overline{\mathbf{c}}$.

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

   Formally, $p = 0$ means a uniform rotation and $p = \infty$ is a translation.

All possible pairs $(\mathbf{c}, \overline{\mathbf{c}})$ actually occur, so 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, the velocity vector field of a smooth motion belongs to one of the three cases, if it is nonzero.

It turns out that it is useful to study *path normals* of motions, i.e. lines that are orthogonal to the velocity vector of one of their points. Here it is convenient to describe lines by their *Plücker coordinates.* If a line G contains a point $\mathbf{p}$ and is parallel to the vector $\mathbf{v}$, then the pair $(\mathbf{g}, \overline{\mathbf{g}}) = (\mathbf{v}, \mathbf{p} \times \mathbf{v})$ is called its Plücker coordinate vector. It is easy to see that $\overline{\mathbf{g}}$ does not depend on the particular choice of $\mathbf{p}$. The Plücker coordinate vector is unique only up to scalar multiples. Its two components $\mathbf{g}$, $\overline{\mathbf{g}}$ are not independent, but fulfill the relation $\mathbf{g} \cdot \overline{\mathbf{g}} = 0$. A point $\mathbf{x}$ is contained in G if and only if $\mathbf{x} \times \mathbf{g}$ equals $\overline{\mathbf{g}}$.

Connections between Plücker coordinates of lines and velocity vector fields are shown by the following two lemmas. For a more detailed treatment, see [13].

Lemma 1 *A line with Plücker coordinates* $(\mathbf{g}, \overline{\mathbf{g}})$ *is a path normal of a smooth motion* $(\mathbf{c}, \overline{\mathbf{c}})$*, if and only if* $\mathbf{c} \cdot \overline{\mathbf{g}} + \overline{\mathbf{c}} \cdot \mathbf{g} = 0$.

Lemma 2 *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, the angular velocity* ω *and the pitch* p *are reconstructed by*

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

2 Reconstruction of simple surfaces

Reconstruction of surfaces is an important aspect of Reverse Engineering (cf. [15]). Here we first consider the reconstruction of 'simple' surfaces from point clouds. 'Simple' means 'invariant' in the sense defined below (Sec. 2.1). This reconstruction process consists of two steps, the first of which determines a motion which is associated with the surface. In a second step we find a curve which generates the given surface when the motion of step 1 is applied to it.

2.1 Invariant surfaces

A curve which undergoes a motion, generates a surface. If this motion is uniform, i.e., its velocity vector field is constant, the surface is called *invariant* with respect to this motion. In this case the uniform motion transforms the surface into itself.

Examples of such surfaces are surfaces of revolution, cylindrical surfaces (generated by an arbitrary base curve), and helical surfaces. The converse is also true: all invariant surfaces are of these three types, and they are generated by a uniform motion.

There is a small number of *multiply invariant* surfaces: A cylinder of revolution is generated alternatively by a rotation about its axis, by a translation parallel to the generator lines, or even by any helical motion which is a superposition of both. A plane is generated by any translation parallel to the surface and by any rotation whose axis is orthogonal to it. A sphere is generated by (and also invariant with respect to) all rotations whose axes contain the center.

This connection between surfaces and motions can be used for reconstruction of such invariant surfaces. The idea of the procedure is that surface normals are path normals for the underlying motion which generates the surface. If there is more than one possible motion, the surface normals must be path normals for all of them.

2.2 Fitting a velocity vector field to surface normals

We assume that we are given a point cloud $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$ representing a surface, and that we have already estimated normal vectors $\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_N$ at these points. The Plücker coordinates of the surface normals $N_1, N_2, \ldots$ are therefore given by $(\mathbf{n}_i, \overline{\mathbf{n}}_i)$ with $\overline{\mathbf{n}}_i = \mathbf{p}_i \times \mathbf{n}_i$. A velocity vector field $(\mathbf{c}, \overline{\mathbf{c}})$ where the surface normals are path normals must fulfill the equation $\mathbf{c} \cdot \overline{\mathbf{n}}_i + \overline{\mathbf{c}} \cdot \mathbf{n}_i = 0$ for all i. Thus we can fit a 'best' velocity vector field to the surface normals if we choose normalized unit vectors $\mathbf{n}_i$ and minimize

$$F(\mathbf{c}, \overline{\mathbf{c}}) = \sum (\mathbf{c} \cdot \overline{\mathbf{n}}_i + \overline{\mathbf{c}} \cdot \mathbf{n}_i)^2 \to \min, \quad (\|\mathbf{c}\| = 1). \tag{3}$$

F is a quadratic function of six real arguments, and the side condition $\|\mathbf{c}\| = 1$ is also quadratic. We can therefore rewrite Equ. (3) in the form

$$(\mathbf{c}, \overline{\mathbf{c}})^T \cdot K \cdot (\mathbf{c}, \overline{\mathbf{c}}) \to \min, \quad (\mathbf{c}, \overline{\mathbf{c}})^T \cdot D \cdot (\mathbf{c}, \overline{\mathbf{c}}) = 1, \tag{4}$$

with two (6×6)-matrices K and D. The matrix D has nonzero entries only in its upper left 3×3 corner. The solution of this problem is straightforward. The minimum is assumed for $(\mathbf{c}, \overline{\mathbf{c}})$ which fulfills

$$(K - \lambda D) \cdot (\mathbf{c}, \overline{\mathbf{c}}) = (\mathbf{o}, \mathbf{o}), \quad \|\mathbf{c}\| = 1, \quad \det(K - \lambda D) = 0, \quad \lambda \text{ minimal.} \tag{5}$$

This means that we have to choose the smallest solution λ of the cubic equation $\det(K - \lambda D) = 0$ and solve the equation $(K - \lambda D)(\mathbf{c}, \overline{\mathbf{c}}) = (\mathbf{o}, \mathbf{o})$. Details can be found in [8, 10, 12, 13].

The resulting velocity vector field $\mathbf{v}(\mathbf{x}) = \overline{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$ fits the given surface normals best in the sense that F is minimized as above. We can compute axis and pitch of the underlying uniform motion using Equ. (2). If the pitch is very small ($\mathbf{c} \cdot \overline{\mathbf{c}} \ll \mathbf{c}^2$), we have a uniform motion very close to a rotation, and if the pitch is very large ($\mathbf{c} \cdot \overline{\mathbf{c}} \gg \mathbf{c}^2$) and/or the axis is very far away ($\overline{\mathbf{c}}^2 - \mathbf{c} \cdot \overline{\mathbf{c}} \gg \mathbf{c}^2$), we have hit a uniform motion very close to a translation. In both cases we may want to consider a restricted fitting problem: To compute a uniform rotation which fits the input data best, we minimize F under the side conditions $\|\mathbf{c}\| = 1$, $\mathbf{c} \cdot \overline{\mathbf{c}} = 0$. To find a uniform translation, we minimize F under the side condition $\mathbf{c} = 0$, $\|\overline{\mathbf{c}}\| = 1$.

Another phenomenon which might occur is that there is not one well-defined smallest solution λ of Equ. (5), but there are two or three. In this case there is a two-space or even three-space of nearly minimizing vectors $(\mathbf{c}, \overline{\mathbf{c}})$, and the point cloud belongs to a multiply invariant motion, i.e., to a cylinder and or a sphere.

Smallness of λ has to be tested in the following way: It follows directly from Equ. (5) that $F(\mathbf{c}, \overline{\mathbf{c}}) = \lambda$. If the diameter of the point cloud is of magnitude δ, the Plücker coordinates $(\mathbf{n}_i, \overline{\mathbf{n}}_i)$ of the normals are of magnitude $(1, \delta)$. The solution vector $(\mathbf{c}, \overline{\mathbf{c}})$ is of magnitude $(1, \delta)$, which follows from Equ. (1). Thus F is of magnitude $N \cdot \delta^2$, and λ is to be compared with $N\delta^2$. The magnitude of λ gives information about how well the velocity field fits the input data.

2.3 Fitting special velocity vector fields

If we have reason to expect a cylinder or sphere which fits the input data, it is not necessary to perform the general minimization algorithm. In the cylindrical case, the direction $\overline{\mathbf{c}}$ of the rulings is orthogonal to the surface normals, so it can be found by solving

$$F'(\overline{\mathbf{c}}) = \sum (\overline{\mathbf{c}} \cdot \mathbf{n}_i)^2 \to \min, \quad \|\overline{\mathbf{c}}\| = 1. \tag{6}$$

Spherical input data mean that the surface normals intersect the sphere's center. If this center is denoted by $\mathbf{m}$, the Plücker coordinates of the normals must equal $(\mathbf{n}_i, \overline{\mathbf{n}}_i) = (\mathbf{n}_i, \mathbf{m} \times \mathbf{n}_i)$. Thus $\mathbf{m}$ may be reconstructed by solving

$$F''(\mathbf{m}) = \sum (\overline{\mathbf{n}}_i - \mathbf{m} \times \mathbf{n}_i)^2 \to \min. \tag{7}$$

These are quadratic problems with (in part) quadratic side conditions, so their solution is straightforward. The quality of the approximation may be tested by the magnitude of $\overline{\mathbf{c}} \cdot \mathbf{n}_i$ in the cylindrical case and by the uniformity of $\|\mathbf{p}_i - \mathbf{m}\|$ in the spherical case. More details can be found in [8, 10, 12, 13].

2.4 Surface reconstruction

The actual surface which fits the input data is reconstructed from the point cloud in a second step. We will first describe the procedure for surfaces of revolution. Suppose we have found a uniform rotation by fitting a velocity vector field to the input data. The axis of this rotation is computed with Equ. (2). Then we choose a half-plane bounded by the axis and rotate all data points $\mathbf{p}_i$ about the axis until they hit this half-plane. Thus we get new points $\mathbf{q}_i$, contained in a certain half-plane. If the points $\mathbf{p}_i$ can be approximated by a surface of revolution Φ, then the points $\mathbf{q}_i$ can be approximated by a curve c, which in turn generates Φ. This is illustrated in Fig. 1.

It is obvious how to modify the procedure for helical surfaces and for cylindrical ones: In both cases we choose a plane orthogonal to the trajectory of a point with respect to the underlying uniform motion. The case of the helical surface in its full generality is not as easy to implement as the other two cases.

3 Composite Reconstruction

There are surfaces which are locally well approximated by the surfaces of Sec. 2. One class of such surfaces are smooth surfaces which have a kind of 'osculating' simpler surface, analogously to an osculating circle. *Pipe surfaces*, which are generated as the envelope of a moving sphere, are locally well approximated by tori. *Profile surfaces*, which are generated by a planar curve, whose plane is rolling on a developable surface, are locally well approximated by surfaces of revolution.

A second class are surfaces composed of several different pieces of simple surfaces. This includes most surfaces of parts used e.g. in mechanical engineering. Surfaces which do not consist of pieces of planes, cylinders, spheres, surfaces of revolution, and helical surfaces are rare in many areas of application.

To reconstruct either type of surface in a satisfactory manner, we have to consider the problem of deciding which subsets of a given point cloud are well approximated by the simple surfaces of Sec. 2. A solution is provided by a suitable *region growing* algorithm, which grows an initially small subset until no simple surface fits well enough. Fig. 2 illustrates the reconstruction of a profile surface.

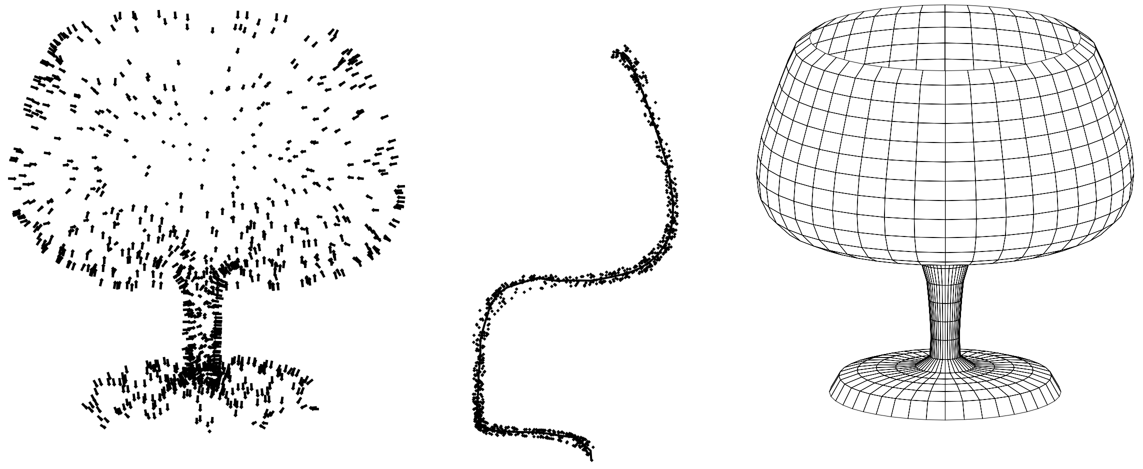

Figure 1: Reconstruction of a surface of revolution. Left: data points, estimates of normal vectors. Center: points projected into a half-plane and a curve approximating this point set. Right: final surface of revolution.

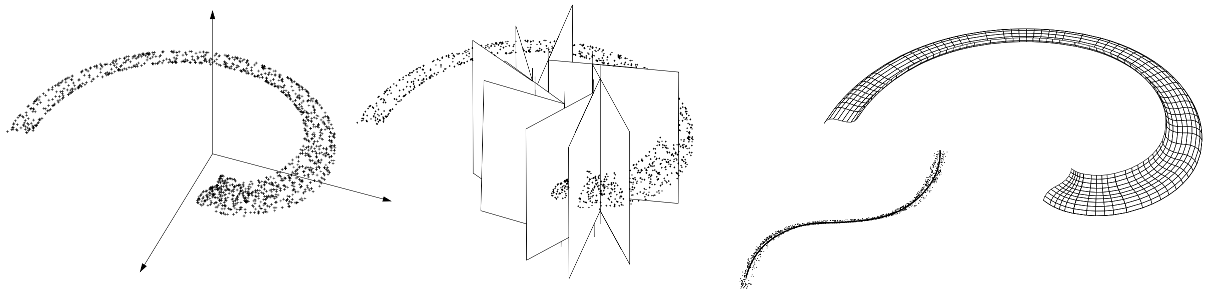

Figure 2: Reconstruction of a profile surface (from left to right). 1. Original point cloud. 2. Subsets which have good approximation by a surface of revolution. 3. Reconstruction of generator curve. 4. Smooth profile surface.

4 Surface Inspection

There is an application of these methods to the problem of optical inspection of parts by light sections. This topic belongs to Computer Vision (see [3, 7]). There a sequence of thin planar sheets of light illuminates curves on the surface of the part. The optical information gathered from these curves is used for inspection or reconstruction of the surface. This method only works well if the angle α enclosed by the surface normal and the light plane does not differ too much from zero: If the light planes have width δ, then the width of the illuminated curve is approximately $\delta/\cos\alpha$ (cf. Fig. 3).

4.1 Light planes by least squares

It is therefore important to choose the light planes such that for all surface points the angle α is bounded by a certain constant. This can be achieved as follows: If the surface to be inspected happens to be a surface of revolution with axis A, we could choose the light planes incident with A, and we would have $\alpha = 0$ always. If the surface is cylindrical, we could choose the light planes orthogonal to the generator lines, with the same result. This motivates to approximate the given surface by a surface of revolution with axis A, or by a cylindrical surface with generator

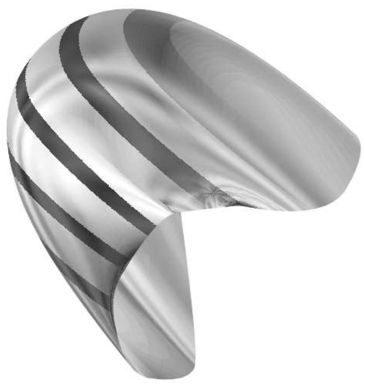

Figure 3: The principle of the light section method (courtesy M. Hofer)

lines parallel to a line L, and to choose the light planes incident with A or orthogonal to L, respectively.

The normal vectors of the given surface enter the approximation in an essential way. However, it lies in the nature of a least squares method that the approximation is good only in the mean, and not necessarily good enough for each single data point.

4.2 Light planes by medial axis transform

For the case of parallel light planes, the following approach of Smith et al. [14] overcomes this difficulty: The existence of a light plane such that $\alpha < c$ for all surface points is equivalent to the existence of a unit vector $\mathbf{v}$ which encloses an angle $\pi/2 - \alpha > c' = \pi/2 - c$ with all unit normal vectors $\pm\mathbf{n}$ of the surface.

In order to make c minimal, i.e., c' maximal, we consider the set of all unit normal vectors $\pm\mathbf{n}$ of the given surface (the *symmetrized Gauss image*), compute the spherical medial axis of the complement, and choose $\mathbf{v}$ such that its distance to the symmetrized Gauss image is maximal. The spherical medial axis is intimately connected with the planar medial axis via the stereographic projection. An algorithm based on this relationship is developed in [14]. Fig. 4 illustrates this method.

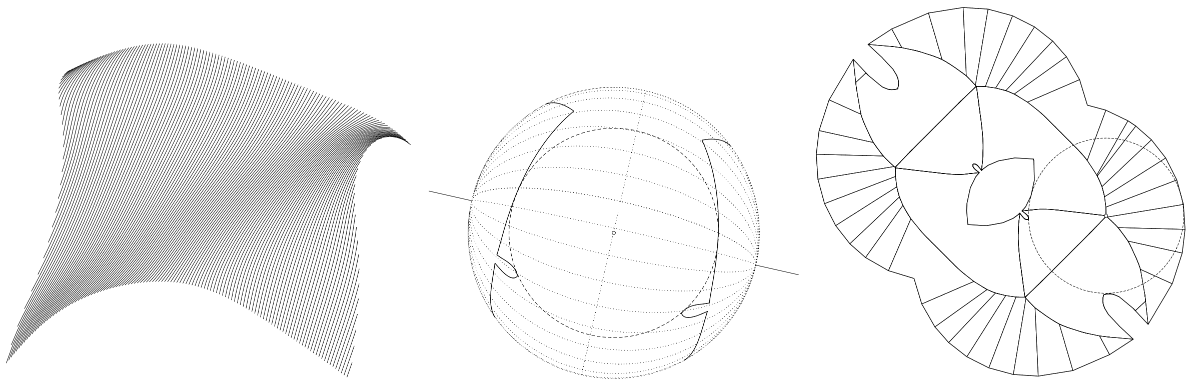

Figure 4: Computation of optimal light section plane. Left: Surface with planar sections (already optimal); Center: Symmetrized Gauss image with largest spherical circle contained in the complement; Right: Stereographic projection of the Gauss image and spherically largest circle.

5 The Registration Problem

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 of the registration problem is the multiple matching of different 3D laser scanner images of some 3D object. To reconstruct the shape of a given workpiece one may, e.g., place it on a turntable and obtain several overlapping images from different viewpoints. The 3D point sets of different views will be given in different coordinate systems, their position in a common 'object' coordinate system is 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 both of the applications mentioned above one fits a given 3D point set to a geometric entity, which is either a CAD model or another 3D point set. In neither case a point-to-point correspondence is known. A well-known standard algorithm to solve such a registration problem is the iterative closest point (ICP) algorithm of Besl and McKay [1]. In Sec. 5.1 we will briefly summarize this algorithm which is based on the representation of 3D Euclidean motions by unit quaternions. The use of quaternions for determining the 'best' motion can already be found in [5, 6]. For an excellent overview on the recent literature on this topic we refer to [4]. In Sec. 5.2 we will propose an approach alternative to the ICP algorithm which is based on instantaneous kinematics. This method shows a faster convergence behavior and is, unlike the ICP algorithm, applicable also for the *simultaneous* matching of *multiple* point sets.

5.1 The ICP algorithm

The point set ('data' shape) is rigidly moved (registered, positioned) to be in best alignment with the CAD model ('model' shape). This is done iteratively:

In the first step of each iteration, for 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 $Y = (\mathbf{y}_1, \mathbf{y}_2, \dots)$ of closest model shape points to the data point sequence $X = (\mathbf{x}_1, \mathbf{x}_2, \dots)$. 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. [1, 6]. 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.

5.2 Registration with instantaneous kinematics

In the ICP algorithm the data points $\mathbf{x}_i$ are moved towards their closest points $\mathbf{y}_i$ on the model surface. Instead of moving $\mathbf{x}_i$ towards $\mathbf{y}_i$ it is a more natural idea to move $\mathbf{x}_i$ towards the tangent plane of the model surface in $\mathbf{y}_i$. As we do not know the corresponding point to $\mathbf{x}_i$ on the model surface anyhow, it is better to bring $\mathbf{x}_i$ close to the tangent plane of $\mathbf{y}_i$ in each iteration step, instead of aiming at $\mathbf{y}_i$ directly. Especially in low curved surface regions the tangent plane in a surface point approximates the surface well and one obtains a much faster convergence of the registration process using the tangent planes compared to the standard ICP algorithm.

Our proposed algorithm will work as follows: the first step is similar to that of the ICP algorithm. For each data point $\mathbf{x}_i \in X$ determine the nearest point $\mathbf{y}_i$ of the model surface shape and determine the tangent plane there. Let $\mathbf{n}_i$ denote a unit normal vector of this tangent plane in $\mathbf{y}_i$. Because $\mathbf{y}_i$ is the nearest point to $\mathbf{x}_i$ on the surface, $\mathbf{x}_i$ lies on the surface normal in $\mathbf{y}_i$, i.e. $\mathbf{x}_i = \mathbf{y}_i + d_i\mathbf{n}_i$ with d_i denoting the oriented distance of $\mathbf{x}_i$ to $\mathbf{y}_i$.

In the second step we would like to move the points $\mathbf{x}_i$ towards the respective tangent planes in $\mathbf{y}_i$. Here it is appropriate to use instantaneous kinematics, an idea that already appeared in a similar form in [2]. Let $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$ denote a velocity vector field according to Equ. (1). The distance of the point $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ to the tangent plane at $\mathbf{y}_i$ with unit normal vector $\mathbf{n}_i$ is given by $d_i + \mathbf{n}_i \cdot \mathbf{v}(\mathbf{x}_i)$, where d_i again denotes the oriented distance of $\mathbf{x}_i$ to $\mathbf{y}_i$. The objective function to be minimized is

$$\sum_i (d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i))^2, \tag{8}$$

which is quadratic in the unknowns $(\mathbf{c}, \bar{\mathbf{c}})$. The unique solution can be given explicitly by solving a system of linear equations.

Note that the transformation which maps $\mathbf{x}_i$ to $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ is an affine map and not a rigid Euclidean motion. Nevertheless, the vector field determined by $(\mathbf{c}, \bar{\mathbf{c}})$ uniquely determines a uniform helical motion M. The axis G and the pitch p of M are computed via Equ. (2). The motion we apply to $\mathbf{x}_i$ is the superposition of a rotation about this axis G through an angle of $\alpha = \cot(\|\mathbf{c}\|)$ and a translation parallel to G by the distance of $p \cdot \alpha$.

Similar to the ICP algorithm we update the data points via $X_{\text{new}} = M(X_{\text{old}})$ and repeat the procedure until the change in mean-square error $\sum d_i^2$ falls below a preset threshold.

Figure 5 shows an example for the registration of a point cloud to a surface. In order to better visualize the spatial position of the point data, a transparent surface is associated with the data point set. This transparent surface does not enter the computation in any way, it is only displayed for reasons of visualization. The pictures show the data point set in its initial position, after the first iteration step, and in its final position after 7 iterations. In this example the error tolerance reached in the final position will be obtained with the standard implementation of the ICP algorithm only after 45 iterations. This is because in the ICP algorithm the data points move towards their nearest position on the surface in each iteration step. A displacement of the data point set in tangential direction to the model surface therefore needs many iterations.

It is straightforward to extend the objective function (8) to a weighted scheme. There are 3D measurement devices that supply for each data point a tolerance for the occurring measurement errors. These can be included in the objective function to downweight outliers.

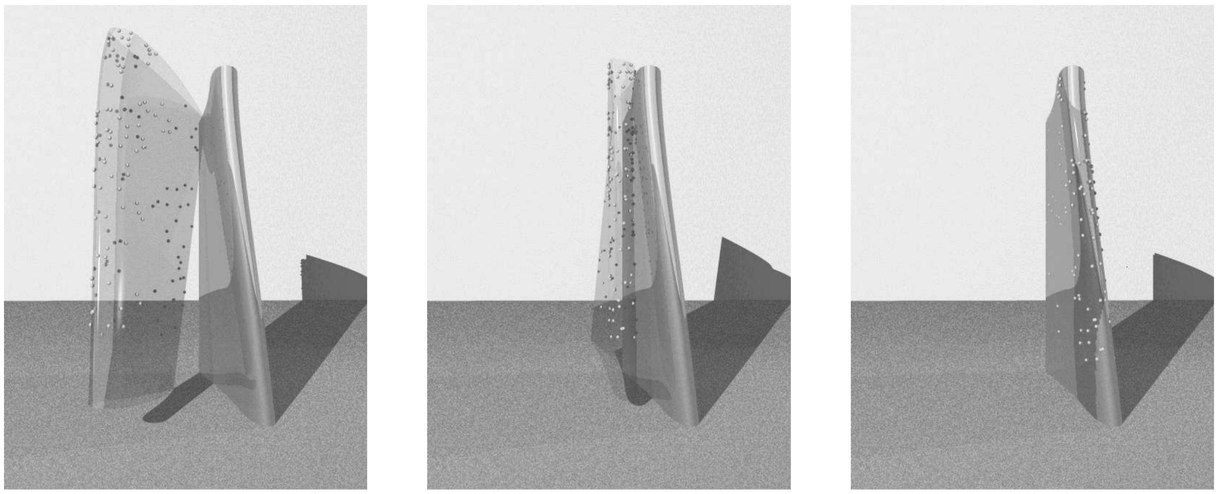

Figure 5: Matching of a point cloud to the corresponding CAD model: Left: initial position of data points and CAD model, Center: position of data points after first iteration step, Right: final position after seven iterations.

References

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

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

[3] R. Cipolla, *Visual Motion of Curves and Surfaces.* Cambridge University Press, 2000.

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

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

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

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

[8] H. Pottmann, I.K. Lee, T. Randrup, *Reconstruction of kinematic surfaces from scattered data*, Proceedings Symposium for Geotechnical and Structural Engineering, Eisenstadt, Austria, 1998, pp. 483–488.

[9] H. Pottmann, M. Peternell, B. Ravani, *Approximation in line space: applications in robot kinematics and surface reconstruction*, in: J. Lenarčič and M. Husty (eds.), Advances in Robot Kinematics: Analysis and Control, Kluwer, 1998, pp. 403–412.

[10] H. Pottmann, T. Randrup, *Rotational and helical surface approximation for reverse engineering.* Computing 60, 307–322, 1998.

[11] H. Pottmann, H.-Y. Chen, I.K. Lee, *Approximation by profile surfaces*, in: R. Cripps (ed.), The Mathematics of Surfaces VIII, Information Geometers, 1998, pp. 17–36.

[12] H. Pottmann, I.K. Lee, J. Wallner, *Scattered data approximation with kinematic surfaces.* Proceedings of "Sampling Theory and Applications '99", Loen, Norway, 1999, pp. 72–77.

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

[14] T. S. Smith, R. T. Farouki, M. al-Kandari, H. Pottmann, *Optimal contour orientations for machining of free-form surfaces.* Technical Report No. 82, Institut für Geometrie, TU Wien, 2001.

[15] T. Várady, P. Benkö, G. Kós, *Reverse engineering regular objects: simple segmentation and surface fitting procedures*, Int. J. Shape Modeling, 4, 127–141, 1998.