On Surface Approximation using Developable Surfaces

H.-Y. Chen, I.-K. Lee, S. Leopoldseder,
H. Pottmann, T. Randrup*), J. Wallner

Institut für Geometrie, Technische Universität Wien
Wiedner Hauptstraße 8–10, A-1040 Wien, Austria

*Odense Steel Shipyard Ltd., P.O. Box 176,
DK-5100 Odense C, Denmark

March 26, 2004

Abstract

We introduce a method for approximating a given surface by a developable surface. It will be either a G^1 surface consisting of pieces of cones or cylinders of revolution or a G^r NURBS developable surface. Our algorithm will also deal properly with the problems of reverse engineering and produce robust approximation of given scattered data. The presented technique can be applied in computer aided manufacturing, e.g. in shipbuilding.

Keywords: computer aided design, computer aided manufacturing, surface approximation, reverse engineering, surface of revolution, developable surface, shipbuilding.

1 Introduction

A developable surface is a surface which can be unfolded (developed) into a plane without stretching or tearing. Therefore, developable surfaces possess a wide range of applications, for example in sheet-metal and plate-metal based industries (see e.g. [1, 2]).

It is well known in elementary differential geometry [3] that under the assumption of sufficient differentiability, a developable surface is either a plane, conical surface, cylindrical surface or tangent surface of a curve or a composition of these types. Thus a developable surface is a ruled surface, where all points of the same generator line share a common tangent plane. The rulings are principal curvature lines with vanishing normal curvature and the Gaussian curvature vanishes at all surface points. Therefore developable surfaces are also called single-curved surfaces, as opposed to double-curved surfaces.

One motivation for our work comes from the design and engineering of the double-curved ship surfaces project, launched by Odense Steel Shipyard Ltd. and the Department of Mathematics at the Technical University of Denmark [4]. A ship surface is designed by segmenting it into parts in each a single steel plate will be placed. These steel plates are shaped in two processes, *rolling* and *heating*. Rolling produces a developable surface, mostly of cylindrical shape. Most of the ship's steel plates are located in a single-curved area, where rolling is sufficient. In double-curved areas, e.g. at the bow, however, the surface is far away from being developable. To obtain the desired double-curved shape, after the process of rolling the plate is shrunk along its longer edges using pattern-heating. The parameters of the heating process are determined merely on the basis of experience and heuristics. Thus heating is much more difficult to control than rolling, resulting in the problem that inaccurately manufactured plates do not fit well together [5]. Our approach to improve this situation is the following. In order to minimize the use of heating, we perform an appropriate decomposition of the ship surface into patches each of which may be well approximated by a cone or cylinder surface [6].

Some shipyards have been using developable surfaces only for design. Both for the strategy above and the design with developable surfaces only, the solution of the following problem plays an important role: Given a surface or a set of data points of a surface, e.g. coming from a double-curved area of a ship surface, approximate the data by a developable surface.

Most CAGD research on developable surfaces has been focused on the construction of developable surfaces to be used in CAD-systems (see [7] and the references therein). Surface approximation by developable surfaces is addressed in several con-

tributions [9, 10, 11, 12, 13, 14], but all these papers either do not allow our input data or the methods they describe could not perform very well due to limitations.

In this paper we introduce a new technique for the approximation problem stated above. It is guided by a well known result: A developable surface possesses for each generator line a cone of revolution (right circular cone) which has second order contact with the surface at all points of the generator [3]. This so-called *osculating cone* is a counterpart to the osculating circle of a curve. It may degenerate to a cylinder of revolution or to a plane, which are limit cases that shall be tacitly admitted in the following. The osculating cone approximates the surface well in some region neighboring the chosen generator, a fact which gives us the idea of segmenting the given data points into regions which can be well approximated by cones or cylinders of revolution. Afterwards our algorithm will connect these elements by patches of cones or cylinders of revolution with G^1 join or calculate an approximating developable G^r $(r \geq 2)$ NURBS surface from these shaping elements. This is done by employing the methods developed by Leopoldseder and Pottmann [15] and Pottmann and Wallner [12], respectively.

We also consider that the given data may have been scanned from a real existing object so that the input is a point cloud subject to measurement errors. The method proposed will be robust to noise and will handle outliers. Thus our surface approximation technique can also be used for reconstructing geometric models. The procedure of creating a CAD model of a real object as input for CAD/CAM systems is called *reverse engineering*. For an introduction into the basic concepts of reverse engineering and a survey of the state of the art we refer the reader to Varady et al. [16].

The paper is organized as follows. We first briefly describe some basic algorithms for approximation by surfaces of revolution introduced by Pottmann and Randrup [17] and Pottmann, Chen and Lee [18]. Then we refine these algorithms to an approximation method for cones and cylinders of revolution. This procedure will find a good fitting cone or cylinder in an appropriate region. We then describe a type of region growing procedure for the computation of a good segmentation of the given data into regions which are well approximated by one cone or cylinder of revolution. After this, we shortly describe how to smoothly join these geometric elements to a G^1 [15] or G^r $(r \geq 2)$ surface [12]. We conclude the paper with some examples illustrating our method.

2 Approximation by surfaces of revolution

First let us discuss how to compute the axis of a cone of revolution well approximating a given data set. Following the approach proposed by Pottmann and Randrup [17] and Pottmann, Chen and Lee [18] we consider the cones of revolution as a subclass of the surfaces of revolution and these as a subclass of the helical surfaces. So we will deal with approximation by helical surfaces and specialize to surfaces of revolution and cones of revolution later. This might seem as an unnecessary complication, but it not only gives theoretical background information, but is in fact only a simpler version of the approximation by a surface of revolution with one constraint less. We now give a short outline of the algorithm.

A continuous motion in 3–dimensional Euclidean space $\mathbb{R}^3$, composed of a rotation around an axis A and a proportional translation parallel to A is called a *helical motion* or *screw motion*. A surface generated by sweeping a curve along a helical motion is called a *helical surface*. The proportionality factor p of the velocities of the rotational and translational part is called the *pitch* of the helical motion. In the special case of a pure rotation where the pitch p vanishes, we obtain a *surface of revolution*.

As is well known (cf. [19]), a helical motion can be characterized by a pair $\mathbf{C} = (\mathbf{c}, \bar{\mathbf{c}}) \in \mathbb{R}^6$. The velocity vector of the point $\mathbf{x}$ is given by:

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}.$$

We consider a given set of data points $\mathbf{d}_i$ and we assume that for each $\mathbf{d}_i$ we have already found a surface normal $\mathbf{n}_i$, either by numerical estimation or by other means. For the estimation of normal vectors, see e.g. [8]. To find a well approximating helical surface, we are looking for a motion such that the velocity vectors $\mathbf{v}(\mathbf{d}_i)$ at data points $\mathbf{d}_i$ form an angle γ_i close to $\pi/2$ with the normals n_i. Because of

$$\cos\gamma_i = \frac{\mathbf{n}_i \cdot \mathbf{v}(\mathbf{d}_i)}{\|\mathbf{v}(\mathbf{d}_i)\|} = \frac{\bar{\mathbf{c}} \cdot \mathbf{n}_i + \mathbf{c} \cdot \bar{\mathbf{n}}_i}{\|\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{d}_i\|},$$

with $\bar{\mathbf{n}}_i := \mathbf{d}_i \times \mathbf{n}_i$, the minimization of

$$G := \sum_{i=1}^{k} \cos^2\gamma_i \tag{1}$$

is a nonlinear problem.

Thus in [17] we proposed to minimize the positive semidefinite quadratic form

$$F(\mathbf{C}) := \sum_{i=1}^{k} (\bar{\mathbf{c}} \cdot \mathbf{n}_i + \mathbf{c} \cdot \bar{\mathbf{n}}_i)^2 =: \mathbf{C}^T \cdot N \cdot \mathbf{C} \tag{2}$$

under the normalization condition

$$1 = \|\mathbf{c}\|^2 =: \mathbf{C}^T \cdot D \cdot \mathbf{C}, \tag{3}$$

where $D = \mathrm{diag}(1, 1, 1, 0, 0, 0)$. Equation (3) normalizes the rotational part of the motion to 1. This excludes pure translations and thus approximation with cylinders. The solution of $F \to \min$ is a well known general eigenvalue problem. Using a Lagrangian multiplier λ, we have to solve the system of equations

$$(N - \lambda D) \cdot \mathbf{C} = \mathbf{0},\ \mathbf{C}^T \cdot D \cdot \mathbf{C} = 1. \tag{4}$$

Hence, λ must be a root of the equation

$$\det(N - \lambda D) = 0, \tag{5}$$

which is cubic in λ. For any root λ and the corresponding normalized general eigenvector $\mathbf{C}$, we have $F(\mathbf{C}) = \lambda$. Therefore, all roots λ are nonnegative and the *solution* $\mathbf{C}$ *is a general eigenvector to the smallest general eigenvalue* $\lambda \geq 0$. The axis $A = (\mathbf{a}, \bar{\mathbf{a}})$ and the pitch p now are computed as follows:

$$\mathbf{a} = \frac{\mathbf{c}}{\|\mathbf{c}\|},\ \bar{\mathbf{a}} = \frac{\bar{\mathbf{c}} - p\mathbf{c}}{\|\mathbf{c}\|},\ p = \frac{\mathbf{c} \cdot \bar{\mathbf{c}}}{\mathbf{c}^2}. \tag{6}$$

The vector $\mathbf{a}$ gives the direction of the axis and $\bar{\mathbf{a}}$ is the cross product of a point on the axis with $\mathbf{a}$ (cf. [18]).

If p is small compared to the size of the object, we better approximate with a surface of revolution. Let us, as is done in [18], impose $p = 0$ as a further constraint. This amounts to the minimization of (2) under the conditions (3) and

$$0 = \mathbf{c} \cdot \bar{\mathbf{c}} =: \mathbf{C}^T \cdot K \cdot \mathbf{C}. \tag{7}$$

With two Lagrangian multipliers λ, μ, we have to solve the system of equations

$$(N - \lambda D - \mu K) \cdot \mathbf{C} = \mathbf{0},\ \mathbf{C}^T \cdot D \cdot \mathbf{C} = 1,\ \mathbf{C}^T \cdot K \cdot \mathbf{C} = 0. \tag{8}$$

Hence, λ, μ are restricted to the algebraic curve S of order 6 given by

$$\det(N - \lambda D - \mu K) = 0. \tag{9}$$

Since the polynomial degree of this equation is too high as shown in [18], we compute the solution numerically by a Newton–type iteration. A good starting point for the iteration is a solution $\mathbf{C}$ for $\mu = 0$, i.e., a solution of the system (4).

For more details the reader is referred to [17] and [18].

3 A refined algorithm for approximation by cones of revolution

Suppose the given data points to be approximated are close to a cone of revolution. We expect that the data normals will have a nearly constant angle with the unknown axis, since the angle between the surface normals of a cone of revolution and its axis is constant.

Thus instead of minimizing the positive semidefinite quadratic form $F(\mathbf{C})$ we consider the quadratic form

$$F_q(\mathbf{C}, g) := \sum_{i=1}^{k} (\bar{\mathbf{c}} \cdot \mathbf{n}_i + \mathbf{c} \cdot \bar{\mathbf{n}}_i)^2 + q \sum_{i=1}^{k} (\mathbf{c} \cdot \mathbf{n}_i - g)^2 =: \mathbf{X}^T \cdot N_q \cdot \mathbf{X}, \tag{10}$$

where g is the cosine of the unknown angle and $\mathbf{X}$ is the pair $(\mathbf{C}, g)$. The parameter $q > 0$ determines the influence of the modifying term and has to be chosen prior to minimization. If the data normals are close to the surface normals of a cone one gets good results by choosing a higher value of q (e.g. $q = 2$), otherwise one better takes a smaller q. In general $q = 0.5$ will cover most cases. Analogously to the previous section we have to solve a system of equations with two Lagrangian multipliers λ, μ:

$$(N_q - \lambda D' - \mu K') \cdot \mathbf{X} = \mathbf{0},\ \mathbf{X}^T \cdot D' \cdot \mathbf{X} = 1,\ \mathbf{X}^T \cdot K' \cdot \mathbf{X} = 0. \tag{11}$$

Here $D' = \mathrm{diag}(1, 1, 1, 0, 0, 0, 0)$ and

$$K' = \left(\begin{array}{cccccc|c} & & & & \frac{1}{2} & & \\ & & & & & \frac{1}{2} & \\ & & & & & & \frac{1}{2} \\ \frac{1}{2} & & & & & & \\ & \frac{1}{2} & & & & & \\ & & \frac{1}{2} & & & & \\ \hline & & & & & & 0 \end{array}\right).$$

To compute the solution we can use the methods of the previous section.

We can easily calculate a generator line of the approximating cone. First we choose a plane through the axis, rotate the data points into it and fit a line to the rotated points. This line will be a generator l. Further we want to measure how well the calculated cone is fitting the given data. This is done by calculating the sum of the squares of distances of the rotated points to l. The distance of a rotated point to l equals the distance of the original point to the cone and shows how well it is approximated by the cone. Note that this error measure does not involve the surface normals.

4 Handling outliers

So far we have dealt with the approximation by a single cone of revolution. Our intention now is to divide the given data into regions in each of which a good fit by a single cone of revolution can be found. Imagine a region R with such a good fit. Data points from other regions will be classified as outliers with respect to R. Since a proper segmentation is unknown at the beginning we will now refine our cone fitting algorithm to recognize outliers. This will help us to distinguish different regions later.

The previous algorithm serves as the base of a weight iteration. Here, instead of F_q in (10), we minimize

$$F_{q,w} := \sum_{i=1}^{k} w_i \left((\bar{\mathbf{c}} \cdot \mathbf{n}_i + \mathbf{c} \cdot \bar{\mathbf{n}}_i)^2 + q \cdot (\mathbf{c} \cdot \mathbf{n}_i - g)^2 \right) =: \mathbf{X}^T \cdot N_{q,w} \cdot \mathbf{X}. \qquad (12)$$

Initially, all weights w_i are set to 1. Later following the robust regression method of Rousseeuw [20] we make use of the following weights:

$$w_i = \begin{cases} 1 & \text{if } |r_i/\hat{\sigma}| \leq 2.5 \\ 0 & \text{if } |r_i/\hat{\sigma}| > 2.5, \end{cases}$$

where the residual r_i is the distance of the data point d_i to the generator line and

$$\hat{\sigma} = \sqrt{\text{med } r_i^2},$$

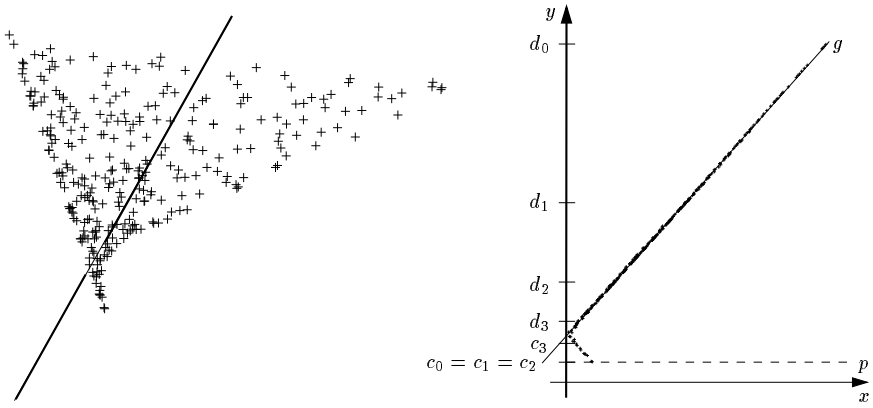
the robust estimate of the error scale; 'med r_i^2' denotes the median of the values r_i^2. With this weighting scheme the solution is recomputed. Iterating this process will finally yield a good axis, where points with vanishing weight are classified as outliers.

5 Recomputing the axis if the vertex lies in the region of interest

In some cases the vertex of the computed cone will lie inside the given data set. This situation for instance will occur if the data points come from a part of a tangent surface close to its regression curve. Since we do not wish to have any singularities of our approximating cone in the region of interest we have to recompute our solution.

5.1 Computing the position of the vertex

First we introduce a method how to recognize that the vertex lies inside the region of interest (Fig. 1(a)). Choose a half-plane through the axis and rotate the points into it. The points will lie on one side of the axis. We introduce a local cartesian



(a) Vertex lies inside the region of interest

(b) Nest of intervals

Figure 1: Computing the position of the vertex

coordinate system with the axis of the cone as y-axis. Then the x-coordinates of the points are their distance to the axis (Fig. 1(b)). To find the position of the vertex we construct a series of nested intervals (c_i, d_i). The first interval is (c_0, d_0), where c_0 equals the minimum of the y-coordinates of the rotated data points and d_0 is their maximum. Now we divide the interval into halves and decide which half possibly contains the vertex. We compute the average x-coordinate in each part and take the one with the lower value as the next interval. Iterating this procedure yields series of nested intervals.

Two cases can occur: If the vertex lies outside (c_0, d_0), the series of nested intervals will move towards the boundary. One of the bounds of the nested intervals will stay constant, i.e., either $c_i \equiv c_0$ or $d_i \equiv d_0$. In this case the computed axis does not intersect the region of interest and there is no need for further improvements.

However if the vertex lies inside (c_0, d_0), there exists an index where the intervals will leave the initial boundary (see (c_3, d_3) in Fig. 1(b)). The coordinates of the vertex can be computed approximately by the series of nested intervals (Fig. 1(b)).

Note that the series of nested intervals is finite due to the finite number of given data points.

5.2 Recomputing the axis by a least square fit

If our algorithm recognizes that the vertex lies inside the region of interest, we recompute the axis by a least square fit. Lukacs [21] and Katrycz [22] formulated and solved the problem of least square fitting by quadrics. To compute the solution both authors use Newton iteration.

Here we follow Katrycz [22]. He uses the axis, opening angle and the coordinates

of the vertex as unknown surfaces parameters. In our case we will include a side condition which ensures that the solution cone's vertex lies outside the region of interest. Consider the planes perpendicular to the computed axis which have the property that all data points are on one side of the plane and among those choose the one with minimal distance to the vertex (see plane p in Fig. 1(b)). Restricting the position of the new vertex to this plane gives us the desired side condition. We compute the solution of this least square fit with side condition according to Wolf [23]. Since this fitting algorithm works iteratively, we use our axis and vertex coordinates as starting values. For further details we refer to [22] and [23].

6 Developable surface approximation

When we assume sufficient differentiability, a developable surface is either a plane, conical surface, cylindrical surface or tangent surface of a curve or a composition of these types. Thus a developable surface is a ruled surface, where all points of the same generator line share a common tangent plane. Furthermore, along each generator line a developable surface possesses an osculating cone of revolution which may degenerate into a cylinder of revolution or a plane. A generator which has an osculating plane is called an *inflection generator*. Maekawa and Chalfant [24] introduced a method for computing inflection lines on developable surfaces. At these lines a segmentation has to be performed.

We will concentrate on modelling a developable surface by smoothly joined cones and cylinders of revolution, called *cone spline surfaces* [15]. In the previous sections we only treated the special case of approximation with cones of revolution. If the data points can be approximated well by a plane or a cylinder of revolution, our algorithm will get unstable. So let us deal with these two cases first. Points lying close to a plane will correspond to a small region in the Gaussian image. In this case one can compute a good fitting plane. Approximation with cylinders of revolution has already been treated by T. Randrup [13]. So before running our algorithm, we better check for a plane or cylinder fit.

Further a local conical or cylindrical approximant can be refined by a nonlinear least squares approach such as developed by Lukacs et al. [21].

We will now present an algorithm for approximation by a cone spline surface. Here is an outline of the algorithm.

i. *Rough segmentation of the data points.* Presuming that the given data set has no special order we first apply a systematizing procedure. Using the Gaussian images we pre-order the data points and perform a rough segmentation. This

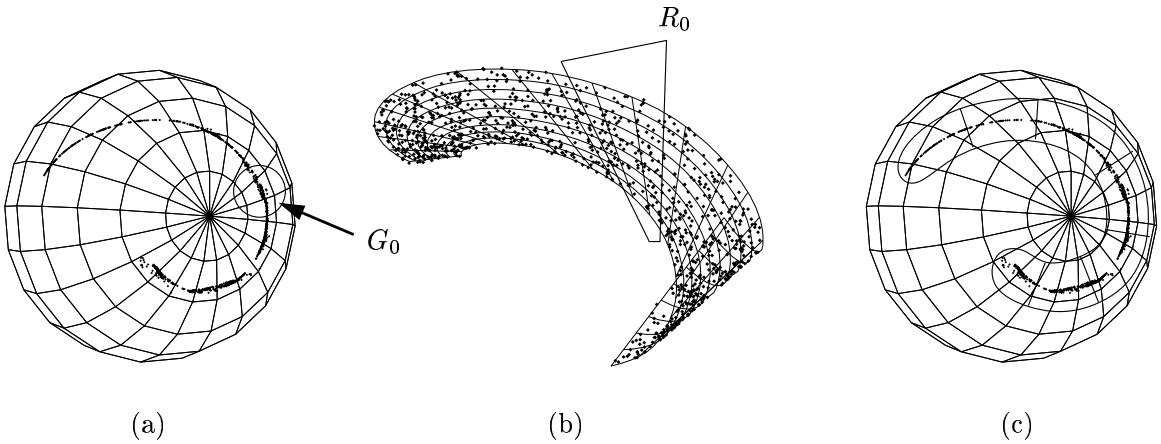


Figure 2: Gaussian image of data points

procedure will help the *region growing* algorithm used in the two next steps by roughly showing the direction where to grow.

ii. *Initial cone of revolution.* After choosing a seed point (or region) in the data set, we grow and modify a region around it as long as there exists a single cone of revolution Σ_0 which approximates the data in the region well. We choose two oriented planes through the axis A_0 of Σ_0 such that all data points within the wedge, which is defined by the intersection of the positive halfspaces with respect to the planes, are sufficiently close to Σ_0. Let us call these two planes *boundary planes*.

iii. *Marching step.* Find an appropriate region adjacent to one boundary plane and compute the next cone of revolution Σ_i. Enclose the approximated data points with two boundary planes through its axis A_i as in the initial step. Repeat this marching step on both sides of the initial wedge until the entire set of data points has been treated this way.

iv. *Smoothly joining the cones of revolution.* Using an algorithm created by Leopoldseder and Pottmann [15], we construct a smoothly joining approximating cone or cylinder between consecutive approximants. Moreover, a G^r $(r \geq 2)$ surface can be obtained by using the approximants Σ_i just for local estimates of the behaviour up to second order and then working with an algorithm for approximation with developable NURBS surfaces [10, 12].

6.1 Rough segmentation of the data points

If the input surface is nearly developable, the Gaussian image of the set of data points can be used as an auxiliary data structure for the algorithm described above. That is, we can use region segmentation/growing on the Gaussian image to help the region segmentation/growing of data points. If two data points are close enough, there is a fair chance that the Gaussian images of these two points are close enough, too. This will help constructing the seed region — the first guess of a point set by which we expect an initial cone of revolution is constructed —, and so we collect a better set of candidate points (see R_0 in Fig. 2(b)) using the region in the Gaussian image (see G_0 in Fig. 2(a)).

A curve approximating the cloud of Gaussian images of data points on the unit sphere is also useful. The normal directions of this curve can be used to compute a rough segmentation of data points (see Fig. 2(c)) as well as to grow and shrink a region. In order to determine an approximating curve for a cloud of (disordered) points, Randrup [13] used an algorithm similar to 'thinning' in image processing, and Lee [25] suggested an algorithm to give an order to the thin cloud of points for computing an approximating curve. Because the data points can be distributed very unevenly, this type of algorithms works better here than e.g. a least squares spline fitting.

6.2 Initial cone of revolution

First we choose at random a seed point $\mathbf{d}$ from the given data set, but take care that $\mathbf{d}$ is not too close to the boundary of the given data set. The seed region R_0 can be defined as a set including all points with distance to $\mathbf{d}$ less than a previously chosen constant. Furthermore, we can exploit the Gaussian image to collect a more correct seed region, as described in the previous subsection. Now we compute a fitting cone of revolution Σ_0. For every point of R_0 we can compute the error as described in section 3. Points having an error exceeding a preset tolerance are considered as outliers for Σ_0 and excluded from R_0.

R_0 is now going to be enclosed into a *wedge* W_0 as follows. We choose two oriented planes $\alpha_0, \bar{\alpha}_0$ through the axis of Σ_0, whose positive halfspaces intersect in W_0. The planes are chosen such that each data point of R_0 lies in W_0 and such that the angle between the planes is minimal. For that choose a local cylindrical coordinate system (r, ϕ, z) with axis A_0. We choose the coordinate system so that there is a data point having the coordinate $\phi = 0$. Then we compute the interval $[a, b] \subset (-\pi, \pi)$, which the polar angles ϕ of data points are lying in. We also determine the neighboring data points to R_0 and compute their polar angles ϕ. Within $[a, b]$ we take the largest

interval $[e, f]$ not covered by any polar angle to a neighbor point; the wedge then is the set of points with polar angle $\phi \in [e, f]$.

Now we incorporate all points enclosed by the wedge into R_0, if they fit well to Σ_0. With the newly grown region R_0 go back right to the start and iterate the whole process until it comes to a halt, i.e., R_0 has not changed after an iteration step. This however is the ideal case, which need not occur, so we will stop if 'not too many' points are moved in or out R_0 or if a maximum number of iteration have been done.

Having fixed R_0, Σ_0 and $\alpha_0, \bar{\alpha}_0$, we include points outside the wedge into R_0, which fit well to Σ_0 and are not too far away from the boundary planes. We designed this region growing algorithm such that points of other parts of the surface which fit by coincidence well to Σ_0, are not considered. This is achieved by restricting our search with respect to the distances of the data points to each other and by recomputing the wedge.

6.3 Marching step

Let us consider a boundary plane α_i of an approximation wedge W_i with axis A_i $(i = 0, \ldots)$, which has data points on its negative side not already contained in one of our regions. This means that one has to compute an adjacent wedge there. We take the points which have a distance less than d_1 on the positive side of α_i inside the previous wedge and points which have a distance less than d_2 on the negative side of α_i as the initial set R_i for region growing. The parameters d_1 and d_2 are previously chosen positive reals, where d_1 must be less than d_2 since the points chosen on the positive side are only for establishing a connection between consecutive cones of revolution. Now region growing and the computation of Σ_i and $\alpha_i, \bar{\alpha}_i$ are done analogously to the previous section.

Repeating this marching step on both sides of the initial wedge until the entire data point set has been treated gives us a sequence Σ_i of surfaces of revolution.

6.4 Smoothly joining the surfaces of revolution

6.4.1 Calculating a G^1 cone spline surface

By the region growing process we have obtained a sequence of cones of revolution Σ_i which is a good approximation of the set of osculating cones of a developable surface. It is now our aim to close the gap between two consecutive surfaces, say Σ_1 and Σ_2, with a cone or cylinder of revolution Σ which touches Σ_1 along a generator $\mathbf{e}_1$ and Σ_2 along $\mathbf{e}_2$. Repetition of this step for all input cones gives a G^1 cone spline surface, i.e., a surface consisting of segments of cones of revolution joined with G^1 continuity.

Leopoldseder and Pottmann [15] describe an algorithm to find appropriate generators $\mathbf{e}_1$ of Σ_1 and $\mathbf{e}_2$ of Σ_2 so that a joining cone of revolution can be fitted in between. In general there are two complex solution cones to this problem and it has been verified that those solutions will be both real and appropriate to the interpolation problem if the input cones are osculating cones to two generators of a developable surface that are sufficiently close to each other.

6.4.2 Calculating a G^r developable NURBS surface

In order to approximate a developable surface U by a developable NURBS surface V, we do the following: We write the family of tangent planes in the form

$$U(t) = (u_0(t), u_1(t), u_2(t), -1).$$

Here the statement that the plane U has coordinate vector $(u_0, u_1, u_2, -1)$ means that its equation is $z = u_0 + u_1 x + u_1 y$.

This parametrization is possible if no tangent plane is parallel to the z-axis. If this is not the case, we either have to perform a coordinate transformation or a segmentation of the surface. U is a developable NURBS surface if the functions u_0, u_1, u_2 are B-spline functions. To achieve G^r continuity, we choose B-splines of degree $r+1$. The restriction to B-splines is not essential, we could use any other finite dimensional linear space of functions. We introduce a *distance* between developable surfaces, which is induced by a scalar product in the vector space of vector-valued differentiable functions $(u_0(t), u_1(t), u_2(t))$. This could be simply the L^2 scalar product:

$$\|U\|^2 = \sum_{i=0}^{2} \int_I u_i(t)^2 dt,$$

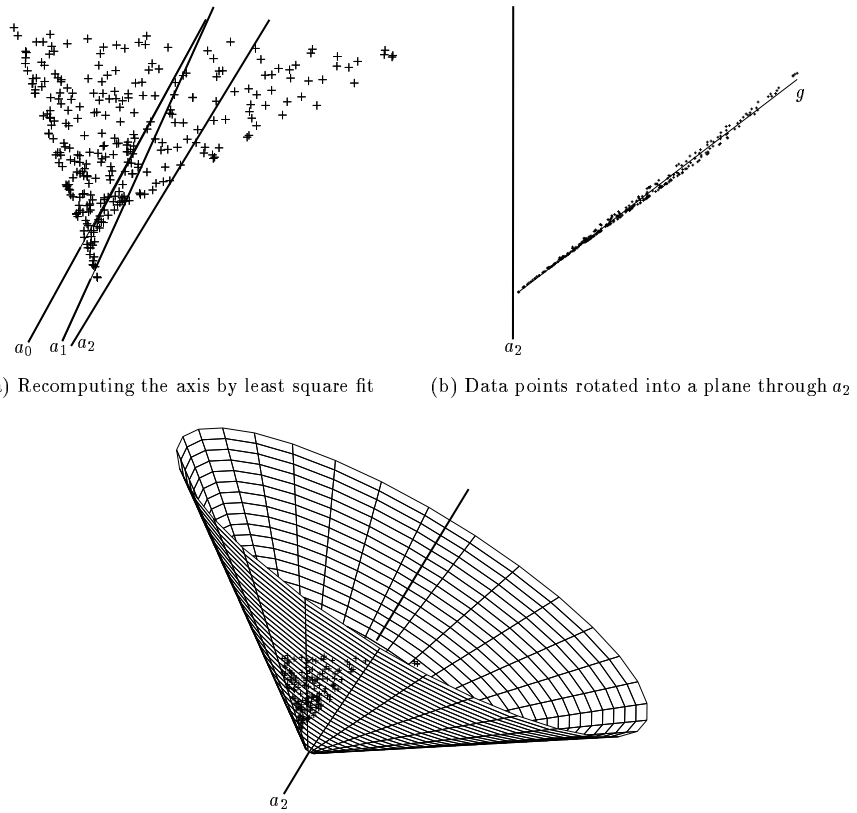
or for example

$$\|U\|^2 = \sum_{t \in T} \|U(t)\|_e^2,$$

where $T = (t_1, \ldots, t_n)$ is a list of parameter values and $\|U(t)\|_e$ is a norm defined on the set of planes. This norm can for example be derived as follows: Two planes $\alpha = (a_0, a_1, a_2, -1)$ and $\beta = (b_0, b_1, b_2, -1)$ define two function graphs over the plane. After choosing a region of interest, which will be called D, we can compute the square integral of the difference

$$d^2(\alpha, \beta) = \int_D ((a_0 - b_0) + (a_1 - b_1)x + (a_2 - b_2)y)^2 dxdy,$$

which defines the norm $\|\alpha\|_e = d(\alpha, 0)$. Now the approximation problem amounts to finding the surface V in the subspace of NURBS surfaces which is closest to the given U, where 'close' is in the sense of a scalar product. The solution is easy.



(a) Recomputing the axis by least square fit

(b) Data points rotated into a plane through a_2

(c) Approximating cone of revolution

Figure 3: Pushing the vertex out of the region of interest

By restricting ourselves to the special parametrization

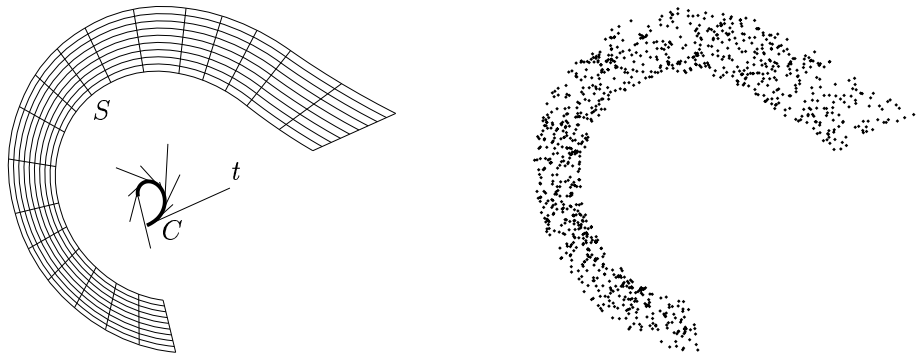
$$U(t) = (u_0(t), u_1(t), t, -1),$$

which is possible after an appropriate segmentation and coordinate transformations, we can further use all techniques developed in [12] in order to control the line of regression, which will eventually lead to the problem of minimizing a quadratic function on the union of two convex polytopes.

7 Examples

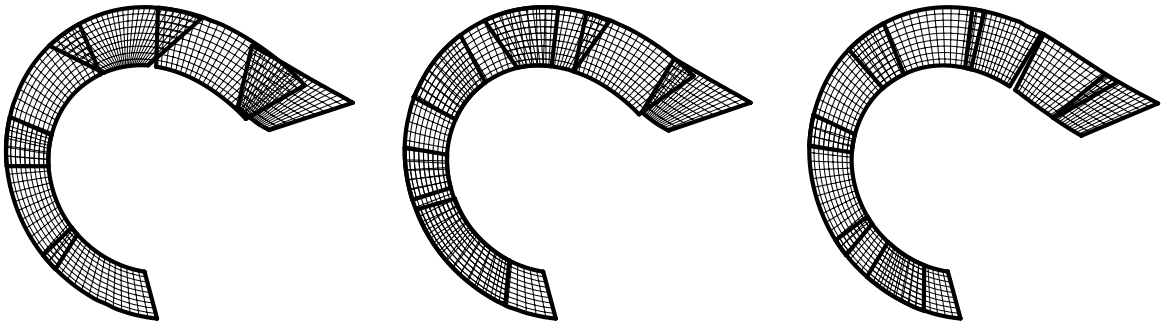
7.1 Approximation with a cone of revolution

This example illustrates how our algorithm for approximation by a cone of revolution works. As input data we take 300 points of a piece of a tangent surface close to its regression curve, Fig. 3(a). The axis a_0 is computed by the refined algorithm for approximation by a cone of revolution introduced in section 3. One can observe the approximation error in Fig. 1(b). As one can see, the error is small because the data



(a) (b)

Figure 4: Input data of example



(a) (b) (c)

Figure 5: Approximation cones for various error tolerances: (a) 6 cones with tolerance = 0.08, (b) 7 cones with tolerance = 0.05, and (c) 8 cones with tolerance = 0.03.

points lie close to the generator g. But the vertex lies inside the given data set so that the approximating cone would have a singularity (see Fig. 3(a)).

So we use a_0 as starting value for the least square fit algorithm introduced in section 5.2 and perform a Gauss-Newton iteration including the side condition as described. After two steps we get axis a_2, Fig. 3(a), where the vertex lies outside. Fig. 3(b) shows that the approximation error has increased due to our side condition. However it is still low, so that we are satisfied with the axis a_2 and compute the final cone of revolution (see Fig. 3(c)).

7.2 Approximating a surface

Fig. 4(a) shows an input surface. The tangent surface $S(u,v) = C(u) + C'(u)v$, $u \in [u_0, u_1]$, $v \in [v_0, v_1]$, is computed from a line of regression $C(u)$, $u \in [u_0, u_1]$.

To the original input surface $S(u,v)$ a smooth deviation $\epsilon(u,v)$ is added such that $\|\epsilon(u,v)\| \leq \delta$, $u \in [u_0, u_1]$, $v \in [v_0, v_1]$ for some small constant real value δ.

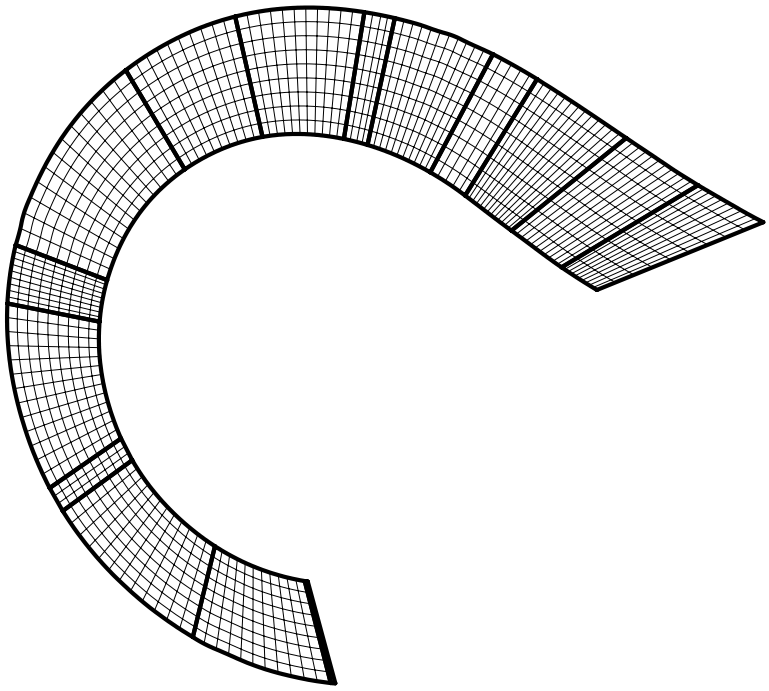


Figure 6: Cone spline surface computed from eight approximation cones

The resulting surface $T(u, v) := S(u, v) + \epsilon(u, v)$ is no longer a developable surface. Finally, we take 1000 random sample points from $T(u, v)$ (see Fig. 4(b)).

Fig. 5 illustrates various results computed from different tolerance values of approximation error. As one may expect, smaller approximation tolerance enforces the algorithm to generate a larger number of approximation cones (thus segmented regions). In Fig. 6, the final cone spline surface is shown, which is computed by the algorithm of Leopoldseder and Pottmann [15], using 8 cones shown in Fig. 5(c). The cone spline surface consists of 15 cones, i.e., 8 original approximation cones and 7 inbetween cones connecting the original cones.

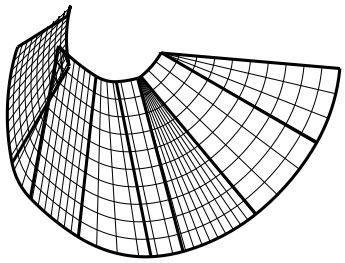


Figure 7: Surface consisting of cones

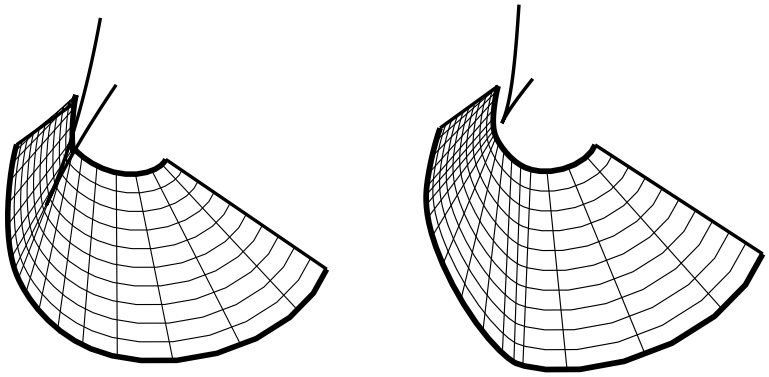


Figure 8: Smooth developable surface of differentiability class C^2

7.3 Computing a G^r NURBS surface

Fig. 7 shows the part of the surface of Fig. 6 which lies between two horizontal planes. These planes are chosen such that some vertices of cones lie in the part of the surface which is shown in the figure.

The algorithms of [12] give an approximation to this surface by a developable G^r NURBS surface (see Fig. 8, left). After pushing the line of regression out of our regions of interest the surface looks like Fig. 8, right. The deviation from the original surface can be seen clearly. But this is only to be expected if we start with a singularity and approximate with the side-condition that a singularity must not occur.

Acknowledgements

This work has been supported by grant No. P11357-MAT and No. P12252-MAT of the Austrian Science Foundation, by grant No. P00401 of the Systems Engineering Research Institute, Republic of Korea and No. EF586/CAGD of the Danish Academy of Technical Sciences.

References

[1] J. Boersma and J. Molenaar, Geometry of the shoulder of a packaging machine, *SIAM Review.* **37 (3)**, 1995, 406–422.

[2] W. H. Frey and M. J. Mancewicz, Developable Surfaces: Properties, Representations and Methods of Design, Technical Report, *GM Research Publication GMR-7637*, 1992.

[3] E. Kruppa, *Analytische und konstruktive Differentialgeometrie*, Springer Verlag, Wien, 1957.

[4] T. Randrup, Design and engineering of double-curved ship surfaces, *ECMI Newsletter.* **20**, 1996, 18–20.

[5] T. Lamb, Shell Development Computer Aided Lofting — Is There a Problem or Not?, *Journal of Ship Production.* **11 (1)**, 1995, 34–46.

[6] T. Randrup and N. Basu, Design of shell plates minimizing the heat input, in *Proceedings of the 27th Israel Conference on Mechanical Engineering, Technion, Israel, 1998.*

[7] H. Pottmann and G. E. Farin, Developable rational Bézier B-spline surfaces, *Computer Aided Geometric Design.* **12**, 1995, 513–531.

[8] J. Hoschek and D. Lasser, *Grundlagen der geometrischen Datenverarbeitung*, Teubner, Stuttgart, 1989.

[9] J. Hoschek and H. Pottmann, Interpolation and approximation with developable B-spline surfaces, in *Mathematical Methods for Curves and Surfaces* (M. Dæhlen, T. Lyche and L. L. Schumaker, Eds.), pp. 255–264, Vanderbilt University Press, Nashville, 1995.

[10] J. Hoschek and M. Schneider, Interpolation and approximation with developable surfaces, in: *Mathematical Methods for Curves and Surfaces* (M. Dæhlen, T. Lyche and L. L. Schumaker, Eds.), pp. 185–202, Vanderbilt University Press, Nashville, 1997.

[11] J. Hoschek and U. Schwanecke, Interpolation and approximation with ruled surfaces, in: *The Mathematics of Surfaces VIII* (R. Cripps, Ed.), pp. 213–231, Information Geometers, Birmingham, 1998.

[12] H. Pottmann and J. Wallner, *Approximation Algorithms for Developable Surfaces* to appear in: *Computer Aided Geometric Design*,1999.

[13] T. Randrup, Approximation of surfaces by cylinders, *Computer Aided Design*, **30**, 1998, 807–812

[14] P. Redont, Representation and deformation of developable surfaces, *Computer-Aided Design.* **21 (1)**, 1989, 13–20.

[15] S. Leopoldseder and H. Pottmann, Approximation of developable surfaces with cone spline surfaces, *Computer Aided Design.* **30**, 1998, 571–582

[16] T. Varady, R. R. Martin and J. Cox, Reverse engineering of geometric models – an introduction, *Computer Aided Design.* **29**, 1997, 255–268.

[17] H. Pottmann and T. Randrup, Rotational and helical surface approximation for reverse engineering, *Computing.* **60**, 1998, 307–323.

[18] H. Pottmann, H.-Y. Chen and I.-K. Lee, Approximation by Profile Surfaces, in *The Mathematics of Surfaces VIII* (A. Ball et al., Eds.), pp. 17–36, Information Geometers, 1998.

[19] H. Pottmann, I.-K. Lee and T. Randrup, Reconstruction of kinematic surfaces from scattered data, Technical Report No. 48, Institut für Geometrie, February 1998. also in: *Proceedings, Symposium on Geodesy for Geotechnical and Structural Engineering, Eisenstadt, Austria, 1998*, pp. 483–488.

[20] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, Wiley, New York, 1987.

[21] G. Lukács, A. D. Marshall and R. R. Martin, Geometric least squares fitting of spheres, cylinders, cones and tori. Preprint, 1997.

[22] W. Katrycz, Least Squares Treatment of Conical Surfaces, *Symposium on Geodesy for Geotechnical and Structural Engineering, Eisenstadt/Austria*, 1998, pp. 518–523.

[23] H. Wolf, *Ausgleichsrechnung II*, Dümmler Verlag, Bonn, 1979, pp. 163.

[24] J. Chalfant and T. Maekawa, Computation of inflection lines and geodesics on developable surfaces, preprint, MIT, 1997.

[25] I.-K. Lee, Curve approximation for unorganized data points, Technical Report No. 55, Institut für Geometrie, February 1998.