Methods to recover constant radius rolling ball blends in reverse engineering

Géza Kós
Dept. of Computer Science
Cardiff University[*]

Ralph R. Martin
Dept. of Computer Science
Cardiff University

Tamás Várady
Computer and Automation Research Institute, Budapest

May 3, 2002

1 Introduction

Reverse engineering of geometric shape is the process of converting large amounts of measured data points into concise and consistent computer representations of geometry. Applications include the reproduction of engineering parts with no available documentation or CAD definition for such purposes as redesign, analysis and visualisation. Other applications include the creation of various mating surfaces for parts of the human body, for example. As described in a recent survey [18], reverse engineering typically consists of four phases: data acquisition, preprocessing, segmentation and geometric model creation. Each of these four phases is quite complicated and they are strongly interrelated. Depending on the basic assumptions made and the quality and quantity of the measured data, there is a great variety of applied algorithms and computer models, as reflected in many recent publications which investigate various aspects of reverse engineering of shape.

The topic of this paper, recovering constant radius rolling ball blends, is mainly of interest when reconstructing mechanical engineering parts. From a geometric point of view, we assume that these objects are bounded by trimmed *primary surfaces*, which determine the basic shape of the object. These are relatively large in comparison to smaller *blending surfaces* which provide smooth transitions between the primary surfaces. Blends strongly depend on the primary surfaces, and are often used for reasons of aesthetics, manufacturability, stress reduction etc. As discussed in [21], there are many methods to create blends and represent them in various mathematical forms. Here we restrict our interest to the most widely used class of *constant-radius rolling ball* blends due to their simplicity and intuitive behaviour. Such blends are nominally generated by sweeping a rolling ball moving in contact with two adjacent primary surfaces. We assume that *edge blends* are small enough that they do not interfere with each other, except where they run together at the vertices; here *vertex blend faces* may need to be inserted to complete the object.

The purpose of the current investigation is to present and compare algorithms for recovering constant radius rolling ball blends. This is a particular subproblem of the final model building phase of reverse engineering. We do not go into details concerning the various steps of preprocessing—algorithms to perform filtering, triangulation and decimation are described elsewhere [10, 17], but we assume that

[*]Visiting from Computer and Automation Research Institute, Budapest

the blend data points are in some triangulated format. We also neglect the real practical difficulties arising in segmenting data points in the current discussion (see [3, 9, 13, 20]. Here we simply assume that the data points belonging to the primary surfaces have already been separated from the rest, and accurate surface representations for the primary surfaces have already been created by applying various surface fitting methods. After this phase, we assume that any remaining data points not belonging to one of the primary surfaces (within tolerance) are points belonging to a blend surface lying between two adjacent primary surfaces. In the case of constant radius rolling ball blends, determining the radius of the blend enables the computation of the whole blending surface once the primary surfaces are known. (We do not consider in this paper the more specialised behaviour of blends near vertices—see [19].)

Of course, building a complete and consistent topological structure for a whole solid model is a difficult task. Related work is discussed in [2, 6, 8]. We do not consider such issues in this paper.

As a basic principle, we believe that the recovery of blends should be a process without user interaction. Edge blends should be automatically derived based on the primary surfaces, and similarly, vertex blends should be automatically derived based on the edge blends and the primary surfaces. Algorithms to extend the current methods to variable radius rolling ball blends and vertex blends are the subject of further research.

The outline of the rest of this paper is as follows. In the first section, algorithms for various basic tools used by one or more of the main algorithms are presented briefly. These cover known results with several useful improvements. The next section is devoted to three general strategies for generating blends based on curvature estimation, spine generation and local fitting of maximal spheres. Although these are applicable for primary surfaces with either algebraic or parametric surface representations, our main aim is to exhaustively handle cases for certain regular (simple) surfaces throughout the whole paper. In particular, classes of special blends are identified and analysed, in order to gain efficiency, accuracy and robustness wherever possible. The last section evaluates the algorithms developed. Several test examples and tables provide a systematic comparison of the various methods in terms of speed, accuracy and robustness. A summary of the preferred methods and open research topics for the future concludes the paper.

2 Tools

In this section we summarise several tools used in our algorithms. Some of these methods are well-known, while others are new. Here only brief details are given.

2.1 Plane and line fitting

For plane and line fitting, the well-known least squares method is used. This means that we minimise the sum $\sum_i (\mathbf{n} \cdot \mathbf{x}_i - d)^2$, under the normalisation $|\mathbf{n}| = 1$, where $\mathbf{n}$ is the normal to the plane (or line), d is its distance from the origin, and the $\mathbf{x}_i$ are the data points. This problem leads to a simple eigenvalue problem, and the result is independent of the co-ordinate system.

In several cases we want a plane which passes through the origin; this means that d must be 0.

2.2 Normal estimation

To estimate the surface normal at some point $\mathbf{x}_0$, we take the closest n neighbouring points, denoted by $\mathbf{x}_1, \dots, \mathbf{x}_n$ ($n = 20$ in our experiments). To do this, we use the neighbourhood information from the triangulation. Then we build a local co-ordinate system with the origin at $\mathbf{x}_0$, and find the implicit quadric of the form

$$Q(\mathbf{x}) = \mathbf{x}^t A\mathbf{x} + \mathbf{n} \cdot \mathbf{x}$$

satisfying the condition $|\mathbf{n}| = 1$, for which

$$\sum_i Q^2(\mathbf{x}_i)$$

is minimal. Then the normal of the surface $Q(\mathbf{x}) = 0$ at the origin is simply $\mathbf{n}$. After some elimination, this leads to a very simple eigenvalue-problem.

The more usual method used in the literature is to fit a least squares plane (equivalent to setting $A = 0$) and optimise only $\mathbf{n}$. If the curvature is high, the error in this method can be very large at points near the surface boundary. In the case of blends this would affect too many points, as many are near the edge of the blend.

We want all estimated normals to point to the same side of the surface, so we must reverse the sign of the normal at some points.. To distinguish the two sides, the average of the triangle normals around each point is used. After this operation, the side to which the normals point will be referred to as the positive side, and the opposite side is the negative side.

2.3 Principal curvature and direction estimation

Some of our methods need the principal curvature values and the principal directions of the blend. It is possible to compute the principal curvature values from the quadric used for normal estimation, but we found that the result is very sensitive to noise. Thus, the well-known parabolic method is used which was found more stable.

First a new local co-ordinate-system is built using the normal as a local z-axis. Then we find the real numbers a, b and c for which

$$\sum_i (ax_i^2 + 2bx_iy_i + cy_i^2 - z_i)^2$$

is minimal, then compute the principal curvature values and directions from the paraboloid $z = ax^2 + 2bxy + cy^2$.

There is also a correction added to the curvature values, to reduce the effect that the paraboloid has maximum curvature at its vertex. Take a surface with principal radii R_1 and $\pm R_2$, given by

$$z = \left(R_1 - \sqrt{R_1^2 - x^2}\right) \pm \left(R_2 - \sqrt{R_2^2 - y^2}\right).$$

Let us assume that we have several points on the surface and the pairs (x, y) are uniformly distributed in the circle $x^2 + y^2 < r^2$. An easy but relatively long computation shows that the parabolic method gives a correct estimate of the principal directions at the origin. Furthermore, instead of the correct curvature values $\frac{1}{R_1}$ and $\frac{1}{R_2}$,

$$k_1 = \frac{1}{R_1} + \frac{21}{128} \cdot \frac{r^2}{R_1^3} \mp \frac{3}{128} \cdot \frac{r^2}{R_2^3} + O\left(\frac{r^4}{R_1^5}\right) + O\left(\frac{r^4}{R_2^5}\right)$$

and

$$k_2 = \pm\frac{1}{R_2} \pm \frac{21}{128}\cdot\frac{r^2}{R_2^3} - \frac{3}{128}\cdot\frac{r^2}{R_1^3} + O\left(\frac{r^4}{R_1^5}\right) + O\left(\frac{r^4}{R_2^5}\right).$$

So we take

$$r = \sqrt{\frac{2}{N}\sum_{i=1}^{N}(x_i^2 + y_i^2)}, \quad N = 20$$

and use

$$\tilde{k}_1 = k_1 - \frac{21}{128}r^2k_1^3 + \frac{3}{128}r^2k_2^3 \quad \text{and} \quad \tilde{k}_2 = k_2 - \frac{21}{128}r^2k_2^3 + \frac{3}{128}r^2k_1^3$$

instead of k_1 and k_2.

This correction doubles the number of correct digits in the curvature estimates.

2.4 Circle and sphere fitting

For fitting a sphere of radius R and centre $\mathbf{c}$, we use the quasi-least-squares method of Pratt [16], which works by minimising the sum

$$S(\mathbf{c}, R) = \sum_i \left(\frac{(\mathbf{x}_i - \mathbf{c})^2 - R^2}{2R}\right)^2. \tag{1}$$

For points lying on the sphere, the gradient of the function $\frac{1}{2R}\big((\mathbf{x}-\mathbf{c})^2 - R^2\big)$ is of unit length. Thus, if the points are close to the sphere or circle, the value of S is approximately equal to the sum of the squares of the Euclidean distances.

The advantage of this method is that S is a rational function of the parameters $\mathbf{c}$ and R, and (1) can be easily computed. To optimise the parameters an equation of degree 5 needs to be solved. (In the corresponding case of circle fitting, the equation is of degree 4.)

In degenerate cases, the method will fit planes or lines respectively, giving $\frac{1}{R} = 0$ as result. (For this reason, it is better to use $\frac{1}{R}$ as a parameter instead of R.)

Note that minimising

$$\sum_i \big((\mathbf{x}_i - \mathbf{c})^2 - R^2\big)^2$$

does not work in practice. It makes the computation straightforward, but, if the points are taken from a relatively small part of the sphere, noise can make the estimated radius much smaller than desired.

2.5 Cylinder fitting

The idea of Pratt can be used to fit cylinders, as noted by Lukács, Marshall and Martin [14]. We take three unit vectors, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, which are perpendicular to each other. $\mathbf{w}$ is the direction of the axis (see Fig. 1), $a\mathbf{u} + b\mathbf{v}$ is a vector to a point on the axis, and R is the radius of the cylinder. We want to minimise the sum

$$S(\mathbf{u}, \mathbf{v}, \mathbf{w}, a, b, R) = \sum_i \left(\frac{(\mathbf{u}\cdot\mathbf{x}_i - a)^2 + (\mathbf{v}\cdot\mathbf{x}_i - b)^2 - R^2}{2R}\right)^2.$$

This is a quasi-least-squares fit again, and S is a rational function of the parameters. Of course, the optimum does not change if the directions $\mathbf{u}$ and $\mathbf{v}$ are rotated around $\mathbf{w}$.

After computing the coefficients in S, we use an iterative method to optimise the parameters. For a given $\mathbf{w}$, the vectors $\mathbf{u}$ and $\mathbf{v}$ can be taken arbitrarily (satisfying

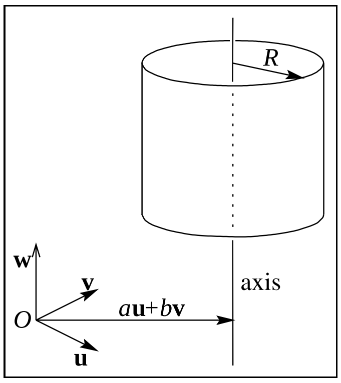

Figure 1: Parametrisation of the cylinder

perpendicularity), and the best a, b and R can be computed explicitly. We compute them and the value of S, together with their first and second derivatives with respect to $\mathbf{w}$. Knowing the derivatives of the error, optimising is simple.

For the initial value of $\mathbf{w}$, we take the normals at 25 random points, and fit a plane containing the origin to them.

The speed of cylinder fitting is high and this method works well in practice.

Just as the circle fitting algorithm can fit lines, this algorithm can fit planes in degenerate cases.

2.6 Cone fitting

For cones and tori, similar quasi-least-squares methods need to be performed over all points for each iteration. This makes cone and torus fitting much slower.

For cone fitting, we follow a similar approach to cylinder fitting. The axis of the cone is parametrised with unit vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ and real numbers a and b. The direction of the axis is $\mathbf{w}$, and it passes through the point $a\mathbf{u} + b\mathbf{v}$. If $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, a and b are given, the task is easy. The points are rotated about the cone axis into a plane, then a least squares line is fitted to them. The error of this line fitting and the error of the cone fitting are identical.

Again, we compute the error and its first and second derivatives respect to changes in $\mathbf{w}$, a and b (4 real parameters). Then the derivatives are used to determine the change in the parameters at each iteration.

The initial axis is computed in the following way. First we take 100 points and the normals at these points. Fit a plane to the normals, which gives the axis direction. Then project the points onto a plane which is perpendicular to the axis. The direction of projecting a point is perpendicular to the normal, and is in the same plane as the axis direction and the normal at the point. Last, a circle is fitted to the projected points. The centre of this circle lies on the axis.

In degenerate cases, this method can fit a cylinder or a plane, when the line fitted to the rotated points is parallel or perpendicular to the axis respectively.

2.7 Torus fitting

This is done using a similar iterative method to cone fitting. We take the same parametrisation of the axis. After rotating the points into a plane, we fit a circle to them and find the error of circle fitting, together with its first and second derivatives.

To compute the initial axis, we use a modified version of the method due to Pottmann and Randrup [15] (see below). In most cases there are two candidate initial axes, so we run the iteration for both of them. At the end, the axis producing the better fit is chosen.

Torus fitting can fit self-intersecting tori, or it can degenerate to finding cones, cylinders or planes.

2.7.1 Modified Pottmann's method

Pottmann and Randrup's method determines the axis of a rotationally symmetric surface. A set of sample points and the normal vectors at each point are required. A line through each point is taken in the normal direction, and the axis line needs to be computed which intersects all of these normal lines.

Let the points be $\mathbf{x}_1, \dots, \mathbf{x}_N$ and the normals $\mathbf{n}_1, \dots, \mathbf{n}_N$. Denote the axis direction by $\mathbf{d}$, and let $\mathbf{a}$ be an arbitrary point of the axis and $\mathbf{a}_0 = \mathbf{a} \times \mathbf{d}$. Also, denote the angle between $\mathbf{d}$ and $\mathbf{n}_i$ by φ_i, and the distance between the normal line $\mathbf{x}_i + t\mathbf{n}_i$ and the axis by δ_i.

The natural method would be to minimise $\sum \delta_i^2$, which is difficult to handle. Instead, Pottmann and Randrup suggest minimising

$$S(\mathbf{d}, \mathbf{a}) = \sum (\delta_i \sin \varphi_i)^2,$$

because

$$\delta_i \sin \varphi_i = \big|(\mathbf{d} \times \mathbf{n}_i) \cdot (\mathbf{a} - \mathbf{x}_i)\big|$$

and

$$(\mathbf{d} \times \mathbf{n}_i) \cdot (\mathbf{a} - \mathbf{x}_i) = (\mathbf{a} \times \mathbf{d}) \cdot \mathbf{n}_i + (\mathbf{x}_i \times \mathbf{n}_i) \cdot \mathbf{d} = \mathbf{n}_i \cdot \mathbf{a}_0 + (\mathbf{x}_i \times \mathbf{n}_i) \cdot \mathbf{d}$$

is a linear function of $\mathbf{a}_0$ and $\mathbf{d}$. Due to this linearity, S can be written as a simple quadratic form on $\mathbf{a}_0$ and $\mathbf{d}$. We need to find its minimum under the conditions $|\mathbf{d}| = 1$ and $\mathbf{a}_0 \cdot \mathbf{d} = 0$.

At this point, Pottmann discards the condition $\mathbf{a}_0 \cdot \mathbf{d} = 0$, changing the problem to a simple generalised eigenvalue problem. The price of this step is that the algorithm gives 0 as error for the axes of helical surfaces.

We use Pottman's method to give an initial estimate for another iterative method. For a given $\mathbf{d}$, the best $\mathbf{a}_0$ and the value of S can be computed explicitly. We compute them together with their first and second derivatives (with respect to $\mathbf{d}$), and compute the change of $\mathbf{d}$ for each iteration from the derivatives.

In some cases, especially in the case of a torus, the function S may have two local minima, giving two candidate axes (see Figure 2.)

If the normals are accurate, Pottman's method gives 0 as the error for the correct axis, and a positive value for the wrong one. Due to the noise, the normals are inaccurate and this increases the error of both axes. Unfortunately, the increase in error is greater for the correct axis if its distance from the points is greater. This phenomenon may lead to the choice of a wrong axis as described in [1, 2].

We take the two better eigenvectors computed by the original method of Pottmann, and compute both local best axes.

3 Estimating the blend radius in the general case

In this section we describe several algorithms that can be used to estimate the blend when the primary surfaces are arbitrary (parametric or implicit) surfaces. The first method does not require these surfaces to be known. The other two methods require some explicit representation of these surfaces, even though they may be of any type.

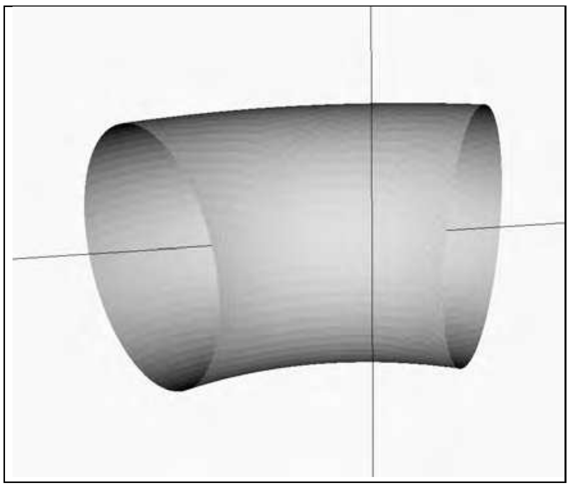
Figure 2: The two axes produced by Pottman's method

3.1 The average principal curvature method

This method works by estimating the principal curvature values at each blend point. We do not use any information about the two primary surfaces and we try to determine the radius of the blend as a piece of a general canal surface.

Before describing the concrete algorithm, we summarise the method and the related problems to be solved.

3.1.1 Relation between the blend radius and the principal curvature values

Theoretically, one of the principal curvature directions is always perpendicular to the spine curve (the locus of the centre of the rolling ball which generates the blend) and the other is parallel to the tangent to the spine. If the radius of the blend is R, the principal curvature value in the perpendicular direction is always $\frac{1}{R}$. The other principal curvature value may vary (see Figure 3).

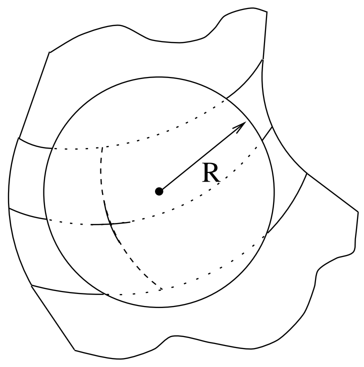

Figure 3: Two principal directions on the blend

3.1.2 Choosing the correct side

A problem is to decide on which side of the blend the rolling ball lies. We can distinguish two cases:

- The ball rolls on the positive side of the surface (the side to which the normals point). We call this the *positive* case.
- The ball rolls on the negative side of the surface (*negative* case).

Let us assume that we have the normal direction at some point $\mathbf{p}_i$ and signed principal curvature values $k_{1,i} \leq k_{2,i}$. In the positive case, the signed curvature value in the perpendicular direction is $+\frac{1}{R}$, and it can not be greater than $\frac{1}{R}$ in the parallel direction unless the blend is self-intersecting. We thus have $k_{2,i} = +\frac{1}{R}$ and $k_{1,i} \in (-\infty, \frac{1}{R}]$.

In the negative case the signs and the order of $k_{1,i}$ and $k_{2,i}$ are reversed, $k_{1,i} = -\frac{1}{R}$ and $k_{2,i} \in [-\frac{1}{R}, \infty)$.

If the blend is convex, $k_{1,i}$ and $k_{2,i}$ have the same sign, and we can choose $\max(|k_{1,i}|, |k_{2,i}|)$. But for concave blends, there is no information about the relation between $|k_{1,i}|$ and $|k_{2,i}|$ (see Figure 4).

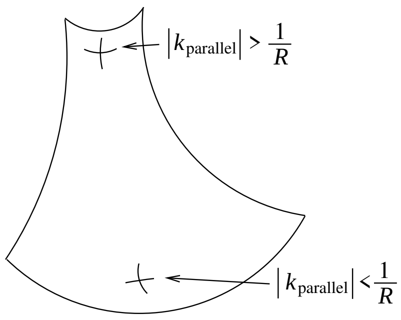
$|k_{\text{parallel}}| > \frac{1}{R}$
$|k_{\text{parallel}}| < \frac{1}{R}$

Figure 4: Relation between $|k_{\text{parallel}}|$ and $\frac{1}{R}$

In most cases, the curvature value in the parallel direction is not constant, and thus we can decide which curvature is which by comparing the variation of the $k_{1,i}$ and $k_{2,i}$ values. Unfortunately, there are exceptions. On blends involving cylinders, curvature is constant and such variation does not exist. For cylinder blends, we have to decide by comparing the magnitudes of $k_{1,i}$ and $k_{2,i}$.

To give a unique rule for the decision, we suggest that the ratio of the average and the deviation of the $k_{1,i}$ and $k_{2,i}$ values respectively should be compared.

3.1.3 The problem of bad points

If a relatively high level of noise is present, curvature estimation is very unstable. The error in the curvature estimates can be larger than the curvature values themselves, and the estimated principal curvature values and directions can be wrong too.

To reduce the effect of such bad estimates, we classify each point as good or bad. For this classification, we compare the estimated principal curvature directions at neighbouring points. A point is called good if its principal curvature directions are similar to those of its neighbours (see Figure 5). Curvature estimates at bad points are discarded.

3.1.4 Averaging vs. choosing the median

After determining whether the positive or the negative case applies, and bad points have been thrown away, we have a set of $k_{2,i}$ or $k_{1,i}$ values as estimates of $\frac{1}{R}$ respectively, and we have to compute a single value as the result.

We have tried three different methods for doing this:

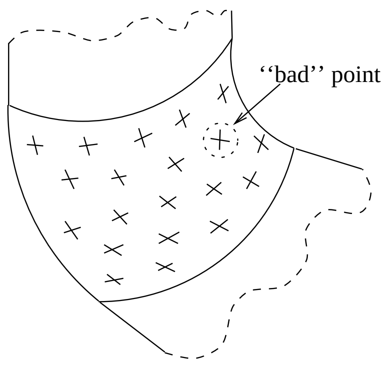

Figure 5: Good and bad points

- Averaging all values
- Throwing away the upper and lower 25% and averaging only the middle 50%
- Taking the median value.

Experiments show that the best results are obtained by averaging the middle half. The reason why it is better than averaging all values is that at some points, even though the principal curvature directions may be good, the curvature values are wrong.

3.1.5 Detailed algorithm

1. Estimate the signed principal curvature values $k_{1,i} \leq k_{2,i}$ and principal directions $\mathbf{d}_{1,i}$, $\mathbf{d}_{2,i}$ at each point.
2. Classify points as good and bad, and throw away bad points. To classify the ith point, we take the neighbours of $\mathbf{p}_i$ from the triangulation (denote their indices by $j_1, \ldots, j_n$), and compute the average of the angles between the direction $\mathbf{d}_{1,i}$ and the directions $\mathbf{d}_{1,j_1}, \ldots, \mathbf{d}_{1,j_n}$, and between the direction $\mathbf{d}_{2,i}$ and the directions $\mathbf{d}_{2,j_1}, \ldots, \mathbf{d}_{2,j_n}$, respectively. We call the point $\mathbf{p}_i$ good if both average angles are less than $15°$. (Note that both computations are needed as $\mathbf{d}_{1,i}$, $\mathbf{d}_{2,i}$ and $\mathbf{d}_{1,j}$, $\mathbf{d}_{2,j}$ span different planes).
3. Sort the sets of the resulting $k_{1,i}$ and $k_{2,i}$ values. Throw away the upper and lower 25%, then compute the average and the deviation of the middle 50% (denote them by $M(k_1)$, $M(k_2)$ and $D(k_1)$, $D(k_2)$ respectively).
4. Determine which case we have. If both $M(k_1)$ and $M(k_2)$ are positive or negative, we have the positive or negative case, respectively. If $M(k_1)$ is negative and $M(k_2)$ is positive, we compare the ratios of averages and deviations; if $\frac{|M(k_1)|}{D(k_1)} > \frac{|M(k_2)|}{D(k_2)}$ or $\frac{|M(k_1)|}{D(k_1)} < \frac{|M(k_2)|}{D(k_2)}$, we assume we have the negative or positive case, respectively.
5. The estimate for R is $\frac{1}{|M(k_2)|}$ in the positive case and $\frac{1}{|M(k_1)|}$ in the negative case.

3.2 The spine reconstrucion method

As is well-known, the blend can be generated from the spine curve which is the path of the centre of the rolling ball. For an arbitrary radius R, the spine is determined by

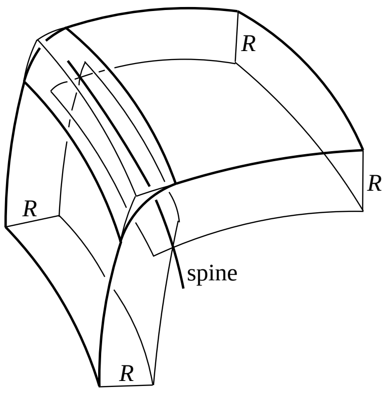

Figure 6: The spine as the intersection of offset surfaces

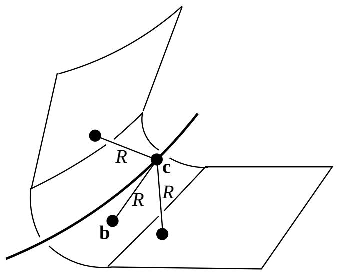

Figure 7: Spine points are a distance R from both primary surfaces and the blend surface

offsetting each primary surface by a distance R and intersecting the offset surfaces (see Figure 6).

For an arbitrary point $\mathbf{x}$ and positive R, denote by $\mathbf{c}(\mathbf{x}, R)$ the closest point of the spine for radius R to $\mathbf{x}$. If $\mathbf{x}$ is on the blend of radius R then $\mathbf{c}(\mathbf{x}, R)$ is the centre of the rolling ball in the position which generates $\mathbf{x}$. In this case the distances between $\mathbf{c}(\mathbf{x}, R)$ and either of the primary surfaces, or $\mathbf{x}$, are all R, and the directions of these three distances are perpendicular to the tangent of the spine at $\mathbf{c}(\mathbf{x}, R)$ (see Figure 7). The algorithm is based on this property of the spine. Denoting the measured points on the blend by $\mathbf{b}_1, \ldots, \mathbf{b}_N$, the method works by minimising the sum

$$S(R) = \sum_i \Big(\big|\mathbf{c}(\mathbf{b}_i, R) - \mathbf{b}_i\big| - R \Big)^2.$$

To find the best R, we use an iterative method. We first give a brief outline then describe the steps in more detail.

1. Decide on which sides of the primary surfaces the blend lies.
2. Compute an initial estimate R_0 of the radius of the blend.
3. From the estimate R_n, compute the points $\mathbf{c}(\mathbf{b}_1, R_n), \ldots, \mathbf{c}(\mathbf{b}_N, R_n)$.
4. Compute the values $S(R_n)$, and its first two derivatives $S'(R_n)$ and $S''(R_n)$.
5. If $|S'(R)|$ is not small enough, compute a new R_{n+1} value and go to step 3.

3.2.1 Deciding which sides of the primary surfaces the blend lies on

To decide on which side of a primary surface the blend lies, we compute the signed distances of $\mathbf{b}_1, \ldots, \mathbf{b}_N$ from the surface. The sign of the average distance determines on which side the blend lies.

After deciding this for both primary surfaces, for an arbitrary point $\mathbf{x}$, denote the signed distances between $\mathbf{x}$ and the primary surfaces by $d_1(\mathbf{x})$ and $d_2(\mathbf{x})$, and the unit gradient vectors of these quantities by $\mathbf{v}_1(\mathbf{x})$ and $\mathbf{v}_2(\mathbf{x})$ respectively (i.e. $\mathbf{v}_1$ and $\mathbf{v}_2$ are unit vectors, and point in the directions of the signed distances from the surfaces.)

3.2.2 The initial radius

We compute the initial radius R_0 by estimating the curvature at 25 points: at an arbitrary point $\mathbf{b}_i$ of the blend, we compute the normal $\mathbf{n}_i$ and the signed principal curvatures $k_{1,i} \leq k_{2,i}$ as in the method in Section 3.1.

We can see that the rolling ball is on that side of the blend into which the vector $\mathbf{v}_1(\mathbf{b}_i) + \mathbf{v}_2(\mathbf{b}_i)$ points. Thus, if the angle between the directions $\mathbf{n}_i$ and $\mathbf{v}_1(\mathbf{b}_i)+\mathbf{v}_2(\mathbf{b}_i)$ is less than $90°$ we use $|k_{2,i}|$ to compute R_0, and use $|k_{1,i}|$ otherwise.

The initial radius R_0 is the reciprocal of the median of the 25 principal curvature values computed.

3.2.3 Computing the projection of the blend points

In the general case we use a nested iteration to compute the projection of each blend point onto the spine. The point $\mathbf{c}_i = \mathbf{c}(\mathbf{b}_i, R_n)$ has to satisfy the following conditions:

- Both distances $d_1(\mathbf{c})$ and $d_2(\mathbf{c})$ are approximately R
- The vectors $\mathbf{c}_i - \mathbf{b}_i$, $\mathbf{v}_1(\mathbf{c}_i)$ and $\mathbf{v}_2(c_i)$ are in the same plane.

We more carefully specify the definition of the tolerances for these conditions in Section 3.2.6.

To find point $\mathbf{c}_i$, we define a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots$ which converges to $\mathbf{c}_i$. In the case of the first outer iteration, $n = 0$, we choose $\mathbf{x}_0 = \mathbf{b}_i$. In subsequent outer iterations, $n > 0$, we use the previous projection, choosing $\mathbf{x}_0 = \mathbf{c}(\mathbf{b}_i, R_{n-1})$.

To compute $\mathbf{x}_{k+1}$ from the point $\mathbf{x}_k$, we first take the distances and vectors $d_1(\mathbf{x}_k)$, $d_2(\mathbf{x}_k)$, $\mathbf{v}_1(\mathbf{x}_k)$ and $\mathbf{v}_2(\mathbf{x}_k)$.

If $d_1(\mathbf{x}_k)$ or $d_2(\mathbf{x}_k)$ is not approximately R, we choose $\mathbf{x}_{k+1}$ in the plane which passes through $\mathbf{x}_k$ and is parallel to both $\mathbf{v}_1$ and $\mathbf{v}_2$. This means that $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{v}_1(\mathbf{x}_k) + \beta\mathbf{v}_2(\mathbf{x}_k)$ for some real parameters α and β (see Figure 8). It is easy to see that if the change of $\mathbf{v}_1$ and $\mathbf{v}_2$ is negligible, the distances between $\mathbf{x}_{k+1}$ and the primary surfaces will be $d_1(\mathbf{x}_{k+1}) \approx d_1(\mathbf{x}_k) + \alpha + (\mathbf{v}_1 \cdot \mathbf{v}_2)\beta$ and $d_2(\mathbf{x}_{k+1}) \approx d_2(\mathbf{x}_k) + (\mathbf{v}_1 \cdot \mathbf{v}_2)\alpha + \beta$. Thus we choose α and β as the solutions to the linear system

$$d_1(\mathbf{x}_i) + \alpha + \big(\mathbf{v}_1(\mathbf{x}_i) \cdot \mathbf{v}_2(\mathbf{x}_i)\big)\beta = R$$

$$d_2(\mathbf{x}_i) + \big(\mathbf{v}_1(\mathbf{x}_i) \cdot \mathbf{v}_2(\mathbf{x}_i)\big)\alpha + \beta = R.$$

If $d_1(\mathbf{x}_k)$ and $d_2(\mathbf{x}_k)$ are both R, then the point $\mathbf{x}_k$ is on the spine. We check whether the vectors $\mathbf{c}_i - \mathbf{b}_i$, $\mathbf{v}_1(\mathbf{c}_i)$ and $\mathbf{v}_2(c_i)$ are in the same plane. To do this, we compute the tangent direction of the spine at $\mathbf{x}_k$ which is

$$\mathbf{t} = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|}.$$

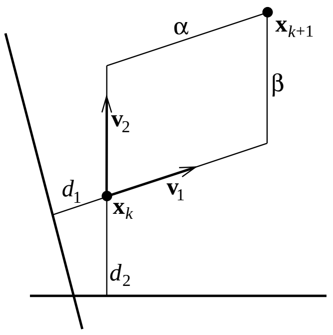

Figure 8: Moving perpendicular to the spine

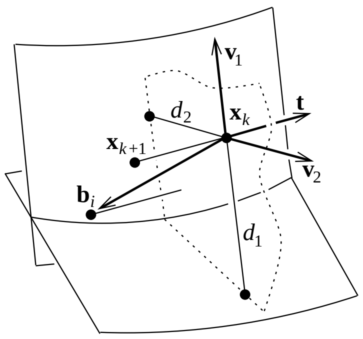

Figure 9: Moving parallel to the spine

The signed distance between the point $\mathbf{b}_i$ and the plane which intersects perpendicularly the spine at $\mathbf{x}_k$ is $\mathbf{t} \cdot (\mathbf{b}_i - \mathbf{x}_k)$. If this is sufficiently small, the iteration is finished, and we can set $\mathbf{c}(\mathbf{b}_i, R_n) = \mathbf{x}_k$.

If $\mathbf{t} \cdot (\mathbf{b}_i - \mathbf{x}_k)$ is not small enough, we choose $\mathbf{x}_{k+1}$ on the tangent line of the spine at $\mathbf{x}_k$ (see Figure 9), namely

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \big(\mathbf{t} \cdot (\mathbf{b}_i - \mathbf{x}_k)\big)\mathbf{t}$$

3.2.4 Iteratively improving the radius estimate

To modify the radius, we use the error $S(R_n)$ and its first two derivatives with respect to R_n, but we ignore changes in the vectors $\mathbf{v}_1(\mathbf{b}_i)$ and $\mathbf{v}_2(\mathbf{b}_i)$. These affect only $S''(R_n)$.

The derivative of the point $\mathbf{c}_i = \mathbf{c}(\mathbf{b}_i, R)$ with respect to R is

$$\frac{\mathrm{d}}{\mathrm{d}R}\mathbf{c}_i = \frac{\mathbf{v}_1(\mathbf{c}_i) + \mathbf{v}_2(\mathbf{c}_i)}{1 + \mathbf{v}_1(\mathbf{c}_i) \cdot \mathbf{v}_2(\mathbf{c}_i)}.$$

This implies that the first two derivatives of $|\mathbf{c}_i - \mathbf{b}_i|$ are

$$\frac{\mathrm{d}}{\mathrm{d}R}|\mathbf{c}_i - \mathbf{b}_i| = \frac{\mathbf{v}_1(\mathbf{c}_i) + \mathbf{v}_2(\mathbf{c}_i)}{1 + \mathbf{v}_1(\mathbf{c}_i) \cdot \mathbf{v}_2(\mathbf{c}_i)} \cdot \frac{\mathbf{c}_i - \mathbf{b}_i}{|\mathbf{c}_i - \mathbf{b}_i|}$$

and

$$\frac{\mathrm{d}^2}{\mathrm{d}R^2}|\mathbf{c}_i - \mathbf{b}_i| = \frac{2}{\big(1 + \mathbf{v}_1(\mathbf{c}_i) \cdot \mathbf{v}_2(\mathbf{c}_i)\big)|\mathbf{c}_i - \mathbf{b}_i|} - \frac{\big((\mathbf{v}_1(\mathbf{c}_i) + \mathbf{v}_2(\mathbf{c}_i)) \cdot (\mathbf{c}_i - \mathbf{b}_i)\big)^2}{\big(1 + \mathbf{v}_1(\mathbf{c}_i) \cdot \mathbf{v}_2(\mathbf{c}_i)\big)^2|\mathbf{c}_i - \mathbf{b}_i|^3}.$$

By summing these quantities, the first two derivatives of $S(R_n)$ can be computed easily. Then, because our goal is to eliminate $S'(R_n)$, we change R_n to

$$R_{n+1} = R_n - \frac{S'(R_n)}{S''(R_n)}$$

using a Newton-Raphson step for S'.

3.2.5 Explicit computation of the spine

In the cases where the primary surfaces are plane-plane, plane-sphere, plane-cylinder or plane-cone, the spine for a given R is always a conic curve, which can be computed explicitly from the parameters of the primary surfaces. This can make the method faster, because projection of blend points onto the spine can be computed directly.

To compare the speed of the general method and the explicit method for special cases, we conducted experiments in the plane-cylinder case.

3.2.6 Thresholds and stopping conditions

Suppose we want at least 5 correct digits in the results. Because of accumulation of errors, we insist that every step should give at least 6 correct digits. Thus in the inner iteration, we say that d_j is approximately equal to R if

$$\left|\frac{d_j}{R} - 1\right| < 10^{-6}.$$

The threshold used for the distance $\left|\mathbf{t} \cdot (\mathbf{x}_k - \mathbf{b}_i)\right|$ is 10^{-6} too.

The main iteration stops when the change between R_n and R_{n+1} is less than $10^{-6} \times R_n$, so we require

$$\left|S'(R)\right| \leq 10^{-6} \cdot R \cdot \left|S''(R)\right|.$$

3.3 The maximum ball method

This method works by computing an estimate for the blend radius at each measured point on the blend.

For a blend point $\mathbf{b}_i$, we attempt to reconstruct the position of the rolling ball which generates the point $\mathbf{b}_i$. This is the largest sphere which contains $\mathbf{b}_i$ and is tangent to both primary surfaces.

After computing the largest sphere at each blend point, we average the radii of these spheres (or take the median value) to give an estimate of the blend radius.

The maximum ball method is in some sense the dual of the iterative spine reconstruction. That method works by optimising the sum of the local errors for the same radius, this works by averaging the local best radii.

We use here the same notation as in the iterative spine reconstruction method.

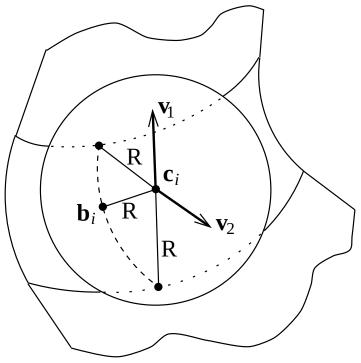

Figure 10: The maximum ball for point $\mathbf{b}_i$

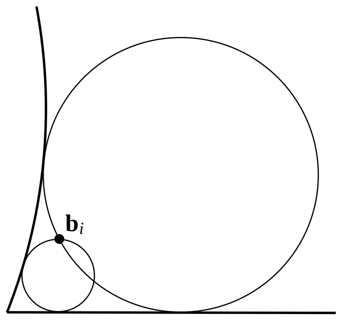

Figure 11: The maximum and minimum balls

Let us denote the centre of the current ball by $\mathbf{c}_i$. The point $\mathbf{c}_i$ has to satisfy the following conditions (see Figure 10):

- The distances $d_1(\mathbf{c}_i)$, $d_2(\mathbf{c}_i)$ and $|\mathbf{c}_i - \mathbf{b}_i|$ must all be equal to R, the radius of the ball.
- The vectors $\mathbf{v}_1(\mathbf{c}_i)$, $\mathbf{v}_2(\mathbf{c}_i)$ and $\mathbf{c}_i - \mathbf{b}_i$ must be in the same plane.

Note that there are two centre points with these properties. The second one is the centre of the smallest sphere which contains $\mathbf{b}_i$ and is tangent to the two primary surfaces (see Figure 11).

3.3.1 Computing the maximum ball in the plane-plane case

In the case where both primary surfaces are planes the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are constant, and $\mathbf{c}$ can be computed explicitly. We may write $\mathbf{c}_i$ in the form

$$\mathbf{c}_i = \mathbf{b}_i + \alpha\mathbf{v}_1 + \beta\mathbf{v}_2$$

using the second condition above, and then α, β and R must satisfy the equations

$$d_1(\mathbf{b}_i) + \alpha + (\mathbf{v}_1 \cdot \mathbf{v}_2)\beta = R,$$
$$d_2(\mathbf{b}_i) + (\mathbf{v}_1 \cdot \mathbf{v}_2)\alpha + \beta = R,$$
$$\left|\alpha\mathbf{v}_1 + \beta\mathbf{v}_2\right| = R$$

resulting from the first condition.

If the signed distances $d_1(\mathbf{b}_i)$ and $d_2(\mathbf{b}_i)$ are positive, this system leads to a quadratic equation for R with two positive roots which are the radii of the minimum and maximum balls. We need the larger root.

3.3.2 Computing the maximum ball in the general case

In the general case, we apply an iterative method to find $\mathbf{c}_i$. The iteration is very similar to the iteration described in Section 3.2.3, and we can refer to Figures 8 and 9 again. We define a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots$ which converges to $\mathbf{c}_i$.

The initial estimate is $\mathbf{x}_0 = \mathbf{b}_i$.

Given $\mathbf{x}_k$, we compute the distances $d_1 = d_1(\mathbf{x}_k)$, $d_2 = d_2(\mathbf{x}_k)$ and $d_3 = |\mathbf{x}_k - \mathbf{b}_i|$, the gradient vectors $\mathbf{v}_1(\mathbf{x}_k)$, $\mathbf{v}_2(\mathbf{x}_k)$, and the direction

$$\mathbf{t} = \frac{\mathbf{v}_1(\mathbf{x}_k) \times \mathbf{v}_2(\mathbf{x}_k)}{\left|\mathbf{v}_1(\mathbf{x}_k) \times \mathbf{v}_2(\mathbf{x}_k)\right|}$$

which is perpendicular to the plane of $\mathbf{v}_1(\mathbf{x}_k)$ and $\mathbf{v}_2(\mathbf{x}_k)$.

The condition that vectors $\mathbf{v}_1(\mathbf{c}_i)$, $\mathbf{v}_2(\mathbf{c}_i)$ and $\mathbf{c}_i - \mathbf{x}_k$ are in the same plane is equivalent to $\mathbf{t} \cdot (\mathbf{c}_i - \mathbf{x}_k) = 0$. If the value $\left|\mathbf{t} \cdot (\mathbf{c}_i - \mathbf{x}_k)\right|$ is greater than a threshold, we compute $\mathbf{x}_{k+1}$ by moving $\mathbf{x}_k$ in the direction of $\mathbf{t}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \big(\mathbf{t} \cdot (\mathbf{x}_k - \mathbf{b})\big)\mathbf{t}.$$

If $\left|\mathbf{t} \cdot (\mathbf{c}_i - \mathbf{x}_k)\right|$ is below the threshold, we compare the distances $d_1(\mathbf{x}_k)$, $d_2(\mathbf{x}_k)$ and $|\mathbf{x}_k - \mathbf{b}_i|$. If they are approximately the same, the iteration is finished and our estimate for the centre of the maximum ball is $\mathbf{c}_i = \mathbf{x}_k$.

If the distances $d_1(\mathbf{x}_k)$, $d_2(\mathbf{x}_k)$ and $|\mathbf{x}_k - \mathbf{b}_i|$ are different, we use a similar method to find $\mathbf{x}_{k+1}$ to that used in the plane-plane case. We choose $\mathbf{x}_{k+1}$ to be in that plane which contains $\mathbf{x}_k$ and is parallel to both $\mathbf{v}_1$ and $\mathbf{v}_2$. This means that $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{v}_1(\mathbf{x}_k) + \beta\mathbf{v}_2(\mathbf{x}_k)$ for some real parameters α and β. If the changes of the gradient vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are negligible, then the new distances will be

$$\begin{aligned}
d_1(\mathbf{x}_{k+1}) &= d_1(\mathbf{x}_k) + \alpha + (\mathbf{v}_1 \cdot \mathbf{v}_2)\beta \\
d_2(\mathbf{x}_{k+1}) &= d_2(\mathbf{x}_k) + (\mathbf{v}_1 \cdot \mathbf{v}_2)\alpha + \beta \\
\left|\mathbf{x}_{k+1} - \mathbf{b}_i\right| &= \left|(\mathbf{x}_k - \mathbf{b}_i) + \alpha\mathbf{v}_1 + \beta\mathbf{v}_2\right|.
\end{aligned}$$

We thus choose α and β to be the solutions of the system

$$d_1(\mathbf{x}_k) + \alpha + (\mathbf{v}_1(\mathbf{x}_k)\mathbf{v}_2(\mathbf{x}_k))\beta = d_2(\mathbf{x}_k) + (\mathbf{v}_1(\mathbf{x}_k)\mathbf{v}_2(\mathbf{x}_k))\alpha + \beta =$$
$$= \left|(\mathbf{x}_k - \mathbf{b}_i) + \alpha\mathbf{v}_1 + \beta\mathbf{v}_2\right|.$$

This system leads to a quadratic equation. Having found α, β, we can then compute $\mathbf{x}_{k+1}$.

Note that the first step of this iteration is exactly the same as the explicit computation in the plane-plane case.

3.3.3 Problems when points are close to the primary surfaces

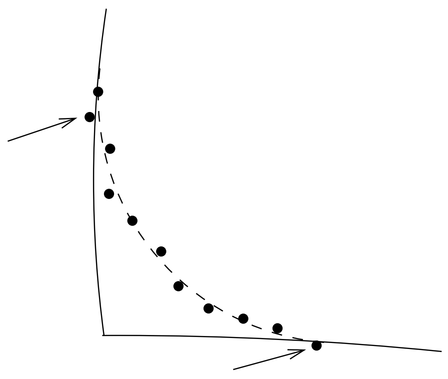

Figure 12: Points on the wrong sides of the primary surfaces

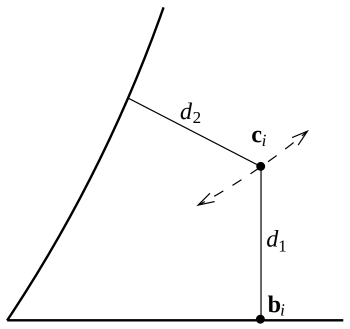

Figure 13: Instability of estimates from points which are close to the primary surfaces

The method works for any point $\mathbf{b}_i$ for which $d_1(\mathbf{b}_i) \geq 0$ and $d_2(\mathbf{b}_i) \geq 0$. However, there are two problems which may arise for blend points which are close to the primary surfaces.

The first problem is that such points, due to noise, can be on the wrong side of one of the primary surfaces (see Figure 12). For such points there is no maximum (or any) ball, and the iteration would result in an infinite loop.

The second problem is that points which are close to the primary surfaces give unstable estimates. If the point $\mathbf{b}_i$ is on one of the primary surfaces, the distances $d_1(\mathbf{c}_i)$, $d_2(\mathbf{c}_i)$ and $|\mathbf{c}_i - \mathbf{b}_i|$ are the same with respect to any $\mathbf{c}_i$ lying in the tangent direction of the bisector of the primary surfaces (see Figure 13).

To avoid such problems we compute the maximum ball only for those blend points $\mathbf{b}_i$, for which $d_1(\mathbf{b}_i)$ and $d_2(\mathbf{b}_i)$ are positive and

$$\frac{1}{10} \leq \frac{d_1(\mathbf{b}_i)}{d_2(\mathbf{b}_i)} \leq 10.$$

3.3.4 Thresholds

Again, suppose we want at least 5 correct digits in the result and that we restrict every step to give at least 6 correct digits.

The threshold for the distance $\left|\mathbf{t} \cdot (\mathbf{x}_k - \mathbf{b}_i)\right|$ is 10^{-6}.

The distances d_1, d_2 and d_3 are considered to be approximately the same if

$$\left|\frac{d_3}{d_1} - 1\right| < 10^{-6} \quad \text{and} \quad \left|\frac{d_3}{d_2} - 1\right| < 10^{-6}.$$

3.3.5 Averaging vs. taking the median

After finding the maximum ball at each measured blend point, we have a set of radii of the maximum balls, and from these we must compute a single value as the result. The maximum ball method is much more stable than curvature estimation, and the computed radii of all balls are very similar. We thus use the average of all computed radii, rather than a median or other estimates.

4 Special case algorithms for estimating the blend radius

There are specific methods for estimating blend radius that work only when the blend is a cylinder or a torus, cases which often arise in practice for engineering objects defined using simple regular primary surfaces. While these methods work only in these special cases, some of them are very simple, have much higher speed, and great accuracy. Such methods are described in this section.

The cases when the blend is a cylinder involve the following primary surface configurations:

- plane-plane; any angle
- plane-cylinder; axis of the cylinder parallel to the plane
- cylinder-cylinder; axes parallel

The cases when the blend is a torus involve:

- plane-sphere; any distance
- plane-cylinder, plane-cone, plane-torus; axis of the cylinder/cone/torus perpendicular to the plane
- sphere-cylinder, sphere-cone, sphere-torus; centre of the sphere on the axis of the cylinder/cone/torus
- cylinder-cone, cylinder-torus, cone-cone, cone-torus, torus-torus; axes of the two surfaces the same.

4.1 Direct surface fitting method

This method works by fitting the best cylinder or torus to the blend points (see [14]). We use the primary surfaces only to determine that the blend should be a cylinder or a torus, then fit the appropriate type of surface to the blend points.

We have implemented this method only for comparison with the other special methods.

4.2 Simple circle fitting method

In the special cases mentioned, the primary surfaces and the blend all belong to the same composite translationally or rotationally symmetric surface. Knowing the parameters of the primary surfaces, the direction of the translation or the axis of the rotation can be computed.

Then all blend points can be projected onto or rotated into a plane which is perpendicular to the translation or includes the axis, respectively. The images of the primary surfaces will be lines or circular arcs, and the image of the blend will be a circular arc.

The simple circle fitting method works by fitting a circle to the transformed blend points. The radius of the circle is the desired blend radius. The images of the primary surfaces are not used.

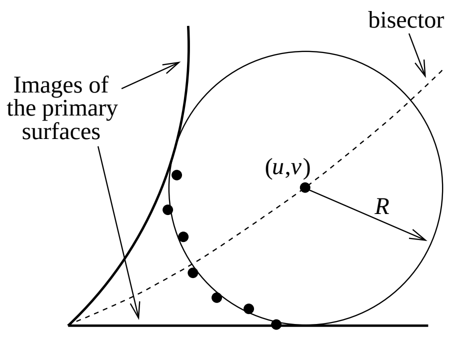

Figure 14: The constrained circle

4.3 Constrained circle fitting method

In this method we also project or rotate the blend points into a plane as above and fit a circle to them, but this time we compute the images of the primary surfaces and constrain the circle fitted to the images of the blend points to be tangent to these primary surface images (see Figure14).

We can distinguish three cases when the images of the primary surfaces are

- line-line
- line-circle
- circle-circle.

In the plane, we build a new (x, y) co-ordinate system, and parametrise the circle by (u, v, R), where (u, v) is its centre and R is its radius.

4.3.1 Direct fitting by equation solving

We have investigated two algorithms to fit a constrained circle. The first explicitly computes the parameters of the circle by solving a system of equations.

The distance of the centre (u, v) should be R from the images of the primary surfaces.

If the image of a primary surface is the line $n_1x + n_2y - d = 0$ where the unit normal vector (n_1, n_2) points to that side of the line which contains the blend, the condition can be written as

$$n_1u + n_2v - d = R.$$

If the image of a primary surface is a circle with the centre (c_1, c_2) and radius r, we have the equation

$$(u - c_1)^2 + (v - c_2)^2 = (r + R)^2 \quad \text{or} \quad (u - c_1)^2 + (v - c_2)^2 = (r - R)^2$$

depending on whether the blend is outside or inside the circle, respectively.

In the line-line case the bisector of the images of the primary surfaces is a line, and u, v can be parametrised by linear functions of R.

In the line-circle case the bisector is a parabola and u, v can be parametrised by quadratic functions of R.

In the circle-circle case the parametrisation is more difficult. The solution of the system of the two equations in the (u, v, R) parameter space is a conic curve, which can be parametrised by rational functions. Thus, we parametrise u, v and R by rational functions of a common parameter t.

Next, we compute the coefficients of the error function

$$S(u, v, R) = \sum \left(\frac{(x_i - u)^2 + (y_i - v)^2 - R^2}{2R} \right)^2. \tag{2}$$

which is another rational function of R or t respectively. Finally, we compute the minimum of S and the corresponding value of R.

We have implemented this method only in the plane-plane, plane-sphere and plane-perpendicular cylinder cases.

The degree of the equation which has to be solved to find the optimal radius is 3 in the line-line case, 5 in the line-circle case and 7 in the circle-circle case. Thus, the method is not fully explicit, and in the second two cases it needs iterative numerical solution.

4.3.2 Iterative solution

This alternative algorithm finds the best circle using iteration. First, we compute the coefficients of the error function $S(u, v, R)$ above. This is a rational function of u, v and R, and the denominator is R^2.

We now define a sequence of circles in the transformed plane by their centres $\mathbf{c}_i = (u_i, v_i)$ and radii R_i, which converge to the best circle. The iteration is similar to the iterative spine reconstruction, but this time we do not need to iterate over the points at each step.

Initially, we fit a circle to the transformed blend points, and choose the point $\mathbf{c}_0$ to be its centre.

From (u_k, v_k), we compute the distances d_1 and d_2 between this point and the images of the primary surfaces, and the gradients of the distance functions: $\mathbf{v}_1$ and $\mathbf{v}_2$. We choose $R_k = \frac{1}{2}(d_1 + d_2)$.

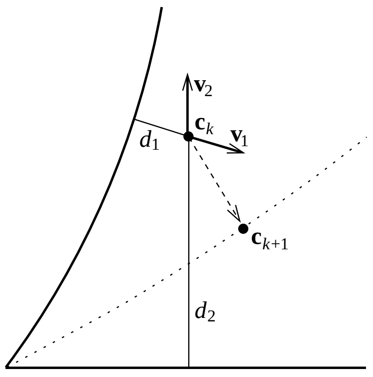

Figure 15: Projecting $\mathbf{c}_k$ onto the bisector

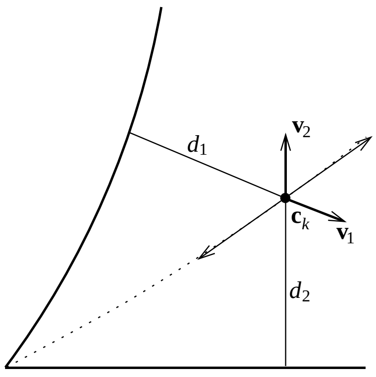

Figure 16: Moving $\mathbf{c}_k$ in the tangent direction

If the difference between distances d_1 and d_2 is greater than a threshold, we choose $\mathbf{c}_{k+1}$ by projecting $\mathbf{c}_k$ onto the bisector (see Figure 15). We ignore the change in $\mathbf{v}_1$ and $\mathbf{v}_2$, and choose the direction of the projection to be $\mathbf{v}_1 - \mathbf{v}_2$. An easy calculation shows that the best choice (when the new distances are equal) is

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \frac{d_2 - d_1}{|\mathbf{v}_1 - \mathbf{v}_2|^2}(\mathbf{v}_1 - \mathbf{v}_2).$$

However, if the distances d_1 and d_2 are approximately the same, we choose $\mathbf{c}_{k+1}$ to lie on the tangent line of the bisector at $\mathbf{c}_k$, which can be parametrised as

$$\mathbf{c}_{k+1}(t) = \mathbf{c}_k + t(\mathbf{v}_1 + \mathbf{v}_2)$$

(see Figure 16). To find the best value for the real parameter t, we assume that $R_{k+1}(t) = R_k + (1+\mathbf{v}_1.\mathbf{v}_2)t$, and compute the error $S(\mathbf{c}_{k+1}(t), R_{k+1}(t))$ and its first two derivatives with respect to t. Then, to eliminate S', we choose $t = -S'/S''$.

To obtain at least 5 correct digits, we say that d_1 and d_2 are approximately the same if

$$\left|\frac{d_1}{d_2} - 1\right| < 10^{-6}.$$

The iteration stops if d_1 and d_2 are approximately the same and the difference between R_k and R_{k+1} would be less than $10^{-6}R_k$.

We have implemented the iterative method for every case of primary surface pairs given earlier.

5 Evaluation and comparison of the methods

In order to evaluate the methods, various simulated datasets were created using the ACIS test harness. Results using this test data are presented in Sections 5.1–5.5, where we compare the methods and their advantages and disadvantages with respect to

- accuracy of results,
- speed,
- cases for which the methods reliably work.

We also briefly comment on their relative ease of implementation. Finally, an example using real measured data is presented in Section 5.6.

5.1 Simulated datasets for testing

Each dataset contained three segments: the two primary surfaces and the blend. Each surface was triangulated to provide neighbourhood information.

The blend radius was set to 10mm for the ease of comparison. Each dataset was generated at three resolutions, namely 5, 10 and 20 points per cm. In most cases, each dataset was transformed into such a position that the whole surface was visible from the z-direction. Gaussian noise was added to the z-coordinates of the points, with deviation 3% of the resolution, a value representative of that claimed by commercial scanners. We did not generate artificial outliers.

The names of the test cases below have the form $\mathbf{t_1 t_2 n.rr}$, where $\mathbf{t_1}$ and $\mathbf{t_2}$ are abbreviations for the types of the two primary surfaces (plane=**p**, sphere=**s**, cylinder=**c**, cone=**n**, torus=**t**), **n** is the number of the particular test case of that type and **rr** is the resolution. For example, dataset **pp3.05** is the third of the plane-plane examples, and the average spacing between points is $10/5 = 2$mm.

In these experiments, we fitted surfaces of known type to the *noisy* primary surface data where an explicit primary representation was required, rather than using our prior knowledge of the ideal primary surfaces.

The datasets used in our tests were as follows:

pp1 Two planes. The angle between them was 90°.

pp2 Two planes. The angle between them was 60°.

pp3 Two planes. The angle between them was 157°.

ps1 A 90° sector of the union of a plane and a sphere. The radius of the sphere was 70mm, and the distance between its centre and the plane was 40mm.

pc1 A plane and a 120° sector of a cylinder. The cylinder's axis was perpendicular to the plane, and its radius was 50mm. The blend was inside the cylinder.

pc2 A plane and a 120° sector of a cylinder. The cylinder's axis was perpendicular to the plane, and its radius was 30mm. The blend was outside the cylinder.

pc3 A plane and a cylinder. The cylinder's axis was perpendicular to the plane, and its radius was 5mm. The blend was outside the cylinder.

pc3a Same as **pc3**, but only a 120° sector.

pc4 A plane and a cylinder. The angle between the cylinder's axis and the plane normal was 30°, and its radius was 5mm. The blend was outside the cylinder.

pc4a Same as **pc4**, but only a 120° sector of the cylinder.

pc5 A plane and a cylinder. The radius of the cylinder was 50, and its axis was in the plane.

pn1 A plane and a cone. The axis of the cone was perpendicular to the plane. The radius of the cone was 46mm at the intersection with the plane, and its semi angle was 22°.

pn1a Same as **pn1**, but only a 90° sector.

pn2 A cone and a plane. The semi angle of the cone was 31°. The angle between the axis and the plane normal was 20°.

pt1 A plane and a piece of a torus. The torus' major and minor radii were 50 and 20mm, respectively. The axis of the torus was parallel to the plane, at a distance of 50mm (the same as the major radius).

ct1 A cylinder (radius=50mm) and torus (major radius=15mm, minor radius 10mm) in a general position. The angle between the axes was 75°.

nn1 Two congruent cones. Their semiangle was 34.5°. The axes were identical, and the cones intersect each other in a circle radius of 3.7mm.

nn1a Same as **nn1**, but only a 90° sector.

cc1 Two pieces of cylinders. Their radii were 70mm and 20mm, respectively. The axes were perpendicular and intersect each other. The blend was outside both cylinders.

cc2 Two cylinders. Their axes were parallel, and their radii were 70mm and 50mm. The distance between the axes was 90mm. The blend was outside the larger and inside the smaller cylinder.

ss1 Two spheres of radii 50mm and 70mm. The distance between their centres was 70mm. The dataset was a 120° segment.

ss2 The difference of two spheres. The spheres were the same as in **ss1**. The blend was inside the larger and outside the smaller sphere.

st1 A sphere (radius=160mm) and a torus (major radius=50mm, minor radius=30mm) in general position. The blend was inside the sphere and outside the torus.

tt1 Pieces of two congruent tori (major radius=60mm, minor radius=50mm). They were in such a position as to form a link of a chain. The blend was outside both.

These datasets are illustrated below.

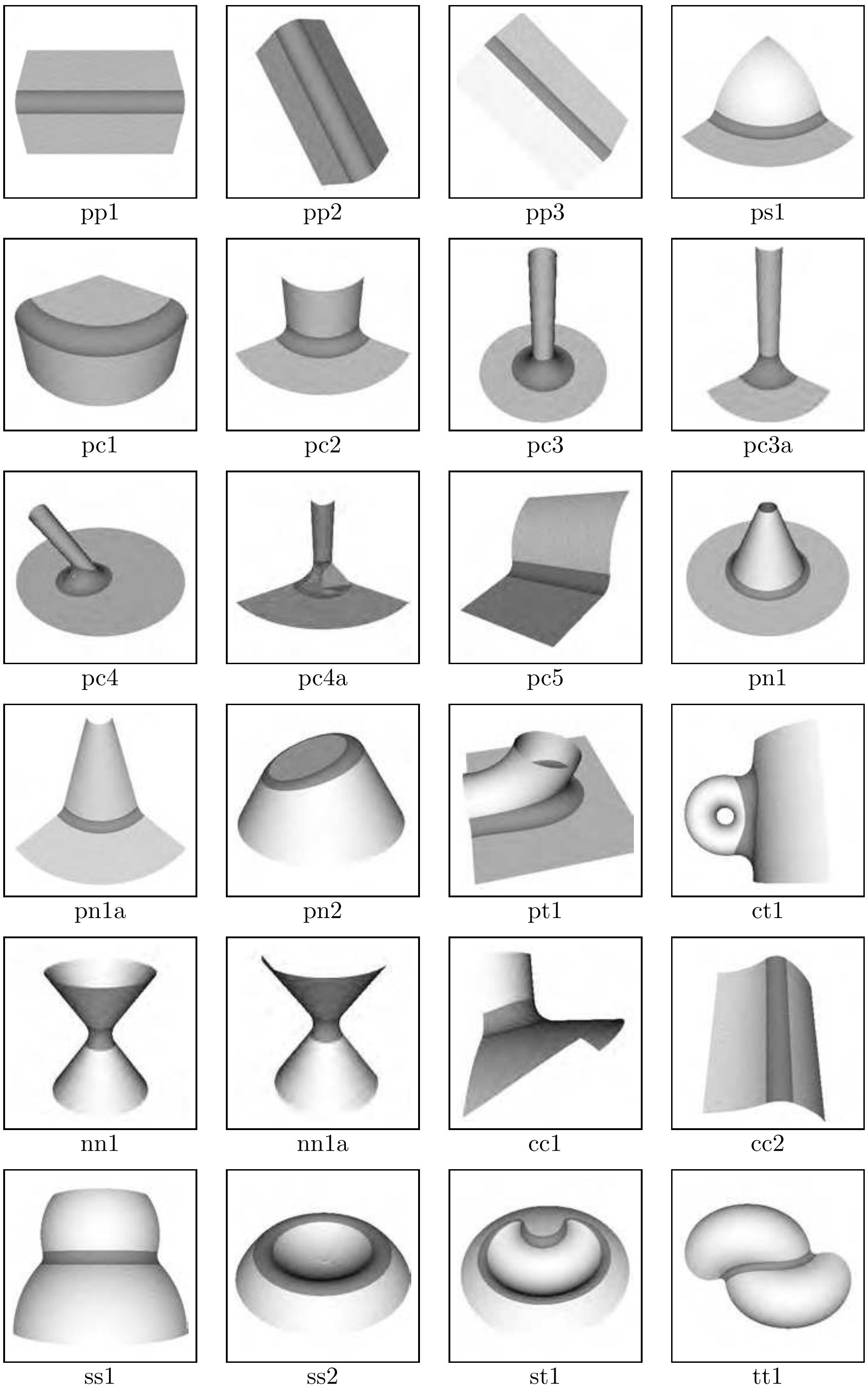

Figure 17: Blends used for testing

The test results presented in the rest of the paper only give results for some of these objects, and fuller results can be found in a technical report [12].

5.2 Applicability of each method

The curvature estimation method works for any type of primary surfaces, and it needs only the blend points. It can be used for any constant radius canal surface.

The iterative spine reconstruction and the maximum ball methods work in those cases when the primary surfaces are known. These methods can be used for parametric surfaces as well as implicit surfaces, but in this case they need another nested iteration to find the projections of points onto the primary surfaces.

The special purpose methods work when the blend is a cylinder or a torus. The direct surface fitting methods do not need any information about the primary surfaces. However, these are iterative methods requiring initial estimates of the axis and curvatures, which may be computed from surface normal estimates at the blend points. In turn, to estimate the normals, a dense set of neighbours is needed around each blend point.

Simple circle fitting needs the parameters of the primary surfaces. It can be used to compute the blend radius if the primary surfaces are both translationally or rotationally symmetric surfaces with the same direction or axis, respectively.

Constrained circle fitting can be generalised to compute the radius of a blend between two parametric translational or rotational surfaces with the same direction or axis respectively. In this case the projections of points onto the images of the primary surfaces can be computed by an iterative method.

5.3 Accuracy of each method

5.3.1 General purpose methods

Results of the general methods						
Dataset	Curv. est.		It. spine rec.		Max. ball	
Name	Clean	Noisy	Clean	Noisy	Clean	Noisy
pp3.05	10.0714	10.7846	10.0000	10.0029	10.0000	9.9927
pp3.10	10.0086	12.3538	10.0000	10.0056	10.0000	10.0221
pp3.20	10.0019	10.3853	10.0000	9.9976	10.0000	10.0079
pc1.05	10.1046	10.2908	10.0000	10.0080	10.0000	10.0103
pc1.10	10.0071	10.6763	10.0000	9.9965	10.0000	9.9967
pc1.20	10.0016	9.9276	10.0000	9.9977	10.0000	9.9983
pc3a.05	9.9176	10.1218	10.0000	10.0110	10.0000	10.0066
pc3a.10	10.0828	10.3424	10.0000	9.9981	10.0000	10.0003
pc3a.20	10.0073	8.3505	10.0000	9.9986	10.0000	9.9980
nn1a.05	9.9634	9.9382	10.0000	9.9974	10.0000	10.0061
nn1a.10	9.9898	6.7135	10.0000	9.9969	10.0000	9.9966
nn1a.20	10.0015	6.3623	10.0000	10.0023	10.0000	10.0028
st1.05	11.8786	12.3756	10.0000	10.0041	10.0000	10.0302
st1.10	9.8563	12.0367	10.0000	9.9961	10.0000	10.0291
st1.20	9.3835	10.7353	10.0000	9.9930	10.0000	10.0192
tt1.05	10.8646	10.7339	10.0000	9.9935	10.0000	9.9920
tt1.10	10.9222	10.2379	10.0000	9.9919	10.0000	9.9964
tt1.20	10.2550	10.3628	10.0000	10.0035	10.0000	10.0027

Table 1: Accuracy of general methods

Tables 1 and 2 show the estimated radii obtained by the general purpose methods and the distributions of errors respectively.

Error of general methods				
	Clean		Noisy	
Method	Average	Maximum	Average	Maximum
Curvature estimation	2.668%	18.786%	8.486%	41.917%
Iterative spine	0.000%	0.001%	0.051%	0.685%
Maximum ball	0.000%	0.001%	0.079%	0.715%

Table 2: Error of general methods

The curvature estimation method gives relatively poor results, even for clean data. If noise is present, the error can be greater than 20%. For concave blends, when the curvature is greater along the blend direction, for example in the cases of datasets **pc3a.20**, **nn1a.10** and **nn1a.20**, the program chooses the wrong side. In these cases, the difference between variation of the k_1 and k_2 values disappears in the noise. The only advantage of this method is that it does not need any information about the primary surfaces.

The accuracy of the iterative spine reconstruction is much better. It gives highly accurate results for clean data. Even when noise is present, the error in the result is less than 0.1% in most cases. The largest error was 0.69% during our experiments. The accuracy does not depend on whether the method of projecting the blend points onto the spine is iterative or explicit.

The results for the maximum ball method are similar to those for the iterative spine method, although the errors are slightly greater.

5.3.2 Special purpose methods

Results of the special methods								
Dataset	Cylinder fit		Torus fit		Simple circle		Constrained circle	
Name	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
pp3.05	10.0000	9.9830	—	—	10.0000	9.9817	10.0000	10.0016
pp3.10	10.0000	9.9731	—	—	10.0000	9.9731	10.0000	10.0043
pp3.20	10.0000	10.0171	—	—	10.0000	10.0118	10.0000	9.9962
ps1.05	—	—	10.0001	9.9833	10.0000	9.9833	10.0000	9.9973
ps1.10	—	—	10.0000	9.9664	10.0000	9.9664	10.0000	9.9992
ps1.20	—	—	10.0000	10.0019	10.0000	10.0019	10.0000	10.0042
pc5.05	10.0000	9.9951	—	—	10.0000	9.9959	10.0000	10.0002
pc5.10	10.0000	10.0115	—	—	10.0000	10.0262	10.0000	9.9913
pc5.20	10.0000	9.9972	—	—	10.0000	10.0001	10.0000	9.9977
pn1a.05	—	—	10.0000	9.9858	10.0000	9.9858	10.0000	10.0053
pn1a.10	—	—	10.0000	9.9883	10.0000	9.9885	10.0000	10.0053
pn1a.20	—	—	10.0000	9.9932	10.0000	9.9933	10.0000	10.0004
cc2.05	10.0000	10.0110	—	—	10.0000	10.0137	10.0000	10.0095
cc2.10	10.0000	10.0081	—	—	10.0000	10.0065	10.0000	9.9982
cc2.20	10.0000	9.9994	—	—	10.0000	9.9239	10.0000	10.0393
ss2.05	—	—	10.0000	9.9969	10.0000	9.9969	10.0000	9.9940
ss2.10	—	—	10.0000	9.9962	10.0000	9.9962	10.0000	9.9991
ss2.20	—	—	10.0000	10.0031	10.0000	10.0031	10.0000	10.0004

Table 3: Accuracy of special methods

Table 3 shows results for the special purpose methods. Table 4 shows the distributions of errors for these methods and for the general purpose methods when

Errors of methods in special cases				
	Clean		Noisy	
Method	Average	Maximum	Average	Maximum
Direct cylinder fitting	0.000%	0.000%	0.081%	0.269%
Direct torus fitting	0.000%	0.001%	0.072%	0.336%
Simple circle fitting	0.000%	0.000%	0.092%	0.761%
Constrained circle fitting	0.000%	0.000%	0.048%	0.393%
Curvature estimation	0.735%	4.986%	8.578%	37.167%
Iterative spine	0.000%	0.000%	0.049%	0.685%
Maximum ball	0.000%	0.000%	0.065%	0.715%

Table 4: Error of special methods

used on the same datasets.

The special purpose methods are all very accurate.

The simple circle fitting and the direct surface fitting methods give very similar results. The direct surface fitting methods do not use the parameters of the primary surfaces, but find the translation direction or the axis of the blend by iteration. The circle fitting methods use the parameters of the primary surfaces to compute the translation direction or axis respectively.

The constrained circle fitting method is roughly twice as accurate as the simple circle fitting method.

5.3.3 Overall conclusion

Clearly the curvature estimation method is much less accurate than the other methods, and is only useful if nothing is known about the primary surfaces. The maximum ball, iterative spine and simple circle fitting all have similar accuracies, the first two being of use for general primary surfaces. The constrained circle fitting and direct torus or cylinder fitting are about twice as accurate as the previous category, but rely on the blend and primary surfaces being of a particular type.

5.4 Speed of each method

5.4.1 General purpose methods

Tables 5 and 6 are organised in a similar way to Tables 1 and 2, but show computation times rather than accuracies. Times for fitting primary surfaces and building triangulations are excluded.

The slowest method is curvature estimation, the fastest is the maximum ball method.

A disadvantage of the curvature estimation method is that this method requires neighbourhoods to be constructed at each blend point, which is a time consuming process.

5.4.2 Special purpose methods

Similarly, Table 7 gives timing results for the special purpose methods.

There is no essential difference between the speeds of the simple and constrained (explicit or iterative) circle fitting methods. Direct cylinder fitting is slower, and torus fitting is very slow.

Comparisons with the general purpose methods are shown in Table 8.

Table 9 shows the times needed by the iterative spine reconstruction using iterative and explicit projection methods respectively.

Computation times of the general methods						
Dataset	Curv. est.		It. spine rec.		Max. ball	
Name	Clean	Noisy	Clean	Noisy	Clean	Noisy
pp3.05	0.40s	0.40s	0.08s	0.10s	0.00s	0.00s
pp3.10	1.36s	1.36s	0.12s	0.17s	0.01s	0.01s
pp3.20	5.23s	5.23s	0.28s	0.45s	0.04s	0.04s
pc1.05	0.64s	0.64s	0.12s	0.12s	0.00s	0.02s
pc1.10	2.42s	2.42s	0.20s	0.28s	0.03s	0.05s
pc1.20	9.50s	9.49s	0.58s	0.83s	0.10s	0.16s
pc3a.05	0.31s	0.31s	0.09s	0.11s	0.00s	0.01s
pc3a.10	1.21s	1.21s	0.15s	0.21s	0.01s	0.04s
pc3a.20	4.83s	4.84s	0.39s	0.64s	0.05s	0.14s
nn1a.05	0.45s	0.45s	0.13s	0.17s	0.00s	0.05s
nn1a.10	1.63s	1.64s	0.24s	0.37s	0.03s	0.09s
nn1a.20	5.69s	5.69s	0.50s	0.97s	0.11s	0.37s
st1.05	1.28s	1.28s	0.50s	0.47s	0.07s	0.07s
st1.10	4.08s	4.09s	0.71s	1.00s	0.23s	0.24s
st1.20	8.26s	8.24s	1.57s	1.98s	0.47s	0.48s
tt1.05	0.63s	0.63s	0.21s	0.22s	0.03s	0.03s
tt1.10	1.01s	1.00s	0.32s	0.38s	0.03s	0.03s
tt1.20	3.81s	3.79s	0.65s	0.97s	0.19s	0.20s

Table 5: Computation times of the general methods

Average computation times of general methods				
	Clean		Noisy	
Method	Average	Maximum	Average	Maximum
Curvature estimation	3.275s	13.74s	3.273s	13.71s
Iterative spine	0.345s	1.57s	0.503s	2.54s
Maximum ball	0.076s	0.47s	0.101s	0.63s

Table 6: Average computation times of the general methods

In blends between a cylinder and a plane, if the axis of the cylinder is perpendicular to the plane, the iterative method is faster. The iteration finds the projection of an arbitrary point onto the spine in a single step, because the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ do not change. If the axis is not perpendicular to the plane, the results show that explicitly computing the spine approximately doubles the speed.

5.4.3 Overall conclusion

Again, curvature estimation is a slow method which should only be used if nothing is known about the primary surfaces. Of the other general purpose methods, the maximum ball method is about four times faster than the iterative spine approach. For the special purpose methods, both circle fitting methods are very much faster than the surface fitting methods; they are also faster than the general purpose maximum ball method.

5.5 Ease of implementation of each method

It is difficult to compare the ease of implementation of the different methods. We implemented all the methods in C++.

Computation times of the special methods								
Dataset	Cylinder fit		Torus fit		Simple circle		Constrained circle	
Name	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
pp3.05	0.07s	0.15s	—	—	0.00s	0.01s	0.00s	0.00s
pp3.10	0.10s	0.17s	—	—	0.01s	0.01s	0.00s	0.01s
pp3.20	0.22s	0.29s	—	—	0.02s	0.01s	0.01s	0.02s
ps1.05	—	—	7.17s	7.22s	0.00s	0.00s	0.01s	0.00s
ps1.10	—	—	38.40s	26.26s	0.01s	0.00s	0.00s	0.01s
ps1.20	—	—	147.31s	334.20s	0.03s	0.03s	0.03s	0.03s
pc5.05	0.07s	0.14s	—	—	0.00s	0.01s	0.01s	0.00s
pc5.10	0.11s	0.14s	—	—	0.00s	0.00s	0.00s	0.00s
pc5.20	0.24s	0.30s	—	—	0.02s	0.02s	0.02s	0.02s
pn1a.05	—	—	10.84s	10.88s	0.00s	0.00s	0.00s	0.01s
pn1a.10	—	—	25.83s	31.38s	0.00s	0.01s	0.00s	0.00s
pn1a.20	—	—	53.43s	64.64s	0.01s	0.01s	0.01s	0.01s
cc2.05	0.08s	0.15s	—	—	0.01s	0.00s	0.00s	0.00s
cc2.10	0.13s	0.20s	—	—	0.01s	0.01s	0.01s	0.01s
cc2.20	0.32s	0.39s	—	—	0.03s	0.03s	0.03s	0.03s
ss2.05	—	—	14.35s	30.32s	0.01s	0.00s	0.00s	0.01s
ss2.10	—	—	105.36s	105.25s	0.02s	0.02s	0.02s	0.02s
ss2.20	—	—	78.22s	122.75s	0.04s	0.04s	0.04s	0.04s

Table 7: Computation times of the special methods

Average computation times in special cases				
	Clean		Noisy	
Method	Average	Maximum	Average	Maximum
Direct cylinder fitting	1.846s	10.04s	1.867s	10.04s
Direct torus fitting	57.002s	334.20s	68.997s	334.20s
Simple circle fitting	0.013s	0.04s	0.013s	0.04s
Constrained circle fitting	0.013s	0.04s	0.015s	0.04s
Curvature estimation	4.070s	13.74s	4.067s	13.71s
Iterative spine	0.316s	0.95s	0.487s	2.54s
Maximum ball	0.058s	0.26s	0.093s	0.63s

Table 8: Average computation times of the special methods, and comparison with general methods

Implementing the iterative spine reconstruction with explicit spine computation is messy, and works only in special cases. We have implemented only one of the simplest cases, when the primary surfaces are a plane and a cylinder, and the spine is an ellipse. In more general cases the spine can be any conic curve, and the type of the parametrisation depends on the nature of the conic.

The explicit constrained circle fitting is similar. We implemented only some of the relatively easy line-line and line-circle cases, when the blend radius itself can be used as the parameter of the bisector. In more general cases we have to distinguish at least four types of bisector.

5.6 Performance on real measured data

The algorithms were also tested on real measured datasets captured by a commercial REPLICA 3D scanner. One such example, a plastic bottle is shown in Figure 18.

Computation times of the iterative spine reconstruction with iterative and explicit spine computing				
Dataset	Iterative method		Explicit method	
Name	Clean	Noisy	Clean	Noisy
pc1.05	0.12s	0.12s	0.14s	0.48s
pc1.10	0.20s	0.28s	0.10s	1.05s
pc1.20	0.58s	0.83s	0.22s	4.85s
pc2.05	0.10s	0.11s	0.08s	0.14s
pc2.10	0.18s	0.25s	0.10s	0.63s
pc2.20	0.55s	0.97s	0.21s	1.27s
pc3.05	0.13s	0.15s	0.15s	0.14s
pc3.10	0.32s	0.44s	0.14s	0.52s
pc3.20	0.73s	1.35s	0.91s	1.75s
pc3a.05	0.09s	0.11s	0.10s	0.11s
pc3a.10	0.15s	0.21s	0.09s	0.21s
pc3a.20	0.39s	0.64s	0.51s	0.60s
pc4.05	0.49s	0.46s	0.26s	0.26s
pc4.10	0.49s	0.55s	0.28s	0.28s
pc4.20	0.53s	0.62s	0.33s	0.32s
pc4a.05	0.21s	0.26s	0.14s	0.16s
pc4a.10	0.21s	0.23s	0.15s	0.17s
pc4a.20	0.28s	0.31s	0.19s	0.21s

Table 9: Comparison of speed of the spine reconstruction

The top part of the object was scanned at a resolution of 0.5mm. Two blends are present. The lower blend occurs where the neck of the bottle meets the shoulder, while the upper blend occurs where the neck meets the topmost face. The primary surface regions and the blend regions are shown in Figure 19. Blends between a sphere and a cylinder were sought for both blend regions.

The number of data points used to estimate the radius of each blend were:

- Upper blend: sphere 730, cylinder 499, blend 613;
- Lower blend: sphere 1253, cylinder 2624, blend 1370;

The estimates produced for the radii of the blends are given in Table 10. On the lower blend, the centre of the sphere is not on the axis of the cylinder, and the special methods therefore could not be used. As can be seen, in the upper case these estimates were almost the same for the various methods, while in the lower case there is a larger difference between the radii obtained.

Upper blend	
Method	Radius
Curvature estimation	2.71
Spine iteration	2.61
Maximum ball	2.60
Direct torus fit	2.62
Simple circle fit	2.77
Restricted circle fit	2.77

Lower blend	
Method	Radius
Curvature estimation	3.29
Spine iteration	3.71
Maximum ball	4.35

Table 10: Results on measured data

Clearly the results obtained for practical data are not as good as for the sim-

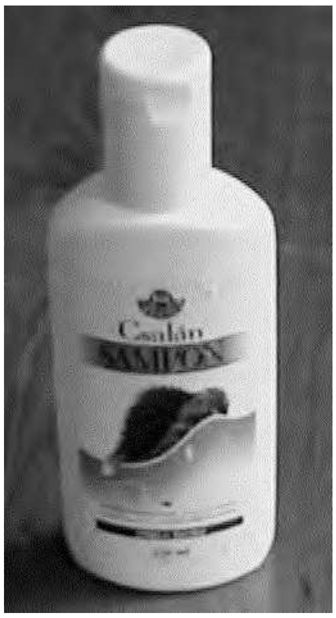

Figure 18: Plastic bottle

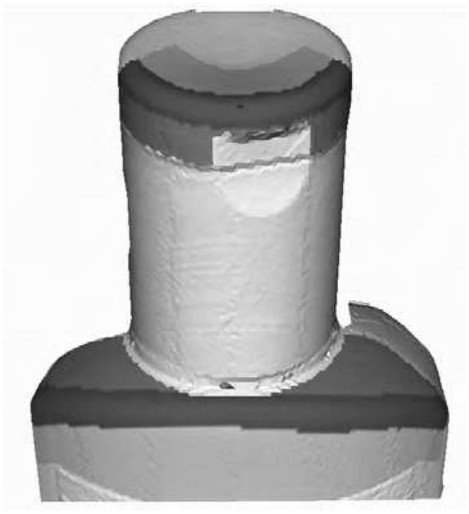

Figure 19: Segmentation of data into primary and blend surfaces

ulated data, but they are still quite usable for the intended purpose of reverse engineering. The practical results are more variable for a variety of reasons. Two of the most important are probably that errors in real scanner data do not follow a Gaussian distribution, and may contain outliers, and the segmentation of the data points is unlikely to be perfect.

6 Conclusions

We have investigated the problem of determining the radius of rolling ball blends from point data, a very important step in the reverse engineering of mechanical components and other objects. We have assumed that the measured data has been preprocessed and segmented, and most of our methods assume that representations of the primary surfaces have been determined.

We have described a variety of methods, some of which are of general use, while others are particular to certain special, but common, blending situations. Experimental tests have been performed on these methods which lead to the following conclusions.

The curvature estimation method is both slow and inaccurate, and should be avoided whenever possible, i.e. when information about the primary surfaces is known.

When a general purpose method must be used, and information about the primary surfaces is available, the maximum ball method is our method of choice, as it has comparable accuracy to the iterative spine method, but it is about four times faster.

When the blend surface takes the special form of a cylinder or torus, while the direct surface (torus or cylinder) methods are slightly more accurate than constrained constant circle fitting, the speed advantage of the latter suggests that in practice it should almost always be used instead. The unconstrained circle fitting method is about the same speed but less accurate, and so can be ignored.

In summary, while previous work on reverse engineering of geometric models has concentrated on the recovery of primary surfaces, it is clear that complete, realistic models also require the determination of blend characteristics. The new methods and analysis given in this paper hopefully provide an appropriate basis for achieving this goal.

7 Acknowledgements

The bulk of this project was accomplished while the first author was a visiting researcher at the Computer Science Department, Cardiff University, whom he wishes to thank. All authors are grateful for support from EU COPERNICUS grant (RECCAD 1068), which provided an ideal framework for this joint research. The support of the National Science Foundation of the Hungarian Academy of Sciences grant 26203 is also highly appreciated.

The authors would also like to thank other members of the RECCAD team for many helpful discussions.

References

[1] P. Benkő, G. Kós, T. Várady *Region growing for regular surfaces.* Geometric Modelling Studies, GML 98/3 **(1998)**, Computer and Automation Research Institute, Budapest

[2] P. Benkő, G. Kós, G. Lukács, T. Várady *Reverse engineering solid objects bounded by regular surfaces.* Geometric Modelling Studies, GML 98/4 **(1998)**, Computer and Automation Research Institute, Budapest

[3] P. J. Besl, R. C. Jain *Segmentation through Variable-Order Surface Fitting.* IEEE PAMI, Vol 10, No 2, **(1988)** pp 167–192

[4] R. M. Bolle, B. C. Vemuri *On Three-Dimensional Surface Reconstruction Methods.* IEEE PAMI, Vol 13, No 1, **(1991)**, pp 1–13

[5] O. D. Faugeras, M. Hebert, E. Pauchon *Segmentation of Range Data into Planar and Quadric Patches.* Proc. 3rd Comp. Vision and Patt. Recog. Conf., Arlington, VA, **(1983)** pp 8–13

[6] A.W. Fitzgibbon, D. Eggert, R.B. Fisher *High-level CAD Model Acquisition from Range Images.* Computer-Aided Design, Vol. 29, No 4, **(1997)** pp 321–330

[7] C. M. Hoffmann *Geometric and Solid Modelling, An Introduction.* Morgan Kaufmann, San Mateo, CA, **(1989)**

[8] A. Hoover, D. Goldgof, K. W. Bowyer *Extracting a valid boundary representation from a segmented range image.* IEEE PAMI, Vol 17, No 9, **(1995)** pp 920–924

[9] A. Hoover et al. *An Experimental Comparison of Range Image Segmentation Algorithms.* IEEE PAMI, Vol. 18, No. 7 **(1996)**, pp 673–689

[10] H. Hoppe et al. *Mesh optimization.* Computer Graphics (SIGGRAPH'93 Proceedings), Vol 27, **(1993)**, pp 19–26

[11] J. Hoschek, D. Lasser *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, **(1993)**

[12] G. Kós, R. R. Martin, T. Várady *Methods to recover constant radius rolling ball blends in reverse engineering.* RECCAD Deliverable Document 4, Copernicus project No. 1068, Geometric Modelling Studies, GML 98/4 **(1998)**, Computer and Automation Research Institute, Budapest

[13] A. Leonardis, A. Gupta, R. Bajcsy *Segmentation of Range Images as the Search for Geometric Parametric Models.* Int. J. Computer Vision, Vol 14 **(1995)** pp 253–277

[14] G. Lukács, R. R. Martin, D. Marshall *Faithful least-squares fitting of spheres, cylinders cones and tori for reliable segmentation.* Computer Vision-ECCV 98, Vol. 1406 of Lecture Notes in Computer Science, Eds: H. Burkhardt and B. Neumann, Springer-Verlag **(1998)**, pp 671–686. (Proceedings of 5th European Conference on Computer Vision–ECCV'98 Freiburg, Germany, June 2-6, 1998)

[15] H. Pottmann, T. Randrup *Rotational and helical surface reconstruction for reverse engineering.* Computing **(1998)**, pp 307-322

[16] V. Pratt *Direct least-squares fitting of algebraic surfaces.* Computer Graphics, Vol 21, ACM, (Procs. of SIGGRAPH'87), (1987), pp 27-31

[17] W. J. Schroeder, J. Zarge, W.E. Lorensen *Decimation of triangle meshes.* Computer Graphics (SIGGRAPH'92 Proceedings), Vol 26, **(1992)**, pp 65-70

[18] T. Várady, R. R. Martin, J. Cox *Reverse Engineering of Geometric Models — An Introduction.* Computer-Aided Design, Vol. 29, No 4, **(1997)** pp 255-268

[19] T. Várady, C. M. Hoffmann *Vertex blending: problems and solutions.* Fourth Intl. Conf. on Math. Methods for Curves and Surfaces, Eds. M. Daehlen, T. Lyche, L. L. Schumaker, Vanderbilt University Press, **(1998)** pp 501–527

[20] T. Várady, P. Benkő, G. Kós *Reverse Engineering regular objects: simple segmentation and surface fitting procedures.* International Journal of Shape Modeling, Vol. 4, No. 3&4 **(1998)** pp 127–141

[21] J. Vida, R. R. Martin, T. Várady *A survey of blending methods that use parametric surfaces.* Computer-Aided Design, Vol 26, No 5, **(1994)**, pp 341–365