Industrial Geometry: Recent Advances and Applications in CAD

H. Pottmann [a,*] S. Leopoldseder [a] M. Hofer [a] T. Steiner [a]
W. Wang [b]

[a] *Vienna Univ. of Technology, Wiedner Hauptstr. 8–10/104, 1040 Wien, Austria*

[b] *University of Hong Kong, 421 Chow Yei Ching Bldg, Pokfulam Rd, Hong Kong*

Abstract

Industrial Geometry aims at unifying existing and developing new methods and algorithms for a variety of application areas with a strong geometric component. These include CAD, CAM, Geometric Modelling, Robotics, Computer Vision and Image Processing, Computer Graphics and Scientific Visualization. In this paper, Industrial Geometry is illustrated via the fruitful interplay of the areas indicated above in the context of novel solutions of CAD related, geometric optimization problems involving distance functions: approximation with general B-spline curves and surfaces or with subdivision surfaces, approximation with special surfaces for applications in architecture or manufacturing, approximate conversion from implicit to parametric (NURBS) representation, and registration problems for industrial inspection and 3D model generation from measurement data. Moreover, we describe a 'feature sensitive' metric on surfaces, whose definition relies on the concept of an image manifold, introduced into Computer Vision and Image Processing by Kimmel, Malladi and Sochen. This metric is sensitive to features such as smoothed edges, which are characterized by a significant deviation of the two principal curvatures. We illustrate its applications at hand of feature sensitive curve design on surfaces and local neighborhood definition and region growing as an aid in the segmentation process for reverse engineering of geometric objects.

Key words: geometric optimization, distance function, curve approximation, surface approximation, active contours, registration, feature sensitivity, mathematical morphology

* Corresponding author.
Email addresses: pottmann@geometrie.tuwien.ac.at (H. Pottmann), leopoldseder@geometrie.tuwien.ac.at (S. Leopoldseder), hofer@geometrie.tuwien.ac.at (M. Hofer), tibor@geometrie.tuwien.ac.at (T. Steiner), wenping@cs.hku.hk (W. Wang).

1 Introduction

During the past decades, geometric methods have played an increasingly important role in a variety of areas dealing with computing for industrial applications; these include Computer-Aided Design and Manufacturing, Geometric Modeling, Computational Geometry, Robotics, Computer Vision, Pattern Recognition and Image Processing, Computer Graphics and Scientific Visualization.

These areas originated from different requirements in specific applications and thus they have seen rather disjunct developments. In fact, very similar problems have been treated by different communities. These communities still have different favorite solutions to nearly the same problems. Let us illustrate this at hand of curve approximation. According to industry standards, the CAD approach uses B-spline curves and a method for data fitting which iterates between parameter estimation and linear least squares approximation [11,35]. Computer Vision and Image Processing developed another method, *active contours* [4,12], which have originally been formulated as parametric curves. Nowadays, the advantages of (discretized) implicit representations and the formulation of the curve evolution via partial differential equations in the *level set method* [19,31] are highly appreciated, in particular for difficult curve approximation problems which arise in image segmentation. Curve approximation also appears in higher dimensional spaces: For example, in the space of rigid body motions it leads to motion design for Robotics [17] or Computer Animation.

In recent years, these different areas of research have started to become increasingly interconnected, and have even begun to merge. A driving force in this process is the increasing complexity of applications, where one field of research alone would be insufficient to achieve useful results. Novel technologies for acquisition and processing of data lead to new and increasingly challenging problems, whose solutions require the combination of techniques from different branches of applied geometry. The thereby emerging research area, which aims at unifying existing and developing new methods and algorithms for a variety of application areas with a strong geometric component, shall be called *Industrial Geometry*.

Let us continue the example from curve approximation addressed above. The viewpoint of Industrial Geometry would be to investigate the various algorithms from a common perspective. Since all the available algorithms are solving nonlinear geometric optimization problems, it is appropriate to study and compare the known approaches from the optimization perspective. In the present paper, we will point to recent results in this direction.

It is impossible to outline all major current research streams in Industrial Geometry in this paper. Therefore, we will focus just on a few topics. We will briefly look at the *level set method* [19,31] and on *hybrid data structures for geometric computing* [15]. The major part of this paper is devoted to *geometric optimization problems* which involve *distance functions*. Here we will present a survey with some new results on a recently developed class of optimization algorithms, which can be called *squared distance minimization*. The benefits of the optimization viewpoint rather than the perspective of a specific application will become obvious. With nearly the same algorithms we can solve a wide class of curve and surface approximation problems and a number of registration (surface matching) problems.

The methods we are using for the topics indicated so far have a relation to Computer Vision and Image Processing. As a further example for the fruitful use of techniques which originate in these fields, we discuss a new metric on surfaces. It is sensitive to features such as smoothed edges, which are characterized by a significant deviation of the two principal curvatures. This new metric can be easily understood with the concept of an image manifold [14], and it has a number of interesting applications [26]: For example, we can design curves on surfaces whose shape is adapted to the features of the surface. Moreover, we briefly address local neighborhood definition and region growing as an aid in the segmentation process for reverse engineering of geometric objects. Image processing frequently uses *mathematical morphology* for basic topological and geometric operations [10,30]; this work describes similar operations on surfaces, which – if desired – can be made sensitive to the features.

2 Geometry representations

The choice of an appropriate representation of a geometric object is a fundamental issue for the development of efficient algorithms. Following a recent survey by L. Kobbelt [15], one may classify the basic types of 3D geometry representations according to the following table.

	unstructured	structured	hierarchical
explicit	point clouds	binary voxel grid	octree
parametric	triangle mesh	NURBS	subdivision surface
implicit	moving least squares surface	3D grid	octree, binary space partitions

Explicit representations are meant as sequences of points and can be seen as maps $f : \mathbb{N} \to \mathbb{R}^3$. Parametric representations are described by maps $f : \mathbb{R}^2 \to \mathbb{R}^3$ and implicit representations by trivariate functions $f : \mathbb{R}^3 \to \mathbb{R}$. In the table above, the basic data structures which are at our disposal are called *unstructured* (list, graphs; they have a sequential or topological ordering, respectively), *structured* (array; has a global index structure) and *hierarchical* (octrees, binary space partitions). Basic operations which are frequently performed within geometric algorithms are evaluation (computing points, normals, ...), queries (inside or outside, distance, closest point,...), and modification of geometry and/or topology. The various entries in the table behave quite differently with respect to these operations.

Whereas Computer Graphics seems to use all these representations by now, CAD so far focuses on a few of them. This is probably not an ideal situation. On the other hand we see the possibility of achieving big progress by looking at the entire collection of representations, and by combining them in an optimal way (see 2.1).

The level set method in CAD

The *implicit representation* of a surface Φ in $\mathbb{R}^3$ describes it as zero set of a function $f : \mathbb{R}^3 \to \mathbb{R}$,

$$\Phi := \{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = 0\}. \tag{1}$$

Associated with f, we have a whole family of *level sets*,

$$\Phi_c := \{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = c = const.\}. \tag{2}$$

It is sometimes an advantage to view the whole family. In connection with curves or surfaces which evolve in some optimization procedure, this is a fruitful approach and one of the basic ingredients in the highly successful *level set method* [19,31]. The level set method formulates the optimization process of the shape under consideration (called *active curve* or *active surface*) with a partial differential equation (PDE) and employs efficient algorithms for the numerical solution of that PDE on a grid.

The level set method is very popular in Computer Vision, Image Processing and Computer Graphics [19,29,31]. We have not seen many applications of the level set method in CAD so far, but it can be expected that this picture will change. A main concern which might have hindered the use of the level set method, is the representation it is based on: an implicit representation, evaluated just on a grid. However, there are a variety of complicated shape

computation problems for which one does not need to work throughout the whole computation with the final NURBS representation. We may decouple the shape finding procedure from the final representation. The level set method can be applied to shape optimization and then one applies a conversion procedure from level sets to NURBS. This conversion is briefly addressed in 4.6, but requires more studies for successful practical use.

2.1 Hybrid geometry representations

A promising direction for future research has been opened in recent research by L. Kobbelt. He proposes *hybrid representations*, which are various clever combinations of geometry representations. The aim is to use the individual parts in these combinations for those operations where they perform best. For example, a combination of a mesh with an implicit representation can be applied to mesh repair. The combination of a polygon and a grid leads to a formulation of an active contour in the plane (called r-snake), which is easy to implement and allows us to control the topology [3]. The latter is an important issue for the level set method, whose original formulation would easily achieve changes in the topology of the deforming shape, but hardly allow us a control over that change.

3 Distance functions

The distance function of a curve or surface M assigns to each point $\mathbf{x}$ of the embedding space the shortest distance $d(\mathbf{x})$ of $\mathbf{x}$ to M. Since d is not differentiable at M one often uses the signed distance function, which agrees with d up to the sign. It is well defined for a closed object and takes on different signs inside and outside the object, respectively. In the following, we will just speak of the distance function for both the signed and the unsigned version.

3.1 A view into the literature

The distance function is an excellent example of a topic which has been addressed by all areas which involve geometric computing. Early work on the geometry of the distance function comes from the classical geometric literature of the 19th century. One looks at its graph surface, which consists of developable surfaces of constant slope and applies results of classical differential geometry, line and sphere geometry (for a modern presentation, see e.g. [27]).

The level sets of the distance function of a geometric object M are the offsets of M, which are of particular importance in Computer Aided Design and Manufacturing (see e.g. [11,20]).

Distance functions are also basic to morphological operators in Image Processing [10,30]. The distance function is not differentiable at points of the *cut locus*, which is a concept that appears in different variants (medial axis, skeleton, bisector,....) in various areas for a number of applications (for CAD related work, see e.g. [20]).

Computer Graphics uses distance functions in many ways, for example in adaptively sampled distance fields [9]. These proved to be a versatile and unifying representation with many applications (NC simulation, interference checking, sculpting,...). Distance functions also blend well with a recent trend in Computer Graphics of working directly with clouds of points rather than meshes.

Optimal robot trajectories are in a natural way related to shortest paths on manifolds and thus distance functions play a central role [17]. They also occur in obstacle avoidance with the potential field (barrier) approach [17].

Algorithms for fast computation of the distance function in two or three dimensions are often performed on a grid. One exploits the fact that the distance function has a normalized gradient field, i.e., it is a solution of the eikonal equation $\|\nabla f(\mathbf{x})\| = 1$. The main types of algorithms are fast marching [31] and fast sweeping [33,38]. Computational Geometry developed different types of algorithms for fast distance computations. We point especially to approximate nearest neighbor algorithms (see e.g. [1]), which are even working well in higher dimensions, where a grid based computation would hardly be feasible.

As an example, we consider the computation of the distance function of a geometric object Φ in the presence of obstacles: The function value $d(\mathbf{x})$ at a point $\mathbf{x}$ (not in an obstacle) is the length of the shortest path from $\mathbf{x}$ to a point of Φ, which avoids the obstacles. Zhao's algorithm [38] is very well suited to solve this problem: we just have to mark the grid points inside the obstacles with a flag; these points will never be updated and therefore never influence the computation of the distance function in the admissible points of the grid. The distance function of a point $\mathbf{p}$ in the plane in the presence of some obstacles, computed with Zhao's algorithm, is shown in Fig. 1. We also see that this is only an approximation, since the precise level sets near the point $\mathbf{p}$ should be circles. Tsai's algorithm [33] does not have this distortion, but on the other hand it is not easily extendable to the presence of obstacles. Fig. 1 furthermore shows the shortest paths connecting the point $\mathbf{p}$ with six other points and respecting the obstacles.

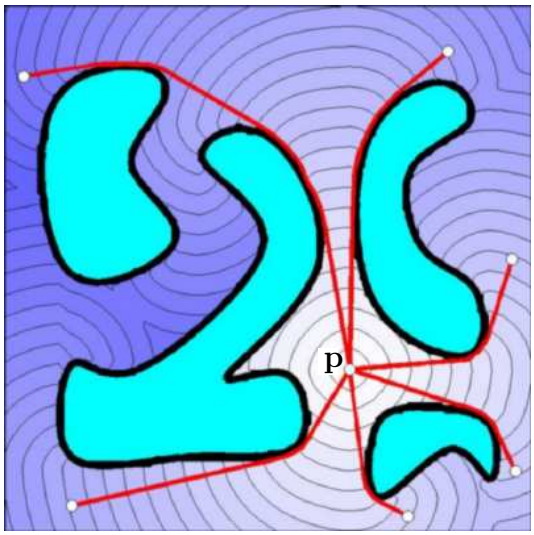


Fig. 1. Level sets of the distance function of a point $\mathbf{p}$ in the presence of obstacles and shortest paths emanating from $\mathbf{p}$, respecting the obstacles.

3.2 Quadratic approximants of the squared distance function

In subsequent optimization algorithms we will have to minimize functions, which contain sums of squared distances of points to a curve or surface. In order to achieve good local convergence, we will use a Newton or quasi-Newton algorithm, and this requires local quadratic approximants of the squared distance function of a curve or surface.

Such local quadratic approximants have been studied in [22]. We briefly summarize here the main results and start with the squared distance function $d^2(\mathbf{c})$ of a planar curve $\mathbf{c}$. Deriving a second order approximant only makes sense at a smooth point $\mathbf{p}$ of that function, and thus we exclude points on the cut locus.

Consider an admissible point $\mathbf{p}$ in the plane. The point $\mathbf{c}_0 \in \mathbf{c}$, which is closest to $\mathbf{p}$ is a normal foot point (see Fig. 2). Let $\mathbf{e}_1, \mathbf{e}_2$ denote unit tangent and normal vector of $\mathbf{c}$ at $\mathbf{c}_0$, respectively. In this Frenet frame, we have $\mathbf{p} = (0, d)$, with $|d|$ being the distance of $\mathbf{p}$ to $\mathbf{c}$. The curvature center $\mathbf{k}_0$ at $\mathbf{c}_0$ has coordinates $(0, \rho)$, where ρ is the inverse curvature $1/\kappa$ and thus has the same sign as the curvature. In that frame, the second order Taylor approximant F_d of the squared distance function at $\mathbf{p}$ is found to be

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

In Fig. 2, the second order Taylor approximant F_d at $\mathbf{p}$ is depicted with some level sets (ellipses). The following special cases should be kept in mind:

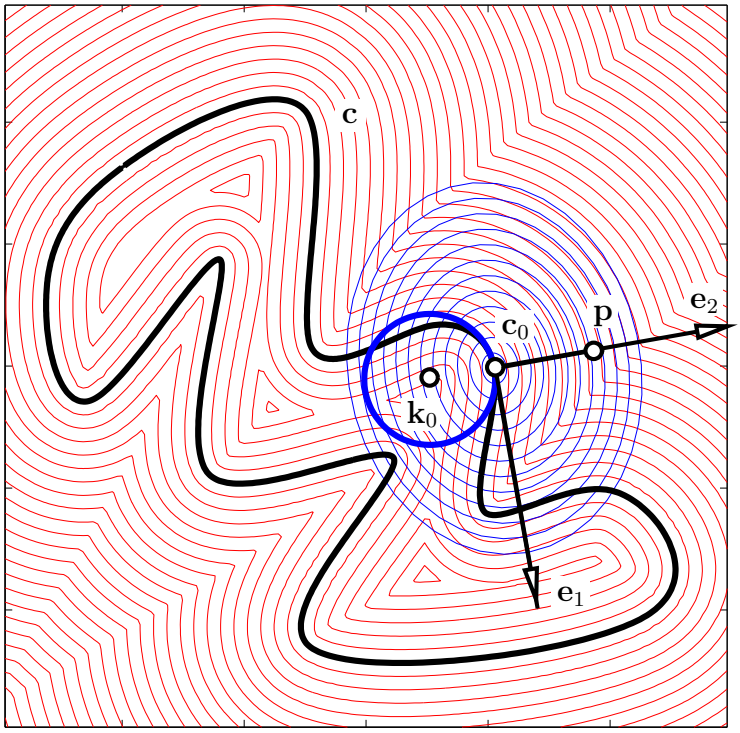


Fig. 2. Planar curve $\mathbf{c}$ with Frenet frame $\mathbf{e}_1$, $\mathbf{e}_2$ in $\mathbf{c}_0$. The squared distance function of the curve $\mathbf{c}$ and the local quadratic approximant of this function in the point $\mathbf{p}$ are visualized by level sets.

- For $d = 0$ we get the Taylor approximant $F_0 = x_2^2$ at the normal foot point. This shows the following interesting result: *At a point* $\mathbf{p}$ *of a curve* $\mathbf{c}$ *the second order approximant of the squared distance function of* $\mathbf{c}$ *and of the curve tangent* T *at* $\mathbf{p}$ *are identical*. Visually, this is not unexpected since curvature depends on the scale. Zooming closer to the curve it appears less and less curved.
- For $d \to \infty$, the Taylor approximant tends to $F_\infty = x_1^2 + x_2^2$. This is the squared distance function to the foot point $\mathbf{c}(t_0)$.

For an implementation which employs the discussed approximants, it is better to express them in the same coordinate system as the curve itself. This is done by viewing F_d as a weighted sum of x_1^2, the squared distance to the normal, and x_2^2, the squared distance to the tangent at the foot point. If $\mathbf{e}_1 \cdot \mathbf{x} + d_1 = 0, \|\mathbf{e}_1\| = 1$ and $\mathbf{e}_2 \cdot \mathbf{x} + d_2 = 0, \|\mathbf{e}_2\| = 1$ are the equations of the normal and the tangent at the foot point $\mathbf{c}_0$, respectively, the quadratic approximant reads

$$F_d(\mathbf{x}) = \frac{d}{d-\rho}(\mathbf{e}_1 \cdot \mathbf{x} + d_1)^2 + (\mathbf{e}_2 \cdot \mathbf{x} + d_2)^2. \tag{4}$$

For the applications we have in mind, it can be important to employ *nonnega-*

tive quadratic approximants to d^2. If the approximant (4) is indefinite, which happens when $A := d/(d-\rho) < 0$, we set A to zero. This means we use the squared distance to the tangent at the foot point.

Analogous considerations can be performed for the squared distance function of a surface $\mathbf{s}$. Given $\mathbf{s}$ and a point $\mathbf{p}$, we compute the closest point $\mathbf{s}_0 \in \mathbf{s}$ to $\mathbf{p}$. At $\mathbf{p}_0$, we use the principal frame, defined by the two principal curvature directions $\mathbf{e}_1, \mathbf{e}_2$ and the surface normal vector $\mathbf{e}_3$. Let κ_i be the (signed) principal curvature to the principal curvature direction $\mathbf{e}_i$, $i = 1, 2$, and let $\rho_i = 1/\kappa_i$. Then the two principal curvature centers at the considered surface point $\mathbf{s}_0$ are expressed in the principal frame as $\mathbf{k}_i = (0, 0, \rho_i)$. It can be shown that the second order Taylor approximant F_d of d^2 at $\mathbf{p} = (0, 0, d)$ is given by

$$F_d(x_1, x_2, x_3) = \frac{d}{d-\rho_1} x_1^2 + \frac{d}{d-\rho_2} x_2^2 + x_3^2. \tag{5}$$

4 Curve and surface approximation using squared distance minimization

4.1 The SDM method with the squared distance field attached to the model shape

As input we consider a model shape M. This can be a curve or surface in any analytical or discrete representation (smoothed mesh or a sufficiently dense point cloud with low noise level). The model shape M shall be approximated by a B-spline curve or surface. We will compute a geometric least squares approximant, where distances are measured *orthogonal to the model shape* M.

For the sake of simplicity in our explanation, we confine ourselves to planar curves, but the concept works for surfaces of arbitrary dimension and codimension in higher dimensional spaces as well.

The method which is proposed here is inspired by active curve models from Computer Vision [4]. An initial B-spline curve is iteratively deformed with an optimization algorithm. The goal is to find a B-spline curve

$$\mathbf{c}(t) = \sum_{i=1}^{n} B_i(t) \mathbf{d}_i, \tag{6}$$

which minimizes the objective function

$$F = \sum_{k=1}^{N} d^2(\mathbf{s}_k, M) + \lambda F_s. \tag{7}$$

Here, $\mathbf{s}_k := \mathbf{c}(t_k), k = 1, \ldots, N$ are curve points at preselected parameter values t_k. These sampled points $\mathbf{s}_k$, called 'sensor points' in the following, must be sufficiently dense so that they describe the shape of the B-spline curve well. The value of λ is a given contant and F_s is a smoothing term, for which we use a combination of L^2 norms of low derivatives of $\mathbf{c}(t)$. Thus the objective function F is the sum of squared distances of the sensor points to the model shape M plus a weighted smoothing term λF_s.

We assume that the basis functions are given; for a B-spline this requires the choice of degree and knot sequence before the optimization is started. The optimization is over the control points $\mathbf{d}_i$. In fact, it is not essential that we use B-splines; any other curve scheme with an expression of the form (6) can be used as well.

From an optimization viewpoint, we have a nonlinear least squares problem [8,13]. The basic optimization procedure is a (stabilized) Newton algorithm, in which we use the local quadratic approximants of the squared distance discussed above. The method will be called *squared distance minimization (SDM)* henceforth. It proceeds as follows:

(1) Initialize the active curve and determine the boundary conditions.
(2) Repeatedly apply the following steps a.–c. until the approximation error or change in the approximation error falls below a predefined threshold:
   a. With the current control points $\mathbf{d}_i$, compute, for $k = 1, \ldots, N$, the active curve point $\mathbf{s}_k = \sum_i B_i(t_k)\mathbf{d}_i$ and a nonnegative local quadratic approximant F_d^k of the squared distance function of the model shape M at the point $\mathbf{s}_k$.
   b. Compute displacement vectors $\mathbf{c}_i,\ i = 1, \ldots, n$, for the control points $\mathbf{d}_i$ by minimizing the function

$$F = \sum_{k=1}^{N} F_d^k[\sum_i B_i(t_k)(\mathbf{d}_i + \mathbf{c}_i)] + \lambda F_s. \tag{8}$$

   F_s is a quadratic function in the unknowns $\mathbf{c}_i$. Since F_d^k are quadratic functions and the argument $\sum_i B_i(t_k)(\mathbf{d}_i + \mathbf{c}_i)$ is linear in $\mathbf{c}_i$, also the first part of F and thus the function F itself is a quadratic function in the displacement vectors $\mathbf{c}_i$ of the control points. Thus, its minimization amounts to the solution of a linear system of equations.
   c. With $\mathbf{c}_i$ from the previous step, we replace the control points $\mathbf{d}_i$ by $\mathbf{d}_i^* = \mathbf{d}_i + \mathbf{c}_i$.

Fig. 3 illustrates the algorithm. The model shape M is a curve which is to be approximated by a B-spline curve. This figure also shows an initial position of the B-spline curve $\mathbf{c}(t)$, with control points $\mathbf{d}_i$, and the updated B-spline curve, with control points $\mathbf{d}_i^*$, after one iteration step. For one of the sample points $\mathbf{s}_k = \mathbf{c}(t_k)$ the local quadratic approximant F_d^k of the squared distance

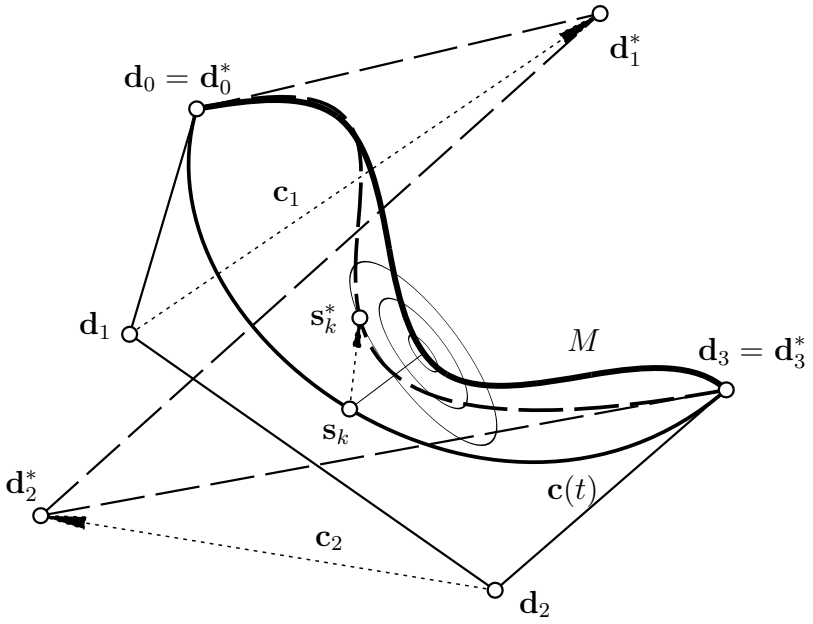


Fig. 3. One step in the curve approximation procedure. The curve M is approximated by a B-spline curve.

function is indicated by three of its level sets, which are concentric ellipses.

There are various issues which need a closer discussion. One has to appropriately preprocess M (or better its distance field), such that one can quickly compute the required local quadratic approximants. Moreover, the adaption of the number of control points (knots in a B-spline model) during the evolution is an important issue. Solutions to these problems are found in [37].

Ongoing research shows that a slight extension of the SDM algorithm can also optimize the weights in the full NURBS model.

4.2 Surface approximation with SDM

The SDM approach to curve approximation has a straightforward extension to surface approximation.

The active surface model we are using shall be of the form $\mathbf{s}(u,v) = \sum B_i(u,v)\mathbf{d}_i$, so that surface points $\mathbf{s}_k$ to given parameter values (u_k, v_k) depend on the control points $\mathbf{d}_i$ in a linear way.

The quadratic function we are minimizing in each iteration step again consists of a distance part, set up via local quadratic approximants of the squared distance function at the sensor points, and a regularization term. For more details, see [23]. An example is presented in Fig. 4. It shows a triangulated CAD surface (data was obtained by 3D laser scanning) and its approximating

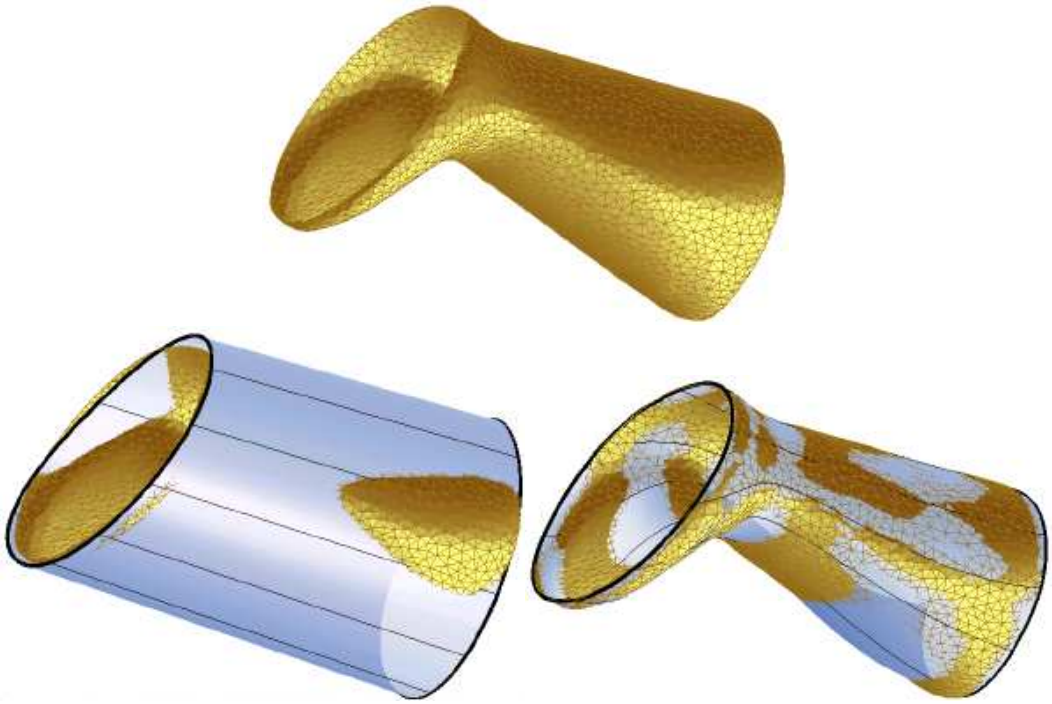


Fig. 4. (Top) Model shape M is a triangle mesh, obtained by 3D laser scanning. (Left) Model shape M and initial position of approximating B-spline surface $\mathbf{s}(u, v)$. (Right) Final B-spline surface $\mathbf{s}(u, v)$.

B-spline surface of bidegree (3,3) with 5x8 control points).

The SDM method is also suitable for approximation with *subdivision surfaces*. An important property of the SDM method is the linear dependence of the sample points $\mathbf{s}_k$ on the control points. For a subdivision surface, surface points also depend linearly on the vertices of the starting mesh of the subdivision procedure. It is therefore possible to optimize the initial mesh so that the sample points well approximate the model shape. Of course, there arise important issues such as the determination of the initial mesh configuration. These are discussed in a recent contribution by Cheng et al. [6]. Figure 5 shows an example of the approximation of a bone structure by a subdivision surface using Loop's scheme.

4.3 Discussion from the viewpoint of optimization

If one uses unmodified second order Taylor approximants F_d^k in the SDM method, the quadratic function (8) is a second order approximant of the objective function F in (7) at the current position (iterate) of the active curve. For smooth model shapes M, the influence parameter λ of the smoothing part is reduced to zero in later steps anyway. Therefore, in this case the algorithm is a *Newton algorithm* and exhibits *local quadratic convergence*.

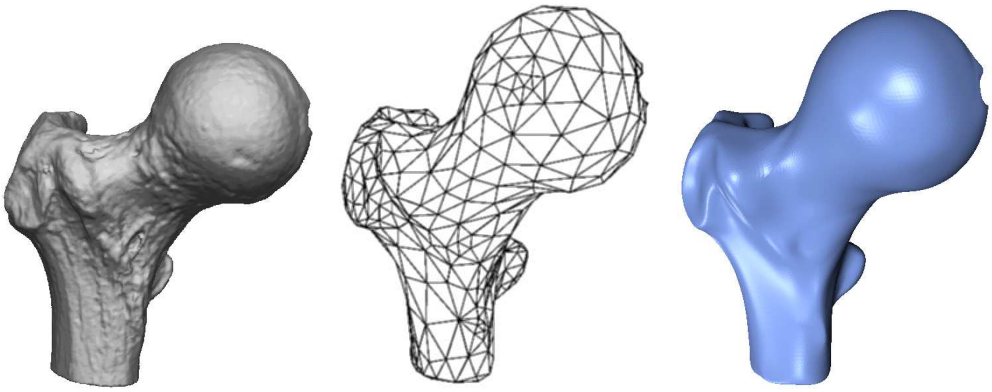


Fig. 5. Approximation with a subdivision surface using Loop's scheme and the SDM method. (Left) Target bone shape, (Middle) control mesh of the subdivision surface, (Right) final subdivision surface obtained by the SDM method

We did, however, suggest to use only nonnegative approximants F_d^k. As a result of this, we do not work with the exact Hessian $\nabla^2 F$ of F, but with a positive definite approximant to it. In this sense, it is a *quasi-Newton algorithm*. Although it is not of a standard type such as BFGS (see e.g. [13]), we expect that one can prove superlinear convergence.

In later steps of the iteration, the sensor points will be very close to M already. Therefore, it is natural to use only the squared distance to the tangent at the foot points of the sensor points as functions F_d^k. This method of *squared tangent distance minimization (TDM)* is exactly a *Gauss Newton iteration* for the solution of the nonlinear least squares problem at hand. Using well-known results from optimization [13] we conclude that TDM exhibits quadratic convergence for a zero residual problem ($F = 0$ at the minimizer, i.e., a spline fits precisely onto the model shape M). TDM converges rapidly for a small residual problem, i.e., if there are sufficiently many control points in the active shape so that it can well approximate the model shape M. Since we have incorporated a regularization term F_s, we have a similar stabilizing effect as in the Levenberg-Marquardt method [13].

Even if we have a positive definite approximate Hessian, a good global convergence behavior would require to check, especially in the initial iteration steps, whether there is sufficient decrease in the value of the objective function F. We propose to apply the following *global convergence improvement of SDM and TDM*: if the new position of the active curve does not have sufficient decrease, one reduces the stepsize and uses as new control points $\mathbf{d}_i + \mu \mathbf{c}_i$, with $\mu < 1$, according to the Armijo rule or a similar stepsize strategy [13]. In Fig. 6 the necessity of a stepsize control is shown for an example where the TDM method was used.

The different behavior of the SDM method and the PDM method for curve approximation can be best visualized with a 3-dimensional plot where the

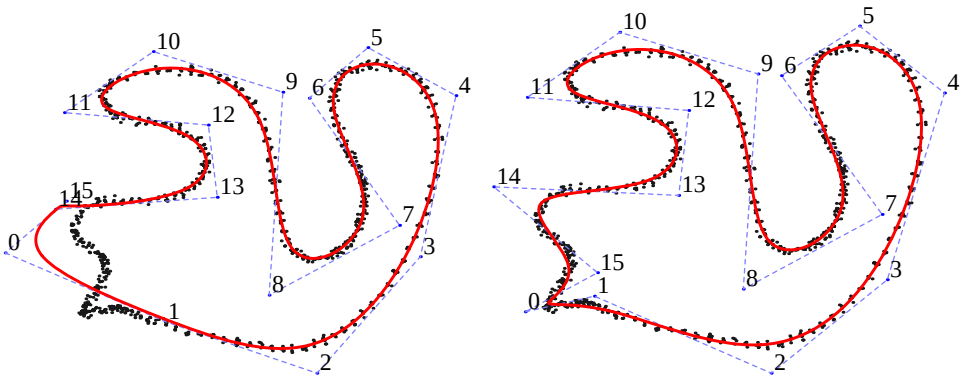


Fig. 6. (Left) The fitting curve generated by TDM without stepsize control. (Right) The fitting curve generated by TDM with stepsize control (Armijo rule).

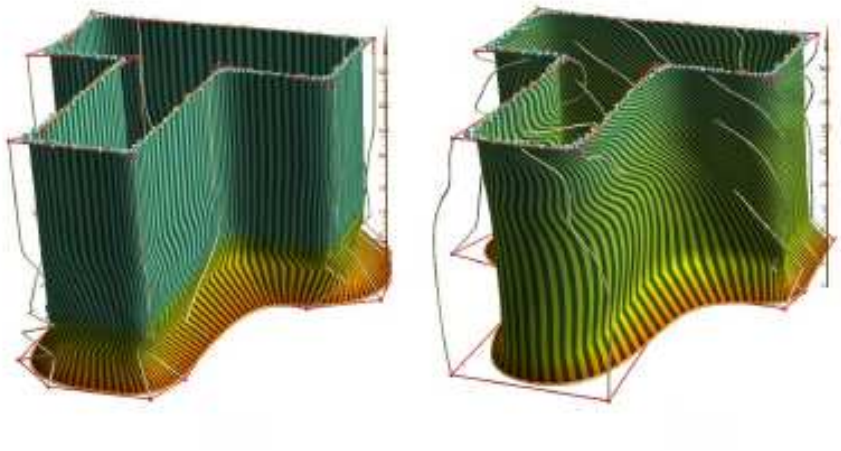


Fig. 7. Curve evolution of (left) SDM method and (right) PDM method.

third dimension represents the time in the curve evolution process, see Fig. 7. The bottom layer of the depicted surface shows the initial position of the active curve, whereas the top layer shows the final shape of the active curve.

In the SDM method the active shape adjusts to the model shape in very few iteration steps, i.e., all the significant changes happen in the lower part of the 3-dimensional plot of Fig. 7(left). In the PDM method, however, the adjusting process needs much more iterations, with mostly tangential movement in the later steps of the algorithm, see Fig. 7(right).

In Fig. 8 we show an extension of the SDM method to an example that requires a global search capability, see Fig. 8. The target shape is the elongated, closed curve at the top level of Fig. 8 and the initial shape is the circular shape at the bottom level. In the evolution process of this curve global properties of the distance function are used, as well as ideas from active contours in image processing.

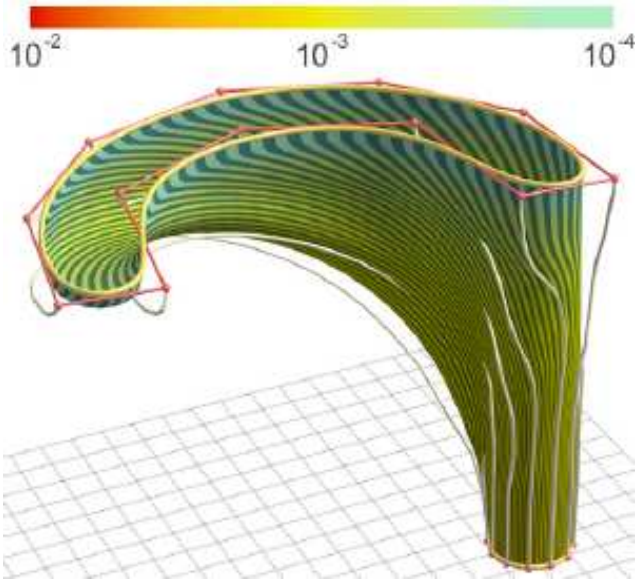


Fig. 8. Curve evolution of SDM method for an example that requires a global search capability.

4.4 The SDM method with the squared distance field attached to the fitting curve

The formulation of SDM given above measures the distance of the active shape and the input data (the model shape M) orthogonal to M. This is fine if M is a smooth curve or a sufficiently dense point sequence with a low noise level. For applications with sparse data points or very noisy measurement data, this approach does not work. In that case, one can measure the error orthogonal to the fitting curve or surface, as proposed in [36].

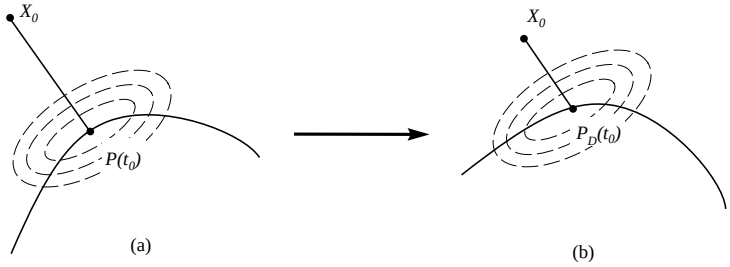


Fig. 9. SDM error measurement orthogonal to the active curve. The SD error function is shown via its iso-value curves (a) before and (b) after the curve has been updated. The change of tangent direction and curvature in one iteration is neglected.

Let us explain this for curves. We have to attach the squared distance field to the active curve. At first sight, this is much more complicated than the previous version. However, it turns out that one can effectively use the following error measurement [36]: at each data point, we compute a nonnegative quadratic

approximant of the squared distance function to the current version of the active curve (Fig. 9, (a)). We then use this quadratic function for measuring the fitting error in the next iteration (Fig. 9, (b)). In this local error metric, points on the same iso-value ellipse have the same approximate squared distance to the fitting curve. The shapes of the ellipses are well adapted to the curve shape. This is one of the reasons why the resulting new SDM method outperforms currently used methods such as the standard CAGD approach based on linear least squares approximation and parameter correction [11,35].

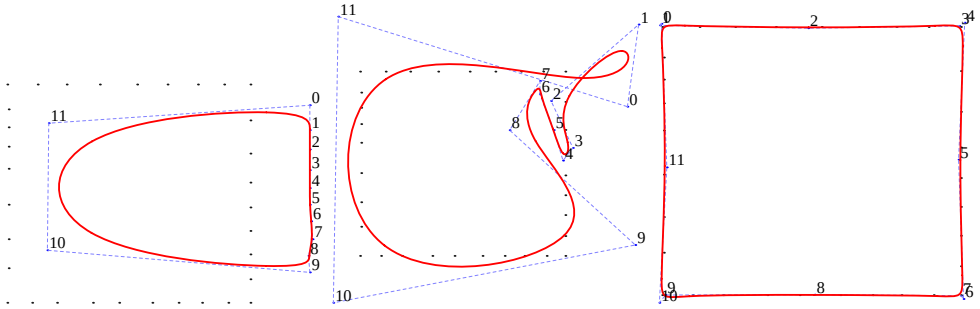
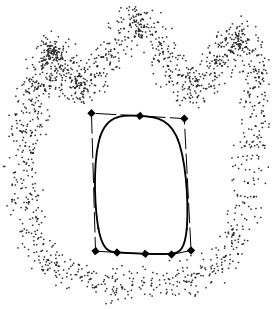


Fig. 10. (a) A target shape (several points arranged in a rectangular shape) and an initial position of the active B-spline curve. (b) The fitted curve generated by point distance minimization (PDM) in 20 iterations. (c) The fitted curve generated by SDM in 20 iterations.

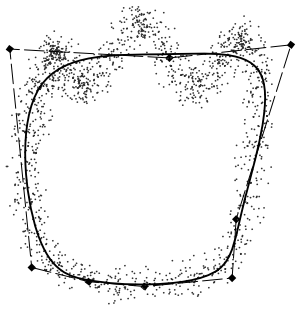
Fig. 10 shows an example for this approach: An active B-spline curve deforms from an initial shape (with a very uneven distribution of its twelve control points) towards a target shape. Two methods are compared, namely PDM (the standard CAGD method of alternation between parameter estimation and linear least squares fitting), and the SDM method. The fact that alternating optimization of parameters and control points is only linearly convergent, and can be improved by Gauss-Newton optimization has already been addressed in [32]. The example in Fig. 6 and a further example in Fig. 11 demonstrate the capability of the SDM method to fit extremely noisy data sets.

4.5 Approximation by ruled surfaces for NC machining and rapid prototyping

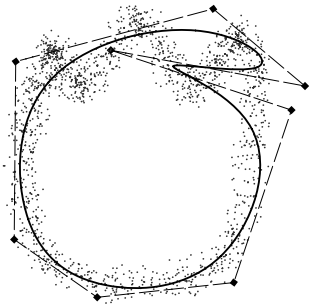
Standard surface approximation methods which require the estimation of the parametrization are hardly applicable in situations where a special parametrization is used to efficiently capture a special surface shape. A good example for that are *ruled surfaces*, which are obtained as B-Spline surfaces $\mathbf{s}(u, v)$ of bidegree $(1, n)$. The u-parameter lines are the straight lines (rulings) on the surface. Approximating a given model shape by a ruled surface has interesting applications in NC machining (peripheral milling with a cylindrical cutter), wire cut electric discharge machining or in architecture (see e.g. [27]). With the SDM method, approximation by a ruled surface becomes a simple task but boundary conditions have to be considered: In the case of a closed surface



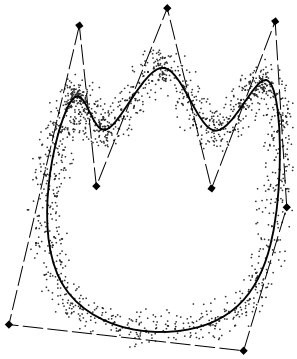
(a) A closed target shape and an initial B-spline curve.



(b) The fitting curve generated by PDM in 50 iterations.



(c) The fitting curve generated by TDM in 50 iterations.



(d) The fitting curve generated by SDM in 50 iterations.

Fig. 11. A comparison of three methods for fitting an extremely noisy target shape.

model M (see e.g. Fig. 12) the initial position of the active surface $\mathbf{s}(u, v)$ has to be chosen outside of M to avoid shrinking of $\mathbf{s}(u, v)$. In the case of an open surface patch M we fix in sufficient distance to M two end rulings of an initial shape and then let the surface flow towards the model shape via SDM. In each iteration, only those sensor points are used whose foot points lie inside M (and not on the boundary).

As noted above, one of the industrial application of surface approximation with ruled surfaces is peripheral milling with a cylindrical cutter [18]: The material is removed by the cutters side, i.e. by the cylindral surface. Thus the milling process generates an offset surface of the ruled surface that is traced out by the moving cutter axis. In order to generate a certain model shape M with this method, using a cutter of radius r, one has to approximate an offset of M at the distance r with a ruled surface and move the cutters axis on this ruled surface. In [16] the authors decompose the model shape M into several parallel strips which are generated with the method of peripheral milling. The ruled layers are then used in a novel rapid prototyping technique. In our context this

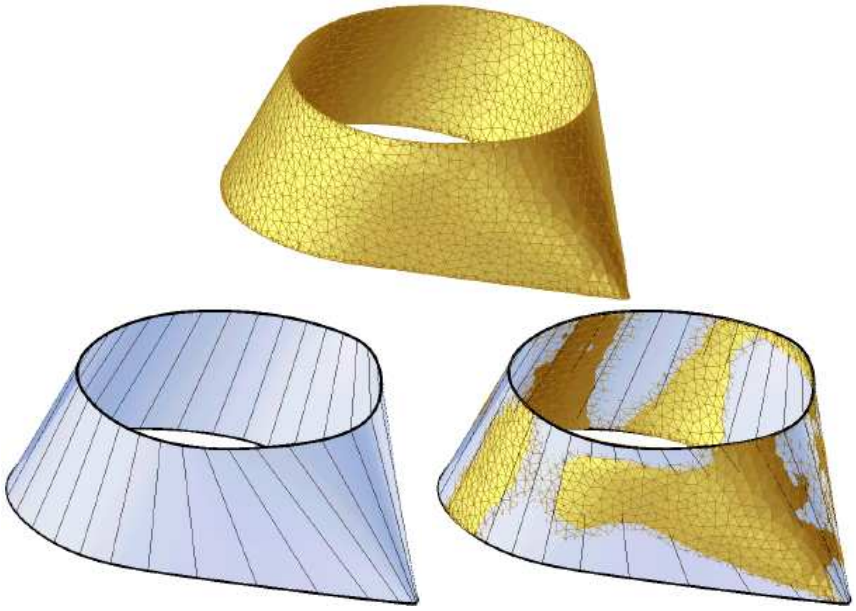


Fig. 12. (Top) Model shape M is a triangle mesh, obtained by 3D laser scanning. (Left) Approximating ruled B-spline surface $\mathbf{s}(u, v)$ of bidegree $(1, 3)$ with 2×10 control points. (Right) Superposition of M and $\mathbf{s}(u, v)$.

problem can be formulated as follows: Approximate the offset surface of our model shape M at distance r with a B-spline surface $\mathbf{s}(u, v)$ of bidegree (1,3) with more than 2 control points in u-direction. This B-spline surface will be composed of several layers of ruled surfaces. It is just a linear side condition in the optimization process to keep the control points of $\mathbf{s}(u, v)$ in layers of parallel planes. Fig. 13(right) shows the result for six layers of ruled surface strips which approximate an offset surface of the model shape M depicted in Fig. 13(left). The middle figure shows an approximation of M itself by such a surface composed of layers of ruled surfaces.

4.6 Approximating level sets by NURBS

The SDM method can be applied to the approximation of an implicitly represented model shape $M : f(\mathbf{x}) = 0$ as well. In fact, SDM implicitizes the model shape M anyway, since it uses the distance function. The level set method is often stabilized by requiring that the level set function f is (close to) a signed distance function. If the output of a level set method is not yet a signed distance function, one can run a few iterations of a solver for the eikonal equation and then achieve a signed distance function. Thus, the output of the level set method as a descriptor of a model shape M is a perfect input for the SDM method. Local quadratic approximants of the squared distance function f^2 can be computed quickly and efficiently from a signed distance function f, even if it is given only on a grid.

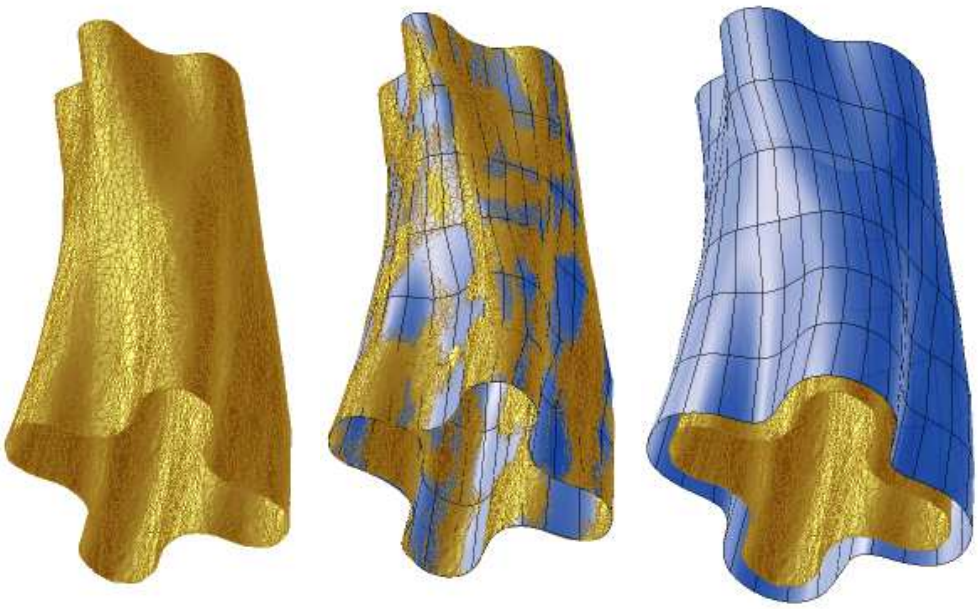


Fig. 13. Ruled layers approximation for rapid prototyping. (Left) Model shape M is a triangle mesh, obtained by 3D laser scanning. (Middle) Model shape M and approximation with a piecewise ruled B-Spline surface, i.e., a surface $\mathbf{s}(u, v)$ of bidegree (1,3) with 7×25 control points. (Right) Approximating B-Spline surface $\mathbf{s}(u, v)$ of bidegree (1,3) approximating an offset of the model shape M.

For a really practical conversion program, one needs an automatic choice of a good initial shape (patch layout) and the incorporation of changes in the patch structure or degrees of freedom (e.g., adding or deleting knots) during the optimization. Note that sharp edges and features should be captured very well, which again requires an appropriate patch layout. This is a topic of current research.

5 Registration based on squared distance minimization

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. This is one instance of a registration problem. Another registration problem concerns the merging of partially overlapping scans of the same object (typically available in different coordinate systems) into a single consistent representation in the same coordinate system.

We will outline an SDM algorithm for the solution of the shape inspection

problem. It involves only two rigid systems (point cloud and CAD model, respectively), but it is fundamental for the entire family of rigid registration problems.

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 [5] 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 [28].

Problem Formulation

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. We will not address those additional issues which come up when only a part of the data shape agrees with a part of the model shape.

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

$$F(\alpha) = \sum_i d^2(\alpha(\mathbf{x}_i^0), \Phi). \tag{9}$$

Here, $d^2(\alpha(\mathbf{x}_i^0), \Phi)$ denotes the squared distance of $\alpha(\mathbf{x}_i^0)$ to Φ. If we view $\alpha: \ \mathbf{x}' = \mathbf{a} + A \cdot \mathbf{x}$ as a special affine map in $\mathbb{R}^3$, 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* [8,13].

In a rather straightforward modification of the SDM method we proceed as follows. Starting with an initial guess, we enter an iteration. In each step, we compute at the current data point positions $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ local quadratic approximants F_i of the squared distance function of the surface Φ. One way of dealing with the rigidity constraints on the moving system is the use of a linearization, i.e., a velocity field. A possible new position $\mathbf{x}_{i,+}$ of a point $\mathbf{x}_i$ is estimated with help of its velocity vector $\mathbf{v}(\mathbf{x}_i) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i$ as

$$\mathbf{x}_{i,+} = \mathbf{x}_i + \mathbf{v}(\mathbf{x}_i) = \mathbf{x}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i. \tag{10}$$

Thus, the estimate for the value of the objective function F after a displacement becomes

$$F_+ = \sum_i F_i(\mathbf{x}_{i,+}) = \sum_i F_i(\mathbf{x}_i + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i). \tag{11}$$

Since the functions F_i are quadratic, F_+ is a quadratic function in the unknown vectors $\mathbf{c}, \bar{\mathbf{c}} \in \mathbb{R}^3$, which characterize the displacement. Hence, minimization of F_+ requires the solution of a linear system in 6 scalar unknowns.

However, we cannot directly move the points $\mathbf{x}_i$ with help of their velocity vectors $\mathbf{v}(\mathbf{x}_i) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i$. This would result in an affine distortion of the moving system. Instead, we compute from the solution $(\mathbf{c}, \bar{\mathbf{c}})$ a helical motion which moves the points $\mathbf{x}_i$ to new positions that are close to $\mathbf{x}_{i,+} = \mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ (for details, see [24,25]). The remark on step size control which we have made in section 4 applies here as well.

In Fig. 14 a set of synthetically generated data points with Gaussian noise is registered to a model of a CAD workpiece. The figure shows the point cloud in its initial position and the final position after 15 iterations.

The present framework contains the two best known algorithms for registration. If we let F_i be the squared distance function to the foot point $\mathbf{y}_i \in \Phi$ of $\mathbf{x}_i$, we obtain an algorithm which is (essentially) the ICP algorithm [2]. If F_i is taken as squared distance function to the tangent plane of Φ at $\mathbf{y}_i$, one obtains the algorithm by Chen and Medioni [5]. Since the data points $\mathbf{x}_i$ in later iterations are very close to Φ, the latter method uses much better approximants than the former (cf. Eq. (5)). In fact, one can show that ICP is essentially a gradient descent method with local linear convergence. The algorithm of Chen and Medioni, the registration analogue to the TDM method, is a Gauss-Newton algorithm and exhibits local quadratic convergence for a zero residual problem. It converges very well also for a small residual problem.

Fig. 14. Registration of a point cloud X to model Φ. X is given in the initial and the aligned position.

The presented SDM registration method based on a linearization of the motion is also just quadratically convergent for a zero residual problem. However, it is not hard to use a second order motion approximant and in this way achieve quadratic convergence even for a larger residual (stronger deviation of the set of data points from Φ). The transition of the presented approach to the simultaneous registration of more than two systems can be performed along the path described in [24]. How much our geometric considerations simplify the Newton approach can be seen in comparison to [34].

6 Image Manifolds for Geometry Processing

Active curves and surfaces as well as registration problems have their origin in Computer Vision and Image Processing. We would like to point here to another concept which comes from this area and is expected to be of great value for Geometric Modelling and CAD. This is the concept of *image manifolds*, which has been introduced by Kimmel, Malladi and Sochen [14]. Given a 2D image, one associates with each point $\mathbf{x} = (x, y)$ in the image plane an auxiliary point

$X = (x, y, f_1, \ldots, f_n)$ in a higher dimensional space, where the f_i's are local image properties such as grey level, color coordinates, texture measures, etc. In this way, the planar image domain is mapped to a two-dimensional surface M embedded in $\mathbb{R}^m$, $m = 2+n$. For certain types of images this manifold M has sufficient smoothness so that it makes sense to apply methods of differential geometry. Depending on the application, one may introduce an appropriate metric in $\mathbb{R}^m$ and use also ideas from pattern classification [7]: if the features f_i have been chosen carefully, image parts with the same local structure will be close to each other in $\mathbb{R}^m$, even if they are not so close in the 2D image itself. Of course, image manifolds can also be associated with volumetric images and movies [14].

It is obvious that one can also handle images defined on a surface $\mathbf{s}$. This might have a number of remarkable applications, also in CAD/CAM. There is also the possibility to define an image which is associated with the underlying surface $\mathbf{s}$ itself. We give an example for such a geometric image manifold [26]: Let us *consider the unit normal vectors as a vector valued image on the surface*. Thus, each surface point $\mathbf{x} \in \mathbf{s}$ with unit normal vector $\mathbf{n}$ is mapped to the point $X = (\mathbf{x}, w\mathbf{n}) \in \mathbb{R}^6$. Here, w is a chosen positive constant. The set of all image points X forms a two-dimensional image manifold $M \subset \mathbb{R}^6$. We now use the canonical Euclidean metric in $\mathbb{R}^6$. It induces on the manifold M a metric, which has remarkable properties when viewed from $\mathbb{R}^3$. *The length of a curve* $\mathbf{c}$ *on* $\mathbf{s}$*, measured in this metric, not only depends on the point set* $\mathbf{c}$ *but also on the variation of the surface normals along* $\mathbf{c}$. We assume that the surface $\mathbf{s}$ has features which are characterized by high curvature (i.e. strong variation of the normals) across the feature, but much smaller curvature along the feature. Then, in this new metric, distances across features appear larger than those along features. We have a *feature sensitive metric*. The constant w guides the sensitivity to features. In other words, w determines what we consider a feature; this is necessary anyway, since features rely on curvatures and thus depend on the scale.

It is not hard to work with the feature sensitive (FS) metric, since FS distances are ordinary Euclidean distances on M. The fact that M lies in $\mathbb{R}^6$ does not cause any major problem for these computations. For details we refer to [26]. Fig. 15 compares the behavior of the distance functions in the ordinary Euclidean metric and in the FS metric. Note that the isolines of the FS distance tend to stop at features.

This new feature sensitive (FS) metric on a surface offers us a variety of applications. Figure 16 shows several geodesic curves we have computed on a parametric surface. Two pairs of input points are each connected with a Euclidean geodesic and a FS geodesic. The latter metric forces the geodesic curves to follow the features of the surface.

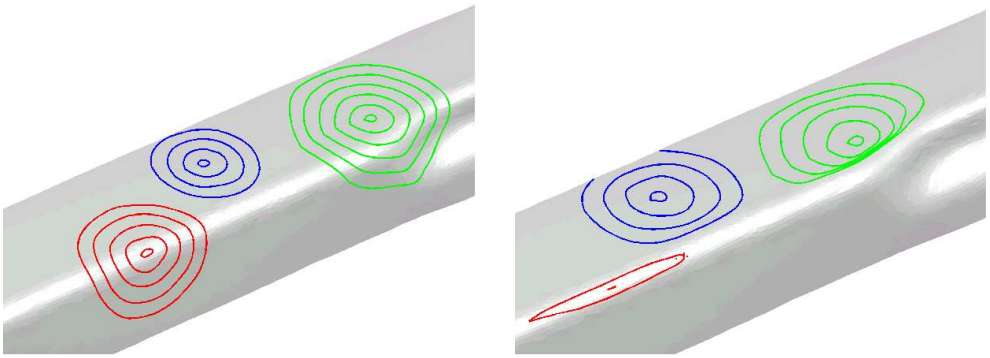


Fig. 15. Approximate geodesic circles on a triangle mesh in the Euclidean metric (left) and feature sensitive metric (right).

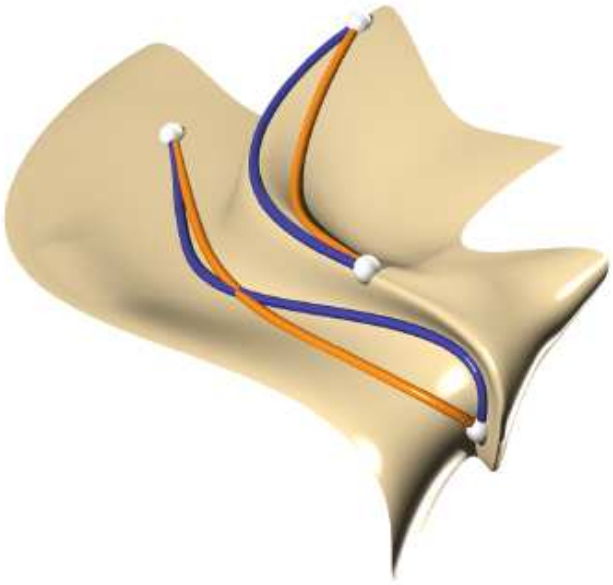


Fig. 16. Geodesic curves on a parametric surface with features: (light colored) computed in the Euclidean metric, i.e. $w = 0$; (dark colored) computed in the FS metric, for $w = 2$.

The original motivation for the introduction of the FS metric comes from *reverse engineering of geometric objects*. There, a variety of shape classification methods have been developed, which aim at a segmentation of the measurement data into regions of the same surface type [35]. Particularly for traditional geometric objects, where most of the surfaces on the boundary of the object are fundamental shapes, the surfaces are often separated by edges or smoothed edges, so-called blending surfaces. Thus, it is natural to look at geometric processing tools on surfaces which are sensitive to such features. The FS metric simplifies the definition of local neighborhoods for shape detection, the implementation of region growing algorithms and the processing of the responses from local shape detection filters (images on surfaces). For example,

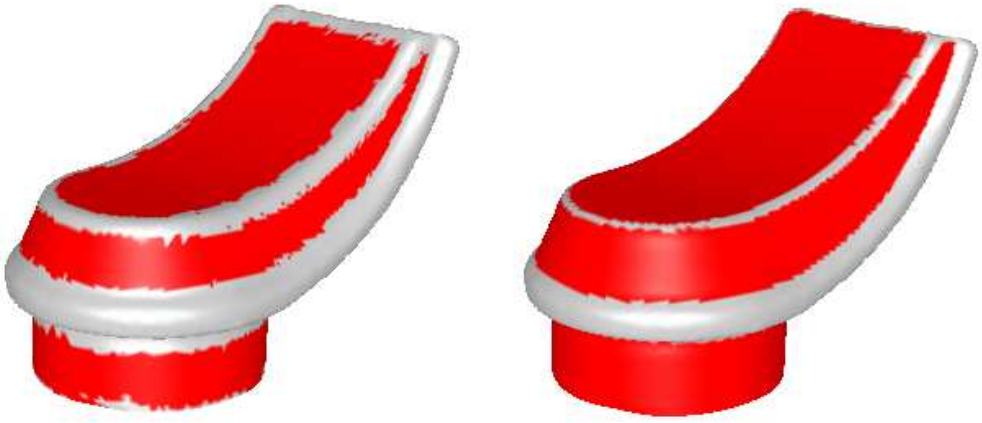


Fig. 17. Dilation in the FS metric with $w = 1$: Starting with the dark regions (left) we get the result shown (right). Note that the FS metric prevents a flow across features.

the neighborhoods of a point shown in Fig. 15 are not equally useful for local shape detection: The neighborhood based on the Euclidean metric (left) flows across the feature. However, the neighborhood based on the FS metric (right) respects the feature and is more likely to belong to the same surface type in an engineering object. Another example is depicted in Fig. 17: One can see the result of an edge detection process (left) which seperates the geometric object into different regions. This result is beautified (right) by region growing with help of the FS distance function. The morphological operation *dilation* in the FS metric can easily be stopped at features.

We expect that the applications of the FS metric go far beyond the examples which have been shown here. Currently, we are exploring feature sensitive surface parameterizations, computed via parameterizations of the image manifold M. Moreover, the global behavior of surface registration algorithms seems to be greatly improvable through the use of appropriately defined image manifolds.

7 Conclusion and Future Research

Exploiting the huge body of knowledge available in various fields that deal with geometric computing, we can search for unifying methods and in this way simultaneously achieve progress for a number of applications. Even just the adaptation of a method known in one field to an application in another field may lead to remarkable progress. This is a basic philosophy behind Industrial Geometry and has been illustrated at hand of optimization problems involving distance functions and concepts taken from Computer Vision and Image Processing (active contours, registration algorithms, image manifolds).

We expect great benefit of CAD from future research in Industrial Geometry. To give just one example, the incorporation of prior knowledge, also shape knowledge, into surface design and reconstruction, could be performed in extension of ideas from Computer Vision and Image Processing. These 'smart surfaces' would certainly be a welcome addition to current design methods.

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 under grant P16002-N05 and by the innovative project '3D Technology' of Vienna University of Technology.

References

[1] Arya S, Mount DM, Netanyahu NS, Silverman S, Wu AY. An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM 1998;45:891–923.

[2] Besl PJ, McKay ND. A method for registration of 3D shapes, IEEE Trans. Pattern Anal. and Machine Intell. 1992;14:239–256.

[3] Bischoff S, Kobbelt L. Parametrization-free active contour models with topology control, The Visual Computer 2004;20:217-228.

[4] Blake A, Isard M. Active Contours. Springer, 1998.

[5] Chen Y, Medioni G. Object modeling by registration of multiple range images, Image and Vision Computing 1992;10:145–155.

[6] Cheng D, Wang W, Qin H, Wong K, Yang H, Liu Y, Fitting subdivision surfaces to unorganized point data using SDM, preprint, University of Hong Kong, 2004.

[7] Duda R O, Hart P E, Stork D G. Pattern Classification, Wiley, New York, 2001.

[8] Fletcher R. Practical Methods of Optimization, Wiley, 1987.

[9] Frisken S, Perry R, Rockwood A, Jones T. Adaptively sampled distance fields: a general representation of shape for computer graphics, Computer Graphics (SIGGRAPH 00 Proceedings), 2000. p. 249–254.

[10] Heijmans HJAM, Morphological Image Operators, Academic Press, Boston, 1994.

[11] Hoschek J, and Lasser D. Fundamentals of Computer Aided Geometric Design, Peters AK, Wellesley, MA., 1993.

[12] Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models, Intl. J. Computer Vision 1987;1(4):321–331.

[13] Kelley CT. Iterative Methods for Optimization, SIAM, 1999.

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

[15] Kobbelt L. Freeform shape representations for efficient geometry processing, 2003. *http://www-i8.informatik.rwth-aachen.de/publications/*

[16] Koc B, Lee Y-S. Adaptive ruled layers approximation of STL models and multi-axis machining applications for rapid prototyping, Journal of Manufacturing Systems 2002;21(3):153–166.

[17] Latombe JC. Robot motion planning, 6th printing, Kluwer, 2001.

[18] Lee Y-S, Koc B. Ellipse-offset approach and inclined zig-zag method for multi-axis roughing of ruled surface pockets, Computer-Aided Design 1998;30:957–971.

[19] Osher S, Fedkiw R. Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2003.

[20] Patrikalakis NM, Maekawa T. Shape Interrogation for Computer Aided Design and Manufacturing, Springer, 2002.

[21] Piegl L, Tiller W. The NURBS Book, Springer, 1995.

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

[23] Pottmann H, Leopoldseder S. A concept for parametric surface fitting which avoids the parametrization problem, Computer Aided Geometric Design 2003;20:343–362.

[24] Pottmann H, Leopoldseder S, Hofer M. Simultaneous registration of multiple views of a 3D object, Intl. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXIV, Part 3A, Commission III; 2002. p. 265–270.

[25] Pottmann H, Leopoldseder S, Hofer M. Registration without ICP, Computer Vision and Image Understanding 2004;95(1):54–71.

[26] Pottmann H, Steiner T, Hofer M, Haider C, Hanbury A. The isophotic metric and its application to feature sensitive morphology on surfaces, In: Pajdla T, Matas J, editors. Computer Vision - ECCV 2004, Part IV, Lecture Notes in Computer Science Vol. 3024, Springer; 2004. p. 560–572.

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

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

[29] Sapiro G. Geometric Partial Differential Equations and Image Analysis, Cambridge Univ. Press, Cambridge, 2001.

[30] Serra J. Image Analysis and Mathematical Morphology, Academic Press, London, 1982.

[31] Sethian JA. Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999.

[32] Speer T, Kuppe M, Hoschek J. Global reparametrization for curve approximation, Computer Aided Geometric Design 1998;15:869–877.

[33] Tsai Y-SR. Rapid and accurate computation of the distance function using grids, J. Comput. Phys. 2002;178(1):175–195.

[34] Tucker TM, Kurfess TR. Newton methods for parametric surface registration. Part I. Theory, Computer Aided Design 2003;35:107–114.

[35] Várady T, Martin R. Reverse Engineering, In: Farin G, Hoschek J, Kim MS, editors. Handbook of Computer Aided Geometric Design, North Holland; 2002. p. 651–681.

[36] Wang W, Pottmann H, Liu Y, Fitting B-Spline curves to point clouds by squared distance minimization. Preprint, University of Hong Kong, 2004.

[37] Yang H, Wang W, Sun J. Control point adjustment for B-spline curve approximation, Computer-Aided Design 2004;36:639–652.

[38] Zhao H-K. A Fast Sweeping Method for Eikonal Equations, Math. Comp., to appear.