Industrial Geometry: Recent Advances and Applications in CAD

Helmut Pottmann[1], Stefan Leopoldseder[1], Michael Hofer[1] and Wenping Wang[2]

[1] Vienna University of Technology, [2] University of Hong Kong
{pottmann,leopoldseder,hofer}@geometrie.tuwien.ac.at
wenping@csis.hku.hk

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 Modeling, Robotics, Computer Vision and Image Processing, Computer Graphics and Scientific Visualization. The methods are mainly taken from classical geometry, computational geometry, CAGD and various branches of applied and industrial mathematics. 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: approximation with general NURBS curves and surfaces, approximation with special surfaces for applications in architecture and manufacturing, and registration problems for industrial inspection and 3D model generation from measurement data.

Keywords: optimization, distance function, curve/surface fitting, active contours, registration.

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 a rather disjunct development. 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 [9], [28]. Computer Vision and Image Processing developed another method, *active contours* [4], [10]. These have been originally 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* [14], [25] 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 [13] 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 solution requires 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 available algorithms are solving nonlinear geometric optimization problems, it is appropriate to study and compare 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 a short conference paper. Therefore, we will focus just on a few topics. We will briefly look at the *level set method* and on *hybrid data structures for geometric computing*. 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 problems.

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 [12], 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
para-metric	triangle mesh	NURBS	subdivision surface
implicit	moving least squares surface	3D grid	octree, binary space partitions

Tab. 1. Basic types of geometry representations.

Explicit representations are meant as sequences of points and can be seen as maps $f\colon N \rightarrow R^3$. Parametric representations are described by maps $f\colon R^2 \rightarrow R^3$ and implicit representations by trivariate functions $f\colon R^3 \rightarrow R$. 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, Computer Aided Design so far focusses 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 R^3 describes it as zero set of a function $f\colon R^3 \rightarrow R$,

$$\Phi := \{\boldsymbol{x} \in R^3 \colon f(\boldsymbol{x})=0\}. \quad (1)$$

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

$$\Phi_c := \{\boldsymbol{x} \in R^3 \colon f(\boldsymbol{x})=c=\text{const.}\}. \quad (2)$$

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* [14], [25]. 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 [14], [23], [25]. 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 avoided the use of the level set method, is the representation it is based on: an implicit representation, evaluated just on a grid. However, we have a variety of complicated shape computation problems where 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 in the system. 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 4.4, 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 $\boldsymbol{x}$ of the embedding space the shortest distance $d(\boldsymbol{x})$ of $\boldsymbol{x}$ 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 for 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. [21]).

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. [9], [15]).

Distance functions are also basic to morphological operators in Image Processing [8],[24]. 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. [15]).

Computer Graphics uses distance functions in many ways, for example in adaptively sampled distance fields [7]. 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 [13]. They also occur in obstacle avoidance with the potential field (barrier) approach [13].

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 $||\nabla f(\boldsymbol{x})||=1$. The main types of algorithms are fast marching [25] and fast sweeping [27], [30]. 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.

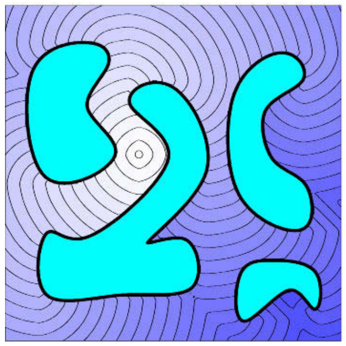

Fig. 1. Level sets of the distance function of a point in the presence of obstacles.

As an example, we consider the computation of the distance function of a geometric object Φ in the presence of obstacles: The function value $d(\boldsymbol{x})$ at a point $\boldsymbol{x}$ (not in an obstacle) is the length of the shortest path from $\boldsymbol{x}$ to a point of Φ, which avoids the obstacles. Zhao's algorithm [30] 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 Φ being a single point $\boldsymbol{p}$ 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 $\boldsymbol{p}$ should be circles. Tsai's algorithm [27] does not have this distortion, but on the other hand it is not easily extendable to the presence of 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.

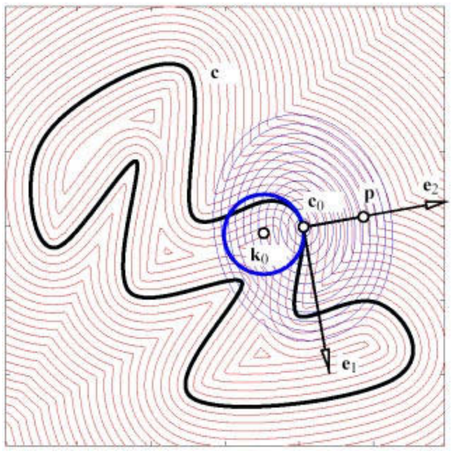

Fig. 2. Planar curve ***c*** with Frenet frame ***e***$_1$, ***e***$_2$ in ***c***$_0$. The squared distance function of the curve ***c*** and the local quadratic approximant of this function in the point ***p*** are visualized by level sets.

Such local quadratic approximants have been studied in [17]. We briefly summarize here the main results and start with the squared distance function $d^2(\mathbf{c})$ of a planar curve ***c***. Deriving a second order approximant only makes sense at a smooth point ***p*** of that function, and thus we exclude points on the cut locus.

Consider an admissible point ***p*** in the plane. The point $\mathbf{c}_0 \in \mathbf{c}$, which is closest to ***p*** is a normal footpoint (see Fig. 2). Let ***e***$_1$, ***e***$_2$ denote unit tangent and normal vector of ***c*** at ***c***$_0$, respectively. In this Frenet frame, we have $\mathbf{p}=(0,d)$, with $|d|$ being the distance of ***p*** to ***c***. The curvature center ***k***$_0$ at ***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 ***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 ***p*** is depicted with some level sets (ellipses). The following special cases should be kept in mind:

1. For $d=0$ we get the Taylor approximant $F_0=x_2^2$ at the normal footpoint. This shows the following interesting result: *At a point **p** of a curve **c** the second order approximant of the squared distance function of **c** and of the curve tangent T at **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.
2. For d $\rightarrow$ ∞, the Taylor approximant tends to $F_\infty=x_1^2+x_2^2$. This is the squared distance function to the footpoint $\mathbf{c}(t_0)$.

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 footpoint. If $\mathbf{e}_1 \cdot \mathbf{x} + d_1 = 0$ and $\mathbf{e}_2 \cdot \mathbf{x} + d_2 = 0$ are the Hesse normal forms of normal and tangent at the footpoint ***c***$_0$, respectively, the quadratic approximant reads

$$F_d(x) = \frac{d}{d-\rho}(e_1 \cdot x + d_1)^2 + (e_2 \cdot x + d_2)^2. \tag{4}$$

For the applications we have in mind, it can be important to employ *nonnegative* quadratic approximants to d^2. If the approximant (4) is indefinite, which happens for $A := d/(d-\rho) < 0$, we set A to zero. This means we use the squared distance to the tangent at the footpoint.

Analogous considerations can be performed for the squared distance function of a surface ***s***. Given ***s*** and a point ***p***, we compute the closest point $\mathbf{s}_0 \in \mathbf{s}$ to ***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 ***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. APPROXIMATION WITH NURBS CURVES AND SURFACES USING SQUARED DISTANCE MINIMIZATION

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 within an optimization algorithm. The goal is to find a B-spline curve

$$c(t) = \sum_{i=1}^{n} B_i(t) d_i \tag{6}$$

which minimizes the objective function

$$F = \sum_{k=1}^{N} d^2(s_k, M). \tag{7}$$

Here, $\boldsymbol{s}_k := \boldsymbol{c}(t_k)$, $k=1,\ldots,N$ are curve points at preselected parameter values t_k. These sampled points $\boldsymbol{c}_k$, called 'sensor points' in the following, must be sufficiently dense so that they describe the shape of the B-spline curve well. The objective function F is the sum of squared distances of the sensor points to the model shape M.
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 $\boldsymbol{d}_i$. 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 [6], [11]. The basic optimization procedure is a (stabilized) Newton algorithm, in which we use the local quadratic approximants of the squared distance discussed above. 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 $\boldsymbol{d}_i$, compute, for $k=1,\ldots,N$, the active curve point $\boldsymbol{s}_k = \Sigma_i B_i(t_k)\boldsymbol{d}_i$ and a nonnegative local quadratic approximant F_d^k of the squared distance function of the model shape M at the point $\boldsymbol{s}_k$.
   b. Compute displacement vectors $\boldsymbol{c}_i$, $i=1,\ldots,n$, for the control points $\boldsymbol{d}_i$ by minimizing the function
   $$F = \sum_{k=1}^{N} F_d^k[\sum_i B_i(t_k)(d_i + c_i)] + \lambda F_s \tag{8}$$
   Here, λ is a given constant and F_s is a smoothing term, for which we use a combination of L^2 norms of low order derivatives of $\boldsymbol{c}$. F_s is a quadratic function in the unknowns $\boldsymbol{c}_i$. Since F_d^k are quadratic functions and the argument $\Sigma_i B_i(t_k)(\boldsymbol{d}_i+\boldsymbol{c}_i)$ is linear in $\boldsymbol{c}_i$, also the first part of F and thus the function F itself is a quadratic function in the displacement vectors $\boldsymbol{c}_i$ of the control points. Thus, its minimization amounts to the solution of a linear system of equations.
   c. With $\boldsymbol{c}_i$ from the previous step, we replace the control points $\boldsymbol{d}_i$ by $\boldsymbol{d}_i^* = \boldsymbol{d}_i + \boldsymbol{c}_i$.

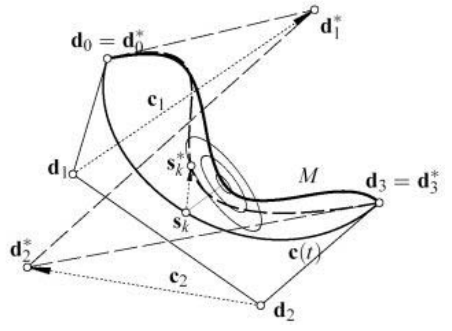

Fig. 3. One step in the curve approximation procedure. The curve M is approximated by a B-spline curve.

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 $\boldsymbol{c}(t)$, with control points $\boldsymbol{d}_i$, and the updated B-spline curve, with control points $\boldsymbol{d}_i^*$, after one iteration step. For one of the sample points $\boldsymbol{s}_k = \boldsymbol{c}(t_k)$ the local quadratic approximant F_d^k of the squared distance 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 topics are found in [29]. The authors of that paper call the method *squared distance minimization (SDM)*.

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

4.1 Approximation with an Active Surface

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 $\boldsymbol{s}(u,v) = \Sigma\ B_i(u,v)\boldsymbol{d}_i$, so that surface points $\boldsymbol{s}_k$ to given parameter values (u_k, v_k) depend on the control points $\boldsymbol{d}_i$ 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 [18]. An example is presented in Fig. 4. It shows a triangulated CAD surface (data was obtained by 3D laser scanning) and its approximating B-spline surface of bidegree (3,3) with 5x8 control points).

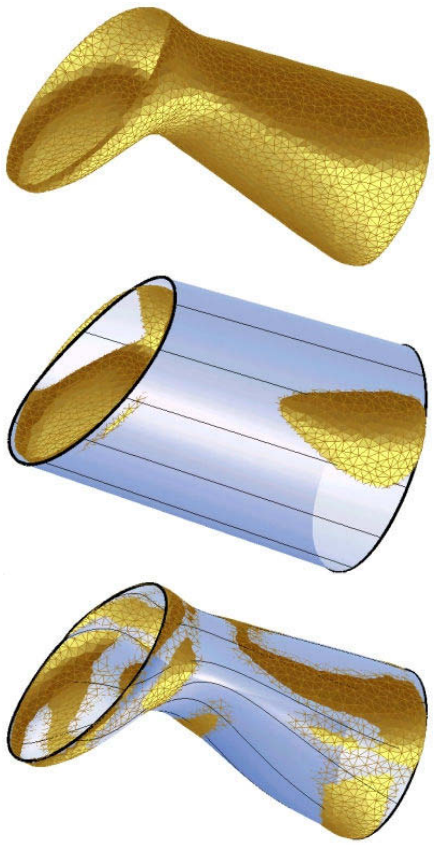
Fig. 4. (Top) Model shape M is a triangle mesh, obtained by 3D laser scanning. (Middle) Model shape M and initial position of approximating B-spline surface $\boldsymbol{s}(u,v)$. (Bottom) Final B-spline surface $\boldsymbol{s}(u,v)$.

4.2 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) – without the smoothing term – 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*.

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, 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 footpoints 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 [11] 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 [11].

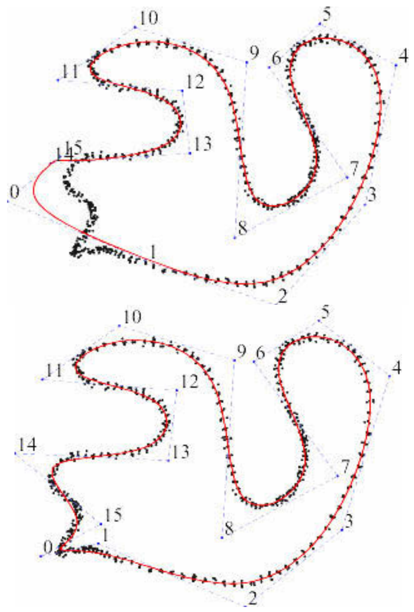

Fig. 5. (Top) The fitting curve generated by TDM without step size control. (Bottom) The fitting curve generated by TDM with step size control (Armijo rule).

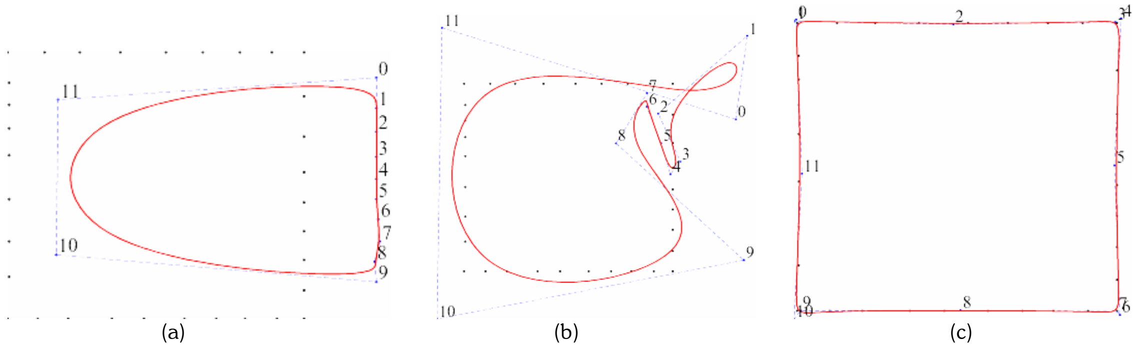

Fig. 6. (a) A target shape (several points arranged in a rectangular shape) and an initial position of the active B-spline curve. (b) The fitted curve generated by point distance minimization (PDM) in 20 iterations. (c) The fitted curve generated by SDM in 20 iterations.

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*: if the new position of the active curve does not have sufficient decrease, one reduces the stepsize and uses as new control points $\boldsymbol{d}_i + \mu\boldsymbol{c}_i$, with $\mu < 1$, according to the Armijo rule or a similar stepsize strategy [11]. In Fig. 5 the necessity of a stepsize control is shown for an example where the the TDM method was used.

The present formulation of SDM 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 has to measure the error *orthogonal to the active shape*. Thus, we have to attach the squared distance field to the active curve. At first sight, this is much more complicated than the present version.

However, ongoing research shows that this approach can be implemented in an efficient algorithm which outperforms currently used methods such as the standard CAGD approach based on linear least squares approximation and parameter correction [9], [28]. Fig. 6 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 [26].

4.3 Reconstruction of special surfaces

Standard surface approximation methods which require the estimation of the parameterization are hardly applicable in situations where a special parametrization is used to efficiently capture a special surface shape.

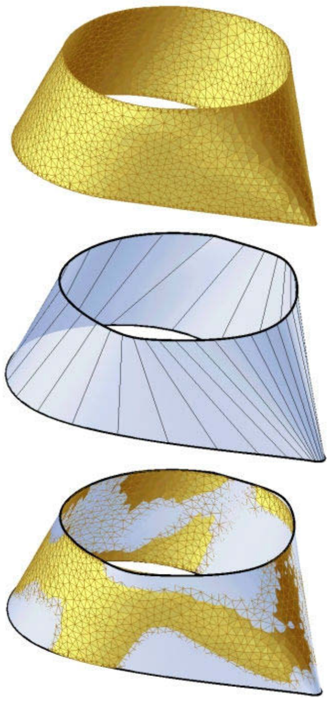

Fig. 7. (Top) Model shape *M* is a triangle mesh, obtained by 3D laser scanning. (Middle) Approximating ruled B-spline surface $\boldsymbol{s}(u,v)$ of bidegree (1,3) with 2×10 control points. (Bottom) Superposition of M and $\boldsymbol{s}(u,v)$.

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. [21]). 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 model M (see e.g. Fig. 7) 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 footpoints lie inside M (and not on the boundary).

4.4 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, we see that 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.

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 [22].

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

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

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 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* [6], [11].

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 $(\boldsymbol{x}_1,\boldsymbol{x}_2,\dots)$ 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 $\boldsymbol{x}_{i,+}$ of a point $\boldsymbol{x}_i$ is estimated with help of its velocity vector $\boldsymbol{v}(\boldsymbol{x}_i) = \underline{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{x}_i$ as

$$\boldsymbol{x}_{i,+} = \boldsymbol{x}_i + \boldsymbol{v}(\boldsymbol{x}_i) = \boldsymbol{x}_i + \underline{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{x}_i. \qquad (10)$$

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

$$F_+ = \Sigma_i F_i(\boldsymbol{x}_{i,+}) = \Sigma_i F_i(\boldsymbol{x}_i + \underline{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{x}_i). \qquad (11)$$

Since the functions F_i are quadratic, F_+ is a quadratic function in the unknown vectors $\boldsymbol{c}$, $\underline{\boldsymbol{c}} \in \mathbb{R}^3$, 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 $\boldsymbol{x}_i$ with help of their velocity vectors $\boldsymbol{v}(\boldsymbol{x}_i) = \underline{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{x}_i$. This would result in an affine distortion of the moving system. Instead, we compute from the solution $(\boldsymbol{c}, \underline{\boldsymbol{c}})$ a helical motion which moves the points $\boldsymbol{x}_i$ to new positions that are close to $\boldsymbol{x}_{i,+} = \boldsymbol{x}_i + \boldsymbol{v}(\boldsymbol{x}_i)$ (for details, see [19], [20]). The remark on stepsize control which we have made in Sect. 4 applies here as well.

In Fig. 8 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.

This concept contains the two best known algorithms for registration. If we let F_i be the squared distance function to the footpoint $\boldsymbol{y}_i \in \Phi$ of $\boldsymbol{x}_i$, 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 $\boldsymbol{y}_i$, one obtains the algorithm by Chen and Medioni [5]. Since the data points $\boldsymbol{x}_i$ in later iterations are very close to Φ, the latter method uses much better approximants than the former (cf. Eqn. (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.

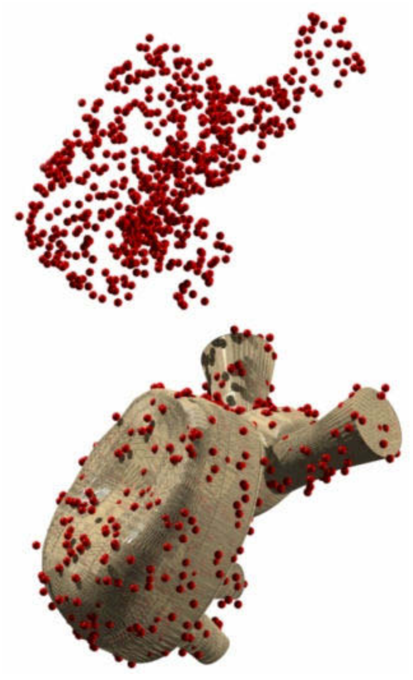
Fig. 8. Registration of a point cloud X to model Φ. X is given in the initial and the aligned position.

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 [19].

6. 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. This is a basic philosophy behind Industrial Geometry and has been illustrated at hand of optimization problems involving distance functions. 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.

7. REFERENCES

[1] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, S. and Wu, A., Y., An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM, Vol. 45, 1998, pp 891–923.
[2] Besl, P. J. and McKay, N. D., A method for registration of 3D shapes, IEEE Trans. Pattern Anal. and Machine Intell., Vol. 14, 1992, pp 239–256.
[3] Bischoff, S. and Kobbelt, L., Snakes with topology control, *The Visual Computer*, to appear.
[4] Blake, A. and Isard, M., Active Contours, Springer, 1998.
[5] Chen, Y., and Medioni, G., Object modeling by registration of multiple range images, *Image and Vision Computing,* Vol 10, 1992, pp 145–155.
[6] Fletcher, R., Practical Methods of Optimization, Wiley, 1987.
[7] Frisken, S., Perry, R., Rockwood, A., and Jones, T., Adaptively sampled distance fields: a general representation of shape for computer graphics, Computer Graphics (Proceedings of ACM SIGGRAPH 00), 2000, pp 249–254.
[8] Heijmans, H. J. A. M., *Morphological Image Operators*, Academic Press, Boston, 1994.
[9] Hoschek, J. and Lasser, D., *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, Wellesley, MA, 1993.
[10] Kass, M., Witkin, A. and Terzopoulos, D., Snakes: Active contour models, *Intern. J. Computer Vision*, Vol. 1, 1987, pp 321–331.
[11] Kelley, C. T., *Iterative Methods for Optimization*, SIAM, 1999.
[12] Kobbelt, L., Freeform shape representations for efficient geometry processing, 2003. http://www-i8.informatik.rwth-aachen.de/publications/
[13] Latombe, J. C., *Robot motion planning*, 6th printing, Kluwer, 2001.
[14] Osher, S. and Fedkiw, R., *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, 2003.
[15] Patrikalakis, N. M. and Maekawa, T., *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer, 2002.
[16] Piegl, L. and Tiller, W., *The NURBS Book*, Springer, 1995.
[17] Pottmann, H. and Hofer, M., Geometry of the squared distance function to curves and surfaces, in *Visualization and Mathematics III*, Hege, H.-C., and Polthier, K., eds., Springer, 2003, pp 221–242.
[18] Pottmann, H. and Leopoldseder, S., A concept for parametric surface fitting which avoids the parametrization problem, *Computer Aided Geometric Design*, Vol. 20, 2003, pp 343–362.
[19] Pottmann, H., Leopoldseder, S. and 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, pp 265–270.
[20] Pottmann, H., Leopoldseder, S. and Hofer, M., Registration without ICP, *Computer Vision and Image Understanding,* to appear.
[21] Pottmann, H., and Wallner, J., *Computational Line Geometry*, Springer-Verlag, 2001.
[22] Rusinkiewicz, S. and Levoy, M., Efficient variants of the ICP algorithm, in Proc. 3rd Int. Conf. on 3D Digital Imaging and Modeling, Quebec, 2001.
[23] Sapiro, G., *Geometric Partial Differential Equations and Image Analysis*, Cambridge Univ. Press, Cambridge, 2001.
[24] Serra, J., *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
[25] Sethian, J. A., *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1999.
[26] Speer, T., Kuppe, M. and Hoschek, J., Global reparametrization for curve approximation, Computer Aided Geometric Design, Vol. 15, 1998, pp 869–877.
[27] Tsai, Y.-S. R., Rapid and accurate computation of the distance function using grids, *J. Comput. Phys.,* Vol. 178, No. 1, 2002, pp 175–195.
[28] Várady, T. and Martin, R., Reverse Engineering, in: *Handbook of Computer Aided Geometric Design*, Farin, G., Hoschek, J., and Kim, M.S., eds., North Holland, 2002, pp 651–681.
[29] Yang, H., Wang, W. and Sun, J., Control point adjustment for B-spline curve approximation, *Computer Aided Design*, Vol. 36, 2004, pp 639-652.
[30] Zhao, H.-K., A Fast Sweeping Method for Eikonal Equations, *Math. Comp.*, to appear.