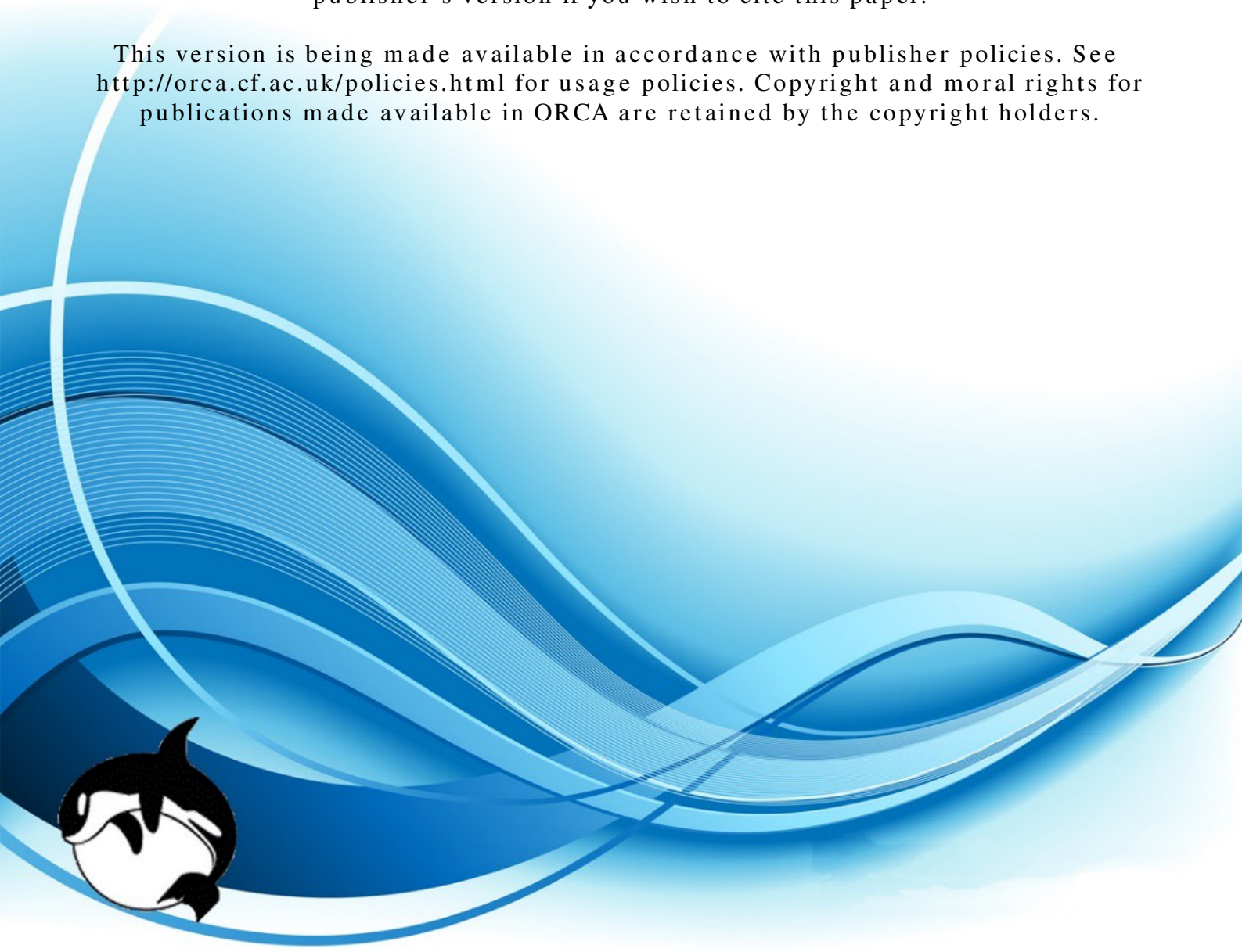

Algorithms for Reverse Engineering Boundary Representation Models

Pál Benkő, Ralph R. Martin (*), Tamás Várady,
Computer and Automation Research Institute, Budapest
(*) University of Wales, Cardiff
e-mail: benko@sztaki.hu

Abstract

A procedure for reconstructing solid models of conventional engineering objects from a multiple-view, 3D point cloud is described. (Conventional means bounded by simple analytical surfaces, swept surfaces and blends.) Emphasis is put on producing accurate and topologically consistent boundary representation models, ready to be used in computer aided design and manufacture. The basic phases of our approach to reverse engineering are summarised, and related computational difficulties are analysed.

Four key algorithmic components are presented in more detail: efficiently segmenting point data into regions; creating translational and rotational surfaces with smooth, constrained profiles; creating the topology of B-rep models; and finally adding blends. The application of these algorithms in an integrated system is illustrated by means of various examples, including a well-known reverse engineering benchmark.

1. Introduction

Reconstructing objects has been a central problem in computer vision for a long time [7, 8, 22]. Technological advances in laser scanning now make it possible to produce multiple 3D point clouds with high density and high accuracy, and in this way it has become a realistic expectation to generate accurate models with correct topology which can be directly utilised by CAD/CAM systems. There is a wide diversity of reverse engineering methods for converting measured data into geometric models [38]. Methods differ in the quality of measurements, the quantity of points, the geometric characteristics of the objects in question, the amount of user interaction required, and the final representation used for the model.

About twenty years ago, the two main areas of geometric modelling development separately addressed (i) geometrically demanding free-form surface models and (ii) topologically complex solid models with simple surface elements. These techniques were eventually integrated into commercial CAD/CAM systems. The current situation in reverse engineering is somewhat similar. For free-form shapes, the current state-of-the-art relies on techniques in which the user manually segments point clouds. Segmentation is

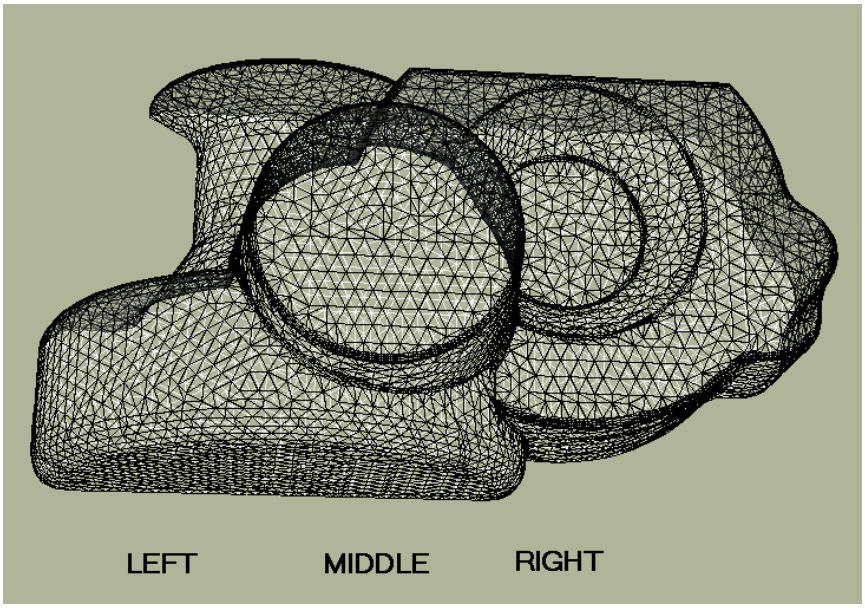

Figure 1. Triangulated and decimated data points

followed by fitting high-quality surfaces [24, 25] and the result is typically a collection of trimmed parametric surface patches. For conventional engineering objects, (i.e. those are bounded mainly by simple analytic surfaces, swept surfaces and blends), the major efforts are directed towards *automatic methods* of building accurate and consistent CAD models with little or no user interaction. These approaches rely on various assumptions concerning the regularity of the objects which make it possible to automate the process.

Although there are many valuable contributions in computer graphics and computer vision concerning object reconstruction, the majority of these cannot directly be applied to building CAD/CAM models for mechanical engineering. Some produce fair, triangulated, approximations suitable for graphics (see e.g. Hoppe et al. [23]), while others represent the boundary of objects using special surface representations such as A-patches or superquadrics (see for example [2, 31]). Such representations are of limited use as commercial CAD/CAM systems must use standard representations for interoperability.

Approaches exist which use standard free-form surfaces, such as B-splines, to reconstruct conventional engineering objects — see [13, 17, 26, 37]. Two difficulties arise. If multiple surfaces are used, it is hard to find good segmenting curves, and to assure C^0 or higher continuity between the adjacent surfaces. On the other hand, fitting a single B-spline surface to the whole data cannot properly reproduce the piecewise nature of the surface with appropriate curvature distribution. Even if these limitations could be overcome, another problem remains. Representing the separate analytical surfaces of the original design is important in order to be able to edit the model in a meaningful way, to be able to plan correct manufacturing operations, and for other related purposes. A free-form model does not provide this information.

The above arguments justify dedicated approaches for conventional engineering objects, such as the well-known benchmark in Figure 1, which was put forward a few years ago by Hoschek [24]. From a functional point of view, this part is the union of three components. LEFT has rotational symmetry, MIDDLE is a part of a blended cone, and RIGHT is bounded by two planar faces and a cone, as well as

a sequence of extruded side faces. All intersection edges have been blended by constant radius rolling ball blends. Clearly, representing this shape with a single B-spline surface would be inadequate, even if possible.

We now state the problem we wish to solve. We assume an ungridded, dense set of point data is available, formed by merging multiple views. We assume the data comes from a conventional object bounded mainly by relatively large *primary surfaces* of simple analytic surface type (planes, natural quadrics and tori), which must be accurately represented as such, not as B-splines. Translationally and rotationally swept surfaces must be recognised and represented by means of swept profiles. Fixed-radius *blends* must also be reconstructed as small transition surfaces which smoothly join the primary surfaces. If two adjacent faces of the original object are tangent continuous, the reconstructed model should also guarantee smoothness. Since surface fitting to measured data sets is inherently inaccurate, we must recognise appropriate engineering constraints such as perpendicularity, parallelism, and concentricity, and enforce them on the model either at the fitting stage or as a postprocess.

In comparison, the majority of previous reverse engineering approaches only considered single view range data, and only concerned itself with simple surfaces—see for example [10, 16, 21, 34].

In the remainder of the paper, in Section 2, we outline our automatic reverse engineering approach for conventional objects, surveying relevant ideas and identifying the successive computational steps. Then, in Sections 3–6, four particular algorithmic components are chosen and discussed in more detail: segmenting point clouds, reconstructing translational and rotational surfaces, building the topology of B-rep models, and adding blends.

2. Basic phases of reconstructing solid models

Our methodology for the reverse engineering procedure can be decomposed into the following steps:

1. Data capture
2. Merging multiple point clouds
3. Triangulation/decimation
4. Segmentation
5. Surface fitting for simple surfaces
6. Reconstructing translational and rotational surfaces
7. Reconstructing smooth multiple regions
8. Building an adjacency graph
9. Constraint identification, constrained fitting
10. B-rep model creation, further beautification
11. Blend reconstruction

For *data capture*, typically laser scanners are used, which produce measured data sets with high density and reasonable accuracy, although outliers may occur, particularly near silhouette curves and concave edges. There are missing portions due to occlusion in single views, meaning that discontinuities in depth occur, and this may hold even for merged point clouds too.

A single view represents only a part of an object. In order to get a complete (or at least a multiple-view model), point clouds need to be *merged*, i.e. appropriate transformations need to be found by means of which the different data sets can be represented in a common coordinate system. This is often called *registration*, which is an important problem in computer vision [5, 12]. Iterative closest point methods match nearest points in several datasets. Such methods tend to be slow, and require relatively good initial estimates of the registration. Some progress has been made towards overcoming these difficulties [30]. An efficient hierarchical method can be used based on computing characteristic curves in the point data and then special points on those curves. Registration is first done using the points, then refined using the curves and finally made accurate using the whole point set. This approach is even effective for obtaining an initial registration. Secondly, iterative *reciprocal* closest point methods are more robust when there is relatively little overlap between the data sets.

Other registration methods use calibrating balls or special markers to find the correspondence between the individual point sets. High-precision rotary tables are often used, since transformations can robustly be computed; however, in this case the bottom face and other downward pointing faces are not visible.

In fact, there is more than one approach for producing models from multiple views. If it is done by *merging point clouds* as described above, a vast data structure is obtained with large, united point regions, which may suffer from registration inaccuracies and uneven, irregular point distributions. The alternative is to create single view, partial B-rep models: in this case we have to deal with discontinuities, but simpler models are obtained. However, the single view models must be consistent and accurate to be able to *combine* them using Boolean operations to produce a final model. The pros and cons of point cloud merging and combining single-view models are discussed in detail in [3].

Even a single point cloud may contain at least 10^4 and perhaps as many as 10^8 data points. In order to determine the surface of an object, first the neighbourhoods and connectivity of the data points must be computed. This is a difficult problem, particularly if we have a merged data set with noise, uneven point distributions, discontinuities, holes and disjoint parts. There are various techniques to build a consistent *triangulation*, which differ in their basic assumptions, their generality and their computational efficiency — see for example [2, 14, 23]. Methods based on variants of α-shapes are popular but require careful implementation and the proper selection of parameters to achieve successful results [18]. As an alternative, we use an efficient method suggested by Kós [28], in which locally created Delaunay triangulations are merged in a consistent manner.

For efficiency reasons, the original triangulation often needs to be *decimated*, forming a so-called *carrier surface*. Decimation needs to be an adaptive process taking into account the local surface geometry; it is a widely studied problem to control the amount of data displayed in computer graphics [20]. The carrier surface reflects the original topology of the point cloud; each data point is either a vertex of the triangulation or associated with the closest triangle of the carrier. The carrier surface plays an important role in speeding up further computations.

One of the most crucial elements of reverse engineering is *segmentation*, i.e. identifying point regions which are disjoint subsets and mark out connected portions within the whole point cloud sharing some common property. These are considered to be pre-images of single faces (or possibly multiple faces) of the B-rep model. For each face a single surface is *fitted* to the related points within a given tolerance.

Various methods for fitting simple analytic surfaces have been described in [8, 15, 16, 19, 32, 40]. We believe that the use of geometric, or geometrically faithful, distances is more satisfactory for surface reconstruction than algebraic distances.

In general, segmentation is a complex process [38]. Often, iterative *region growing* techniques are applied [7, 31, 37]. While it is possible to apply the region growing paradigm to segment conventional engineering objects [33], in this paper we prefer another solution called *direct segmentation* [40]. This method is an efficient, non-iterative approach, which satisfies the requirements we set for the accurate reconstruction of solid models.

Direct segmentation is based on the fact that it is possible to compute local characteristic quantities (e.g. normal direction) within the interior of faces. If these quantities are consistent within a neighbourhood, this implies that we are in the interior of some region. If they change rapidly or fluctuate, we may deduce that we are in between two or more regions. Based on this principle we can segment the point cloud. We discard points in the neighbourhood of unstable estimates and use only the stable data points for surface fitting. As described in Section 3, direct segmentation first splits the object into smaller regions along sharp edges and small radius blends. For each region an attempt is made to approximate the points by a *single* analytic surface. The regions for which this cannot be done are considered *multiple, smooth regions* and need to be further subdivided. We first search for *translational or rotational* symmetries, and if either is found we determine a smooth, sweeping *profile* (see Section 4) and split the region accordingly. If further multiple, smooth regions remain, we need more sophisticated segmentation methods. These regions contain surface elements with tangential contact and surfaces must be simultaneously fitted with smoothness constraints.

Direct segmentation produces disjoint regions, each of which is approximated by a simple analytic or swept surface. During this process we have also identified *sharp edges*, which can be computed by surface-surface intersection, and *smooth edges* where surface-surface intersection must not be used. Smooth edges are directly computed by means of *constrained fitting*.

Based on the segmented regions, a so-called *region adjacency graph* is built, which reflects the complete topology and serves as the basis for building the final B-rep model, as discussed in Section 5. The actual construction of B-rep models is generally performed using a solid modelling kernel. We prefer to build so-called *stitched* models, where the individual faces are glued together along their common edges, and so a composite shell object is formed gradually. Alternatively, as mentioned earlier, it is possible to produce *volumetric* models by combining prismatic objects constructed from single views [3].

Smooth edges which are the boundaries of *blending surfaces* occur in a different way. Blends are temporarily ignored during segmentation and fitting, and are reconstructed only in the very last phase of model building [27]. Blending surfaces are constructed to be tangential to the primary surfaces: see Section 6.

A crucial step is the *beautification* of the final model. This can take place both before and after B-rep model building. In the presence of noisy measured data, the generated model is likely to be imperfect. The exclusion of very small edges and facets, filling of little holes, ensuring orthogonality, parallelism, concentricity, etc. are all important requirements for real-life CAD/CAM models. The latter types of beautification can also be solved by constrained fitting. As discussed earlier, for composite regions, smoothness constraints are added and the surface elements are simultaneously fitted. Similarly, for example, if two planes are found to be nearly perpendicular, the points of the two related surfaces can be refitted under a perpendicularity constraint, which will lead to a better model. There are many papers on constrained geometric modelling [9], but its relevance to reverse engineering has been recognised only

recently [4, 41].

Having concluded the overview of the reconstruction process, we continue by providing more details of particular algorithmic phases within the above general framework.

3. Direct segmentation

In this section we describe the basic concept of direct segmentation, but lack of space prevents us from giving full details, for which see [4, 40]. Our purpose is to subdivide the point data into disjoint subsets, which are smooth and to which one (or more) surfaces can locally be fitted. As noted earlier, the fundamental structure of the object is determined by primary surfaces. Along the intersection of two primary surfaces there is either a sharp edge or a small-radius blend. Alternatively the surfaces may be smoothly connected.

In the first stage of direct segmentation we attempt to separate regions by removing triangles which are in the vicinity of sharp edges or small blends. Thus, disjoint regions are obtained, which can be classified either as *simple*, i.e. which can be approximated by a single analytic surface, or as *multiple*, which need to be further subdivided by smooth edges.

An important feature of the current algorithm is that, for each data point, we estimate a surface *normal vector* and a value which characterises the *planarity* of the point neighbourhood. Our experience shows that the best normal vector estimates are based on fitting a second order algebraic surface to surrounding points in the neighbourhood. This means that first we translate all the points in the neighbourhood so that the given point $\mathbf{p}$ goes to the origin (the image point set is denoted by P'), then we look for a symmetric matrix A and unit vector $\mathbf{d}$ which minimise

$$\sum_{\mathbf{x}\in P'} \left(\mathbf{x}^T A\mathbf{x} + \langle \mathbf{d}, \mathbf{x}\rangle\right)^2 .$$

This leads to a three dimensional eigenvalue problem and the normal we seek is just $\mathbf{d}$. $\langle\cdot,\cdot\rangle$ denotes the scalar product of two vectors.

The planarity is characterised by taking a point and the related least-square plane fitted to its neighbours, and computing the normalised fitting errors over a small disk: for relatively smooth primary surfaces this value will be small, while for highly curved parts this value will be greater. There are other alternatives to identify highly curved parts using the triangulation; for example, the angular variation of normals was suggested in [26], or curvature estimates can be computed [1].

After choosing an appropriate threshold value, planarity filtering will remove those triangles which are classified as belonging to highly curved regions of the point cloud. For example, in Figure 2, we have filtered out the small blends between the components LEFT, MIDDLE and RIGHT, and the small blends between the side faces and the top and bottom planar faces. The larger blends contained in the rotational and translational regions were kept. We obtained two smooth, multiple regions, as well as many simple regions.

Note that our example object is a variant of the original benchmark of Hoschek, which did not contain sufficiently dense data for our purposes, making it impossible to reconstruct blends reliably. Our variant was redesigned and sampled in such a way that we can demonstrate important features of our algorithm.

Given a region, we do not immediately know whether it is simple or multiple. A sequence of hypotheses is tested to decide whether the region's point set can be approximated by some simple surface.

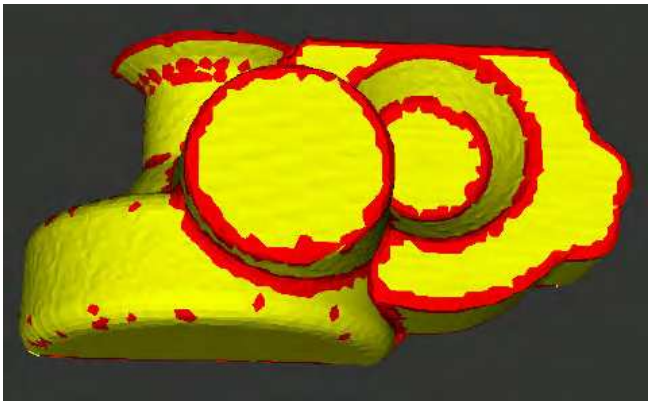

Figure 2. Planarity filtering — simple and smooth, multiple regions

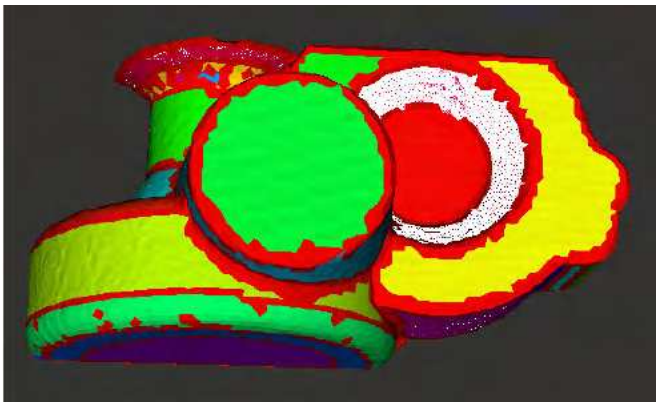

Figure 3. Segmented object, simple translational and rotational surfaces detected

If an hypothesis fails, we proceed to try more complex surface types. Finally, multiple regions are decomposed. We always proceed in order of increasing complexity. Fortunately, simpler surfaces (i) occur more frequently in mechanical engineering, and (ii) can be detected in a more reliable and efficient manner. The decision procedure thus follows the simplified scheme below:

1. Test whether the region is a plane or a sphere; if so: determine parameters. OTHERWISE
2. Test for translational symmetry; if so:
   (a) Test whether it is a cylinder; if so: determine parameters, ELSE
   (b) Test whether the sweeping contour can be decomposed into a smooth sequence of straight line segments and circular arcs; if so: multiple region, ELSE
   (c) Compute a free-form profile. OTHERWISE
3. Test whether it is a cone; if so: determine parameters. OTHERWISE
4. Test for rotational symmetry; if so:
   (a) Test, whether it is a torus, if so: determine parameters, ELSE
   (b) Test whether the sweeping contour can be decomposed into a smooth sequence of straight line segments and circular arcs if so: multiple region, ELSE
   (c) Compute a free-form profile. OTHERWISE
5. Decompose non-regular, multiple smooth regions
   (a) Detect planes
   (b) Detect translational subregions
   (c) Search for independent axes of rotation, and for each axis detect rotational subregions

(For 5b and 5c proceed as in 2 and 4.)

The above tests are based on consecutive least-squares procedures for the geometric parameters of these simple surfaces — see [40]. The scheme shows the significance of detecting translational and rotational symmetries, which are based on the surface normal estimates associated with each point.

We will give details of certain key steps shortly, but in the meantime two minor points are worthy of note in the above procedure. Firstly, consider cone fitting. True least-squares cone fitting, of course, leads to a nonlinear system of equations. We may detect a cone as a rotational surface with a linear profile, but our experiments show that estimating the apex by a least-squares fit using the tangent planes at each data point, and computing an least-squares fit plane for the normal vectors on the Gaussian sphere provides more reliable results. This process gives the axis direction and the half angle of the cone.

Secondly, consider general free-form surfaces. The last paragraph of the decision hierarchy above should include an additional item (d), to extend the concept to include free-form surfaces too. Here our hypothesis would be that for all isolated regions which have not been confirmed earlier to be a simple analytic or swept surface with either straight-circular or free-form profile, there exists a B-spline surface which can well approximate the remaining data points. Of course, in these cases special care is required to enforce at least numerical positional and tangential continuity with the surrounding analytic geometric elements.

3.1. Determining the Translational Direction

We can estimate the translational direction by means of a least-squares fit. The normal vectors of a translational surface must be perpendicular to a common translational direction. We may minimise the deviation of these angles from $\pi/2$, but the simplest method is to minimise the cosine of the angles, which is just the scalar product of the given normal vectors and the unknown direction. The translational characteristic of a shape can be visualised with the help of the Gaussian sphere: the normal vectors of a translational surface lie on a great circle, that is, in a plane which goes through the origin and is perpendicular to the translational direction.

Formally: if the estimated normal vectors of the region are denoted by $\{\mathbf{n}_i\}$, then we look for the unit vector $\mathbf{d}$ which minimises

$$\sum \langle \mathbf{n}_i, \mathbf{d} \rangle^2 .$$

This is a well-known three dimensional eigenvalue problem.

3.2. Determining the Rotational Axis

The normal lines of a rotational surface must intersect a common axis (in a projective sense, i.e. the lines may also be parallel to the axis). The first problem is how to measure the error of a noisy normal line. Using the distance between the axis and the normal line has two problems: firstly, a normal line parallel to the axis does not have zero error; secondly, the error in a line near to the axis is penalised less than a line farther from the axis with the same angular deviation. Another idea is to use the angle between the normal line, and the plane spanned by the axis and the data point corresponding to the normal line. Now we do not face the first problem as above, but have the opposite of the second problem: the noise in a data point near to the axis is penalised more than in a further point. Using the product of the distance and the sine of the angle defined above, however, eliminates the problems (and has the advantage of

being easy to compute as will be shown below). This measure was suggested by Pottmann and Randrup in [36].

We cannot go into details here, and just the basic concepts are presented. Lines are represented using Plücker coordinates. A line with direction $\mathbf{d_i}$ through point $\mathbf{p_i}$ is represented by a sextuple $(\mathbf{d_i}, \mathbf{d_i} \times \mathbf{p_i})$ (the latter term, the *momentum vector* is denoted by $\bar{\mathbf{d}}_\mathbf{i}$. Note that it is independent of the choice of $\mathbf{p_i}$ on the line). The axis a is also given in the same form as $(\mathbf{a}, \bar{\mathbf{a}})$.

Using this formalism, our error function is linear in the coordinates of the unknown axis: we minimise

$$\sum [\langle \bar{\mathbf{d}}_\mathbf{i}, \mathbf{a} \rangle + \langle \mathbf{d_i}, \bar{\mathbf{a}} \rangle]^2.$$

We solve the above least squares expression for $(\mathbf{a}, \bar{\mathbf{a}})$ under the constraints $\|\mathbf{a}\| = 1$, $\langle \mathbf{a}, \bar{\mathbf{a}} \rangle = 0$ (which are needed for normalisation and for the 6 numbers to be correct Plücker coordinates). Initially, the second constraint is omitted, and the remaining system (which is a generalised eigenvalue problem) is solved. Then the full system is solved iteratively, the initial result being used as a starting value. Generally just a few (three or four) iterations are needed. For a geometric interpretation of the initial value see [36].

3.3. Multiple regions

In the second stage of direct segmentation we decompose any remaining smooth multiple regions. Many are defined by sweeping smooth profiles, which consist of tangential line-segments-and-circular-arcs or free-form profiles. Assuming translational or rotational symmetry has been detected in the given region, we can compute the best sweeping profile, as detailed in the next Section. In other cases in step 5, we deal with *general* multiple regions. Figure 4 shows a simple object, by means of which we illustrate in Section 5 how the topology of the final model is built . The object has not only a translational region which includes the five faces (TF1-TF5), but a general, multiple face region (SF1–SF7) as well. This latter has one toroidal, three planar, and three cylindrical faces, which were generated presumably in the original object by means of a blending operation, but due to the large radii used, these were not separated in the planarity filtering phase of reverse engineering.

The standard technique for segmenting regular surfaces is to analyse the signs of the estimated mean and Gaussian curvatures [6]. The curvatures can be estimated based on the triangulation, or by fitting the best fit quadric at the given point. For noisy data, we apply a useful tool called *dimensionality filtering*. Taking the normal vectors of a multiple region mapped onto the Gaussian sphere, the normals of any planar region will be mapped into a point cluster; those of a translational or conical region onto a circular arc. Classifying the normal vector distributions on the Gaussian sphere as 0- or 1-dimensional, the planar and translational surfaces can be separated and detected [40].

The only problem left is when more than one rotational surface exists within a smooth subregion. In this case, we may identify all likely rotational axes using an approach suggested by Lukács in [32]. The basic idea is that any four normal lines determine two axes which intersect all four lines, and which can be computed easily by a second degree equation. By clustering the potential axes, we detect rotational surfaces locally. This step concludes the direct segmentation procedure.

Returning to our segmented benchmark object in Figure 3, the regions are coloured differently and detected simple surfaces have already been fitted. The multiple regions, the rotational axis of LEFT and the sweeping direction of the side face of RIGHT have also been computed and decomposed according to their smooth profile curves, i.e. we have generated a sequence of connected cylinders, cones, and tori

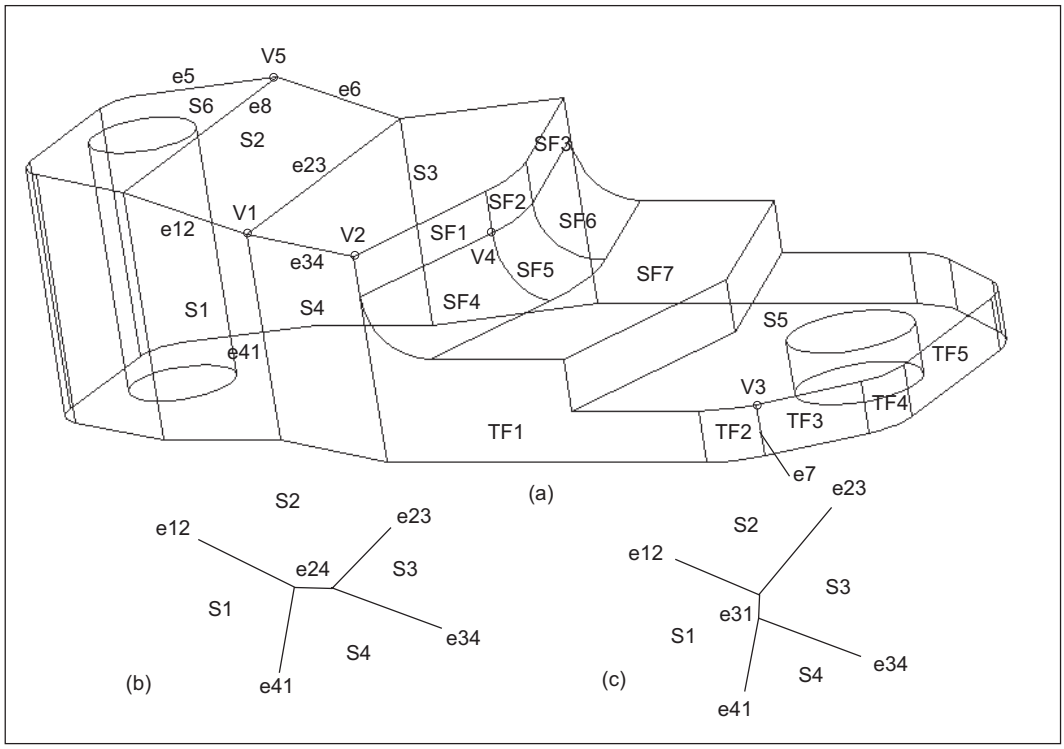

Figure 4. Simple object

for LEFT, and connected cylinders for RIGHT, respectively. The segmentation of the other simple test object can be seen in Figure 5. These point regions are sufficiently stable to serve as an initial value for constrained fitting which enforces smoothness between adjacent surface elements. Our approach to constrained fitting is given in [4].

4. Constructing swept surface profiles

We continue with our approach to computing profiles curves for swept regions. These are made up of smoothly connected straight line segments and circular arcs. The underlying smooth, multiple region has already been detected in the segmentation phase.

4.1. Guiding Polygon

The *guiding polygon* is a key element in profile fitting. Approaches which are based only on point data may find it difficult to approximate the point set after projection. Due to the inaccuracies of the translational direction or rotational axis, we may obtain a point cloud which is 'thick', and thus it is hard to order the points, a step which is generally needed for approximating methods. An approach for thinning point clouds was reported recently in [29]. We propose to work over the triangulation, which has the advantage that not only the point information, but their *connectivity* can be transformed into the plane of the profile.

Once we know the translational direction or the rotational axis, we can create a polygon which approximates the profile by intersecting the triangulation with a suitably chosen plane (perpendicular to

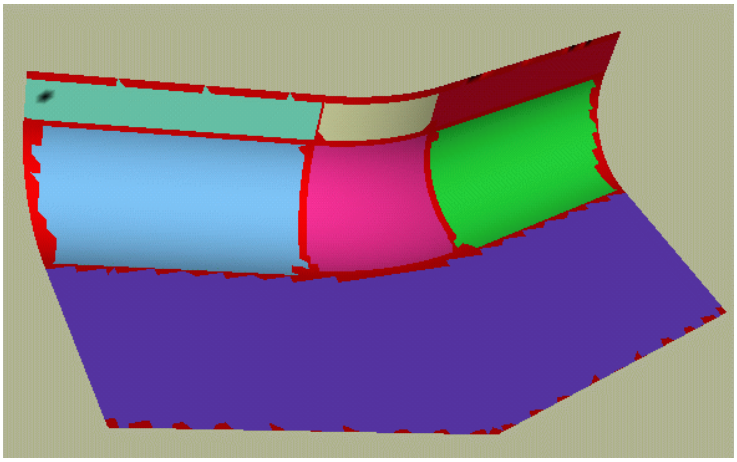

Figure 5. Segmenting the multiple smooth region of the simple object

the translational direction or containing the rotational axis as appropriate). In choosing a suitable plane, the problem is that many candidate profile planes do not contain the *entire* profile due to trimming of the swept face by adjacent faces — it may even be the case that *all* planes have this unfortunate property. This is why we have to apply a gradual approach to build up the complete polygon. We start at an arbitrary triangle of the swept region, chosen near the centre of the point set. We intersect it with the profile plane through its barycentre and extend the polygon by taking as many adjacent triangles intersected by that plane as possible. Of course, this may be just part of the guiding polygon and it may need to be further extended. After building the initial polygon we check all triangles remaining in the region. Each triangle is projected (or rotated) into the profile plane and its image is projected onto the polygon. If the current polygon does not fully cover the image, the polygon must be extended accordingly. In order to build up the polygon correctly, we do not traverse the triangles in an arbitrary order: we always take a triangle adjacent to a previously processed one.

If the distance between the two endpoints of the polygon becomes too small (say, smaller than the average distance of adjacent vertices), we close the polygon and stop the building process.

4.2. Constrained Profiles

Of course, every profile can be approximated by a free-form curve, but many engineering applications require an explicit straight line–circular arc representation of the profile where applicable. While for free-form curves the guiding polygon provides a sensible parametrisation of the projected points, here it helps to segment the profile into circular arcs and straight line segments by applying a 'recover & select' [31] procedure.

Circles are represented in Pratt's form [35], which includes lines as special cases and not as singularities. After segmentation, circles whose radii are too great are replaced by straight lines. Then the points are grouped with the help of the guiding polygon and either a circle or a straight line is fitted to each group, such that each is tangent to its neighbouring elements. There are two strategies to achieve this goal. One of them is to fit the elements sequentially, always satisfying the smoothness constraint with the previous element. This strategy may lead to a contradiction in the case of a closed profile; moreover, errors are accumulated: the first object to be fitted will be quite exact, but as we proceed we obtain worse and worse approximations. We prefer an alternative strategy which simultaneously approximates

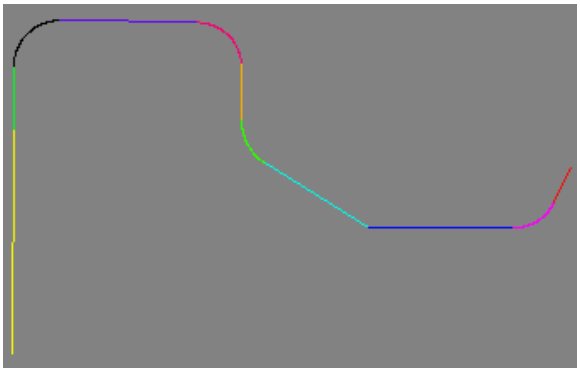

Figure 6. Reconstructed rotational profile

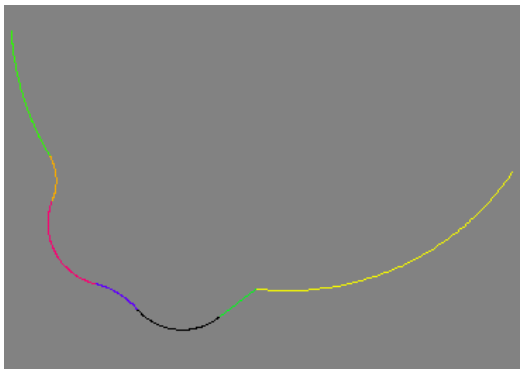

Figure 7. Reconstructed translational profile

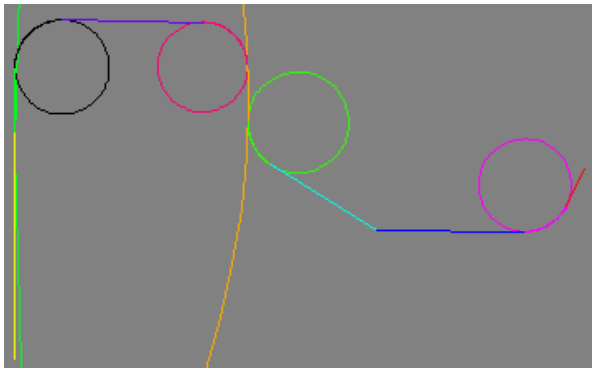

Figure 8. Circles of the rotational profile

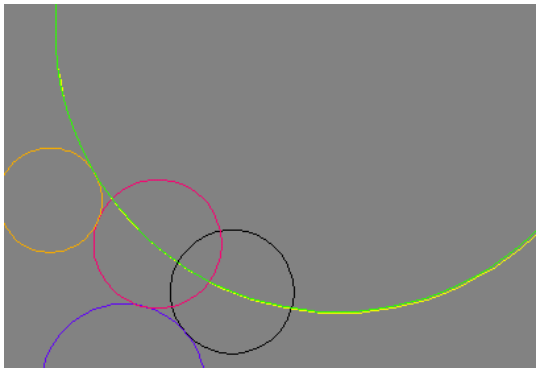

Figure 9. Circles of the translational profile

the points and satisfies the tangency constraints for all elements at once.

The equation of a circle written in Pratt's form is the following:

$$A(x^2 + y^2) + Bx + Cy + D = 0,$$

where $B^2+C^2-4AD = 1$. With the given normalisation condition, the expression on the left hand side is an approximation of the signed geometric distance of the point (x, y) from the circle, and consequently it is appropriate for a least squares minimisation. Taken in itself, this corresponds to a generalised eigenvector problem. As one can see, if $A = 0$, we obtain the equation of a straight line. The constraint expresses that the normal vector to the curve is a unit vector.

For constrained fitting we need equations which express tangency between two circles or between a circle and a line. These are, for two circles:

$$(A_1B_2 - A_2B_1)^2 + (A_1C_2 - A_2C_1)^2 - (A_2 \pm A_1)^2 = 0;$$

for a line and a circle:

$$B_lB_c + C_lC_c - 2D_lA_c \pm 1 = 0.$$

Based on these, a large system is set up, which minimises the square of the distances of the data points from the profile elements while at the same time satisfying a set of constraint equations. One may apply the general method of Lagrangian multipliers, or an alternative method of constrained fitting, as described in detail in a companion paper [4]. This solution requires an iterative process starting

from initial values. These are computed during segmentation of the guiding polygon by independent unconstrained fits to each profile element.

Returning to the benchmark object, Figure 6 shows the segmented profile curve of the LEFT component. In Figure 7 the profile of RIGHT is shown. The construction elements, i.e. the full circles generating this profile, can be seen in Figures 8 and 9. Having the profile curves, we can create the exact surface elements for each multiple face, which can directly be used in the final model building phase.

5. Building the boundary representation model

In order to build the B-rep model we have to compute all the edges and vertices explicitly. As discussed earlier, direct segmentation has produced a set of disjoint regions. Each simple region has already been approximated by a single analytic surface. The smooth, multiple regions have been further segmented and approximated using the constrained fitting method, which has produced not only a set of analytic surfaces, but smooth internal curves as well, representing common boundaries between pairs of adjacent surface elements.

Although our two-step segmentation procedure looks complex, it was formulated to avoid the potential difficulties in computing edge curves. Surface-surface intersections work well to produce sharp edges whose existence was deduced during the first planarity filtering phase. Surface-surface intersections are likely to fail for smooth edges, where the adjacent surfaces are in principle tangential, due to inaccurate measured data and possible numerical inaccuracies of the surface fitting procedure. Nevertheless, smooth internal curves can be easily computed for translational and rotational surfaces based on the vertices of the profile curve and the type of sweeping — extrusion or revolution. The internal curves within smooth, multiple regions are by-products of the constrained fitting procedure, and are explicitly available together with their points of intersection.

Let us return to our simple object shown in Figure 4 to describe the procedure of creating B-rep models. Of course, this description cannot contain all the details, but it is sufficient to introduce the basic concepts and most important steps of the algorithm.

5.1 Fattening

Model building starts by creating a *region adjacency* data structure, which fundamentally reflects the topology of the final B-rep model. In the segmentation phase, triangles which were not unambiguously classified were discarded. Now, we reclassify these triangles and remove the gaps between the disjoint regions. This phase is called *fattening*, in which regions are extended outwards until all triangles have been classified. In one pass, those unclassified triangles are considered which have at least one common edge with the classified triangles. Take one of these triangles. If it has only one classified edge, the distance of the opposite vertex from the surface to which the edge belongs is computed. If it is within tolerance, the triangle is added to the region. If the triangle has more than one classified edge, the distances between the related surfaces and the opposite vertices are computed, and the closest region (assuming it is within tolerance) will acquire the triangle. If there are no further triangles which can be added because they satisfy the tolerance criterion, we must continue based only on the adjacency of the triangles. In this pass all regions are simultaneously grown in a reasonably even manner.

By means of fattening we have practically created the topology of the object. Edges of the triangulation which are shared by two adjacent regions form polylines, from which we can deduce the existence

of edges in the final model. Vertices of the triangulation which are shared by at least three regions and are located at the ends of the polylines correspond to vertices in the final model. Connected sets of triangles which logically all belong to the same region correspond to faces.

Special treatment is needed in the case of 'very short' edges and 'very small' faces, which presumably have arisen due to inaccurate determination of vertices with more than three edges. For example, take vertex $V1$ in Figure 4. The common intersection point of the ideal surfaces $S1$, $S2$, $S3$ and $S4$ is quite likely to appear in the region adjacency structure in a different way. Due to numerical errors, fattening may detect $S2$ and $S4$, or $S1$ and $S3$ as neighbouring pairs, leading to the topologies shown in Figures 4(b) and 4(c). However, the true topology must be consistent with the surface geometries, and this may differ from the configuration determined by fattening. This issue will be addressed later.

Another kind of special treatment is needed if the model is incomplete. Take, for example, edges $e5$ and $e6$ in Figure 4, where the back faces adjacent to these edges are supposed to be missing, or the partial reconstruction of an object based on a single view in Figure 13. While typically, region boundaries are shared by two regions, in these cases there are no neighbouring triangles on the other side and *silhouette edges* must be generated to complete the model.

5.2 Edges and vertices

Computing edge curves is simple. In the case of a sharp edge, such as $e34$, we just call a surface-surface intersection routine to compute it, while if we have a smooth edge, such as $e7$, we just deduce the curve from the results of the constrained fitting phase, which guarantees that the smooth edge curve lies on both surfaces. For efficiency reasons, and to avoid ambiguities, we compute a bounding box for each edge, based on the related approximating polyline. We always leave processing of very short and silhouette edges to as late as possible as they are ill-defined.

Vertices where three surfaces meet are well determined. These can be determined by curve-surface intersection. For example, to compute $V2$, one option is to take $e34$, which is the common intersection curve of surfaces $S3$ and $S4$, and intersect it with surface $SF1$. If one of the edges is smooth, for example at vertex $V3$, special care is needed: here the smooth edge $e7$ shared by $TF2$ and $TF3$ must be intersected with the remaining third surface $S5$. Completely smooth vertices such as $V4$ are either determined during constrained fitting, or if necessary, by curve-curve intersection. Silhouette vertices, such as $V5$, are computed only approximately, constrained by the underlying edge curve $e8$ in our case.

If there have been no constraints enforced during surface fitting, vertices with more than three edges need special care. Take the previous example of vertex $V1$. While the fattening procedure may tell us that the topology is as shown in Figure 4(b), upon intersecting the various faces concerned, the geometry may produce the result shown in Figure 4(c). We must be careful to take note of this and modify the topology accordingly. For example, moving towards the critical area on $e12$, one has to intersect edge $e12$ with $S3$ and $S4$; if the intersection point with $S4$ occurs 'earlier', we have configuration (b) and we insert a small edge $e24$, otherwise configuration (c) is correct and we have to insert edge $e13$. Of course, for complex edge configurations there are many more cases which may arise, but the principle of ordering the intersections helps to establish the correct topology. At a later stage, one of the basic tasks of beautification is to discover these short edges and explicitly enforce coincidence.

At the edge of the triangulation (if any), we must terminate the model by silhouette edges. In order to complete the edge loops of the corresponding faces, we must compute approximate edges by tracing the border polylines of the triangulation, for example, along edges $e5$ or $e6$. In the general case, we

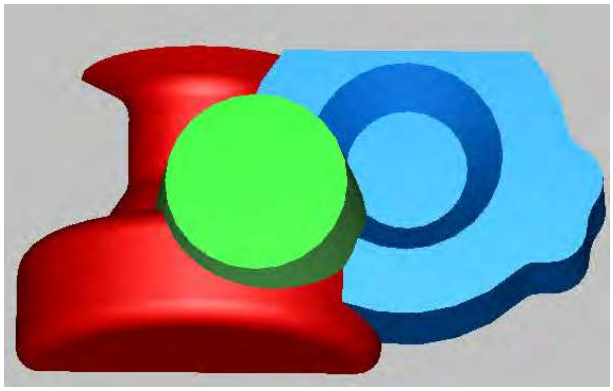

Figure 10. Test object without small blends

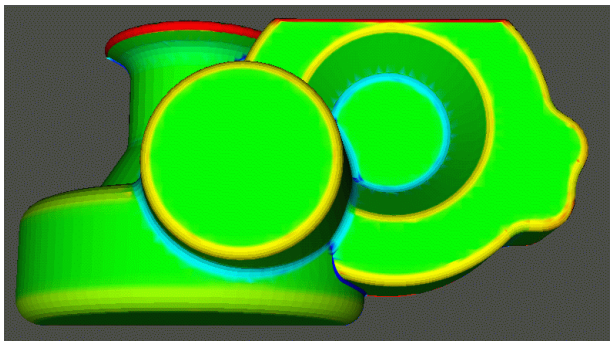

Figure 11. Test object with small blends

generate a piecewise linear approximation to the polyline and project it unto the underlying surface. Thus a sequence of small edges lying on this face will be obtained. If the face involved is a plane, an attempt is made to reconstruct the silhouette edge as a better composite curve made up of straight and circular segments, using the same methods as for profile reconstruction.

5.3 Stitching

We must now create the complete topological structure by stitching together the faces, edges and vertices. This is quite a straightforward process, since in the previous phases the consistency of the geometrical and topological entities has been assured. Taking an edge loop of a given face, the procedure applied guarantees that each real edge is shared by another edge of a neighbouring face and the related end vertices are identical. Thus, taking all edges of the loops of a given face, all adjacent faces can be stitched together. This operation is supported by most solid modelling kernels.

6. Blending surfaces

In the early planarity filtering phase we temporarily removed data points which belong to blending surfaces. Now, having already reconstructed the primary surfaces, we can reconstruct the blends. It is assumed that these were generated by constant-radius rolling-ball blending. We must determine the best approximation to the appropriate radius and just forward this information to the underlying solid mod-

eller which will explicitly construct an appropriate blend and incorporate it into the B-rep model. Based on the region neighbourhood graph and the triangulation, we ignore point regions around vertices, and take the disjoint point sets which represent single edge blends between two given primary faces. Note that in this context the primary face can possibly be a smooth, multiple face: this makes no difference from the point of view of blending.

There are different approaches to estimating blend radii; these are thoroughly analysed in [27]. *Iterative spine* methods are based on the principle that the spine curve of a rolling ball blend is computed by intersecting offsetted primary surfaces. Having an initial radius estimate, the spine and the corresponding blend can be computed. After calculating the distances of the data points from this estimated blend, the radius can be corrected and a new spine computed; this is done iteratively until all the points are within tolerance. This method is particularly efficient in special cases when the intersection of the primary surfaces is a simple, analytic curve, but it also works for surfaces of arbitrary type.

Another approach is to compute a so-called *maximum ball* for each data point such that the ball contains the point and touches the two adjacent primary surfaces. Having the radius estimate for the maximum ball at each point, we compute an average for the whole blend. This method is also efficient and numerically stable, and it has the advantage that it can be generalised to reproduce varying radius rolling ball blends.

We attach blend information to the edges, and let the modelling kernel incorporate the edge blends into the B-rep model. We do not reconstruct vertex blends in the current development phase, since in the case of small blends we only have a few data points. Vertex blends are considered to be by-products which automatically connect edge blends and are constructed by the solid modeller. Setback vertex blends provide a reasonable transition surface in the case of arbitrary edge blend configurations [39].

Figure 10 shows the reconstructed benchmark object with sharp edges. The blended object can be seen in Figure 11.

7. Implementation and experimental details

A prototype system for solid model reconstruction is under development at the Geometric Modelling Laboratory of the Computer and Automation Research Institute. A REPLICA laser scanner system (3D Scanners, London, UK) is used for data acquisition. The program is written in C++ and uses the VTK graphics library; it runs under LINUX and Windows NT. The ACIS solid modelling kernel [11] is used for solid model creation and blending.

The original simulated data set for our benchmark test object shown in Figure 1, contained 18970 points. The size of the bounding box was $100 \times 50 \times 40$. The noise added to the sampled data points had standard deviation 0.05 units. In Figure 1 the point set was decimated to 8238 points with 16432 triangles. The estimated rotational axes and the translational direction were (0.0028, 0.9999, -0.0008) and (0.0232, 0.0052, 0.9997); compare these with the exact vectors of (0, 1, 0) and (0, 0, 1). The radius of the large cylinder of LEFT was computed as 30.1478 (cf. 30); that of RIGHT was computed as 26.876 (cf. 27). The radii of the profile elements were reconstructed with reasonable accuracy. For example, for the rotational profile, the computed radii are 4.02 (cf. 4), 3.84 (cf. 4), 4.31 (cf. 4), 3.97 (cf. 4). The blend reconstruction also gave reasonable values, for example, for the top faces of MIDDLE and RIGHT we obtained radii of 2.03 (cf. 2) and 2.01 (cf. 2); along their intersection their intersection the radius was 1.01 (cf. 1).

To illustrate the difficulties of dealing with real measured data another object was selected; the seg-

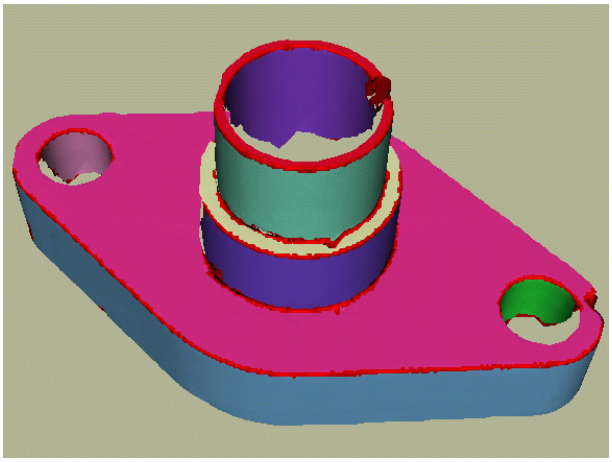

Figure 12. Segmentation of a single view point cloud

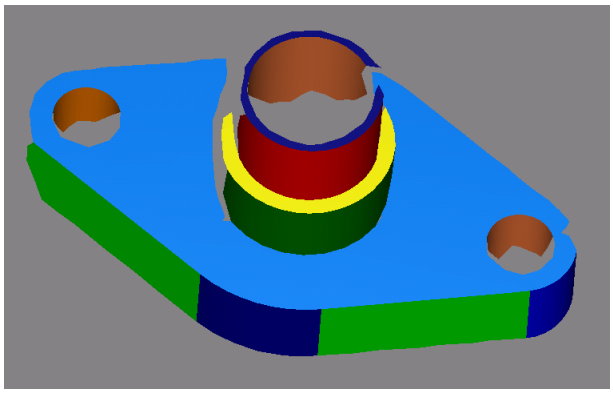

Figure 13. Solid object reconstructed from the single view

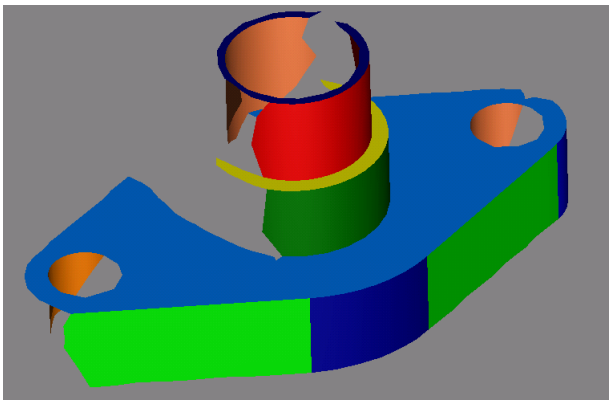

Figure 14. Another view of the reconstructed solid object

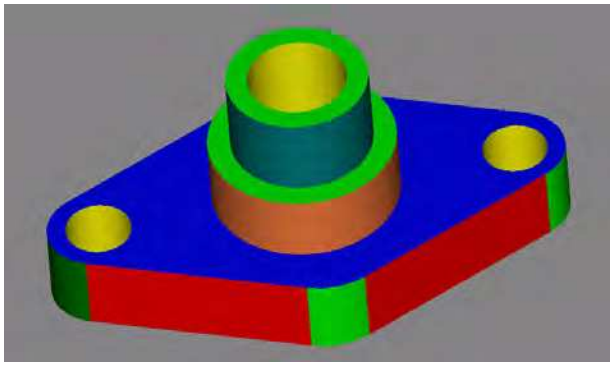

Figure 15. Solid object reconstructed from multiple views

mentation of the corresponding data is shown in Figure 12. This is a single view data set, and so the stitched model is incomplete. Compare figures viewed from the scanning direction (Figure 13) and another slightly rotated view (Figure 14). The latter may look horrible, but it is a valid ACIS model. Of course, if we start with a suitable multiple view point cloud representing the complete solid, the artifacts disappear, as shown in Figure 15.

8. Conclusion

A methodology for reverse engineering conventional solid objects has been presented, producing boundary representation solid models, which can be directly used in mechanical engineering CAD/CAM. The complex nature of the procedure was illustrated by describing the consecutive algorithmic steps and their links. The importance of reproducing the regularity of the objects — simple surfaces, translational

and rotational symmetries, constraints — cannot be overemphasised. The current prototype implementation shows that for reasonably dense and accurate measured data, solid reconstruction is possible with minimal user interaction by setting only certain threshold and tolerance values which control the various steps of the algorithm. Current and future efforts will be mainly directed towards reliably segmenting complex multiple regions, automatically deriving constraints, constrained fitting and model beautification.

9. Acknowledgements

The solid reconstruction project started in 1995 within the framework of an EU supported COPERNICUS grant (RECCAD 1068). The authors highly appreciate the contribution of the GML research team, including Géza Kós, Gábor Lukács, Péter Salvi and László Andor from CADMUS Ltd.. This research was also supported by the National Science Foundation of the Hungarian Academy of Sciences (OTKA, No. 26203) and by the UK EPSRC (Grant GR/M 78267).

References

[1] L. Alboul and R. van Damme, "Polyhedral metrics in surface reconstruction: tight triangulations", In: *The Mathematics of Surfaces VII*, Eds: T. Goodman, R. R. Martin, Information Geometers Ltd, 1997, pp. 309–336

[2] C. Bajaj, F. Bernardini, J. Chen and D. Schikore, "Automatic reconstruction of 3D CAD models", In: *Proc. Int. Conf. on Theory and Practice of Geometric Modeling II*, Eds: W. Strasser, R. Klein, R. Rau, Blaubeuren, Germany, October 1996

[3] P. Benkő, G. Lukács, R. R. Martin and T. Várady, "Algorithms to create B-rep models from measured point clouds", *Geometric Modelling Studies*, GML1998/6, Computer and Automation Research Institute, Budapest, 1998

[4] P. Benkő, L. Andor, G. Kós, R. R. Martin and T. Várady, "Constrained fitting in reverse engineering", submitted to *Computer Aided Geometric Design*, 2000

[5] R. Bergevin, M. Soucy, H. Gagnon and D. Laurendeau, "Towards a general multi-view registration technique", *IEEE PAMI*, Vol 18, No 5, 1996, pp. 540–547

[6] P. J. Besl, *Surfaces in Range Image Understanding*, Springer Verlag, 1988

[7] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting", *IEEE PAMI*, Vol 10, No 2, 1988, pp. 167–192

[8] R. M. Bolle and B. C. Vemuri, "On three-dimensional surface reconstruction methods", *IEEE PAMI*, Vol 13, No 1, 1991, pp. 1–13

[9] X. Chen and C. M. Hoffmann, "On editability of feature-based design", *Computer Aided Design*, Vol 27, No 12, 1995, pp. 905–914

[10] P. N. Chivate and A. G. Jablokow, "Solid-model generation from measured point data", *Computer Aided Design*, Vol 25, No 9, 1992, pp. 587–600

[11] J. Corney: *3D Modeling using the ACIS Kernel and Toolkit*, John Wiley and Sons, Chichester, 1997

[12] C. Dorai, G. Wang, A. K. Jain and C. Mercer, "Registration and Integration of Multiple Object Views for 3D Model Construction", *IEEE PAMI*, Vol 20, No 1, 1998, pp. 83–89

[13] M. Eck and H. Hoppe, "Automatic reconstruction of B-spline surfaces of arbitrary topological type", *Computer Graphics* (SIGGRAPH 96), Vol 30, 1996

[14] H. Edelsbrunner and E. Mücke, "Three-dimensional alpha shapes", *ACM TOG*, Vol 13, No 1, 1994, pp. 43–72

[15] O. D. Faugeras and M. Hebert, "The representation, recognition and locating of 3D objects", *Int. J. Robotics Research*, Vol 5, No 3, 1986, pp. 27–52

[16] A. W. Fitzgibbon, D. W. Eggert and R. B. Fisher, "High-level CAD model acquisition from range images", *Computer-Aided Design*, Vol 29, No 4, 1997, pp. 321–330

[17] G. Greiner and K. Hormann, "Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines", In: *Surface Fitting and Multiresolution Methods*, Eds: A. Le Méhauté, C. Rabut and L. L. Schumaker, Vanderbilt University Press, 1997, pp. 163–172

[18] B. Guo, J. Menon and B. Willette, "Surface reconstruction using alpha shapes", *Computer Graphics Forum*, Vol. 16, No. 4, 1997, pp. 177–190

[19] M. Hebert and J. Ponce, "A new method for segmenting 3D scenes into primitives", *Proc. 6th Intl. Conf. on Pattern Recognition*, (München), 1982, pp. 836–838

[20] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms", *Course Notes, Course 25, SIGGRAPH 97*, 1997

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

[22] A. Hoover et al., "An experimental comparison of range image segmentation algorithms", *IEEE PAMI*, Vol 18, No 7, 1996, pp. 673–689

[23] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganised points", *Computer Graphics* (SIGGRAPH 92), Vol 26, 1992, pp. 71–78

[24] *Reverse Engineering*, Eds.: J. Hoschek and W. Dankwort, B. G. Teubner, Stuttgart, 1996

[25] *Advanced Course on FAIRSHAPE*, Eds.: J. Hoschek and P. Kaklis, B. G. Teubner, Stuttgart, 1996

[26] J. Hoschek, U. Dietz and W. Wilke, "A geometric concept of reverse engineering of shape: approximation and feature lines", In: *Mathematical Methods for Curves and Surfaces II*, Eds: M. Dæhlen, T. Lyche and L. L. Schumaker, Vanderbilt University Press, 1998, pp. 253–262

[27] G. Kós, R. R. Martin and T. Várady, "Recovery of blend surfaces in reverse engineering", *Computer Aided Geometric Design*, Vol 17, 2000, pp. 127–160

[28] G. Kós, "An algorithm to triangulate surfaces in 3D using unorganised point clouds", *Geometric Modelling Studies*, GML1999/3, Computer and Automation Research Institute, Budapest, 1999 (to appear in Computing, 2000)

[29] I. K. Lee, "Curve reconstruction from unorganised points", *Technical Report No. 55*, Institut für Geometrie, Technische Universität Wien, 1998

[30] P. Krsek, T. Pajdla, V. Hlaváč and R. R. Martin, "Range image registration driven by a hierarchy of surface differential features", *Proc. 22nd Workshop of the Austrian Association for Pattern Recognition*, May 1998, pp. 175–183

[31] A. Leonardis, A. Jaklič and F. Solina, "Superquadrics for segmenting and modeling range data", *IEEE PAMI* Vol 19, No 11, 1997, pp. 1289–1295

[32] G. Lukács, A. D. Marshall and R. R. Martin, "Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation", In: *Computer Vision–ECCV 98*, Lecture Notes in Computer Science, Eds: H. Burkhardt and B. Neumann, Springer-Verlag, 1998

[33] A. D. Marshall, G. Lukács, R. R. Martin, "Robust segmentation of primitives from range data in the presence of geometric degeneracy", submitted to *IEEE PAMI*, 2000

[34] B. Parvin and G. Medioni, "B-rep from unregistered multiple range images", *Proc. Int. Conf. Robotics and Automation*, Nice, 1992, pp. 1602–1607

[35] V. Pratt, "Direct least-squares fitting of algebraic surfaces", *Computer Graphics* (SIGGRAPH 87), Vol 21, No 4, 1987, pp. 145–152

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

[37] N. S. Sapidis and P. J. Besl, "Direct construction of polynomial surfaces from dense range images through region growing", *ACM TOG*, Vol 14, No 2, 1995, pp. 171–200

[38] T. Várady, R. R. Martin and J. Cox, "Reverse engineering of geometric models — an introduction", *Computer-Aided Design*, Vol 29, No 4, 1997, pp. 255–268

[39] T. Várady and C. M. Hoffmann, "Vertex blending: problems and solutions", In: *Mathematical Methods for Curves and Surfaces II*, Eds: M. Dæhlen, T. Lyche and L. L. Schumaker, Vanderbilt University Press, 1998, pp. 501–528

[40] T. Várady, P. Benkő and G. Kós, "Reverse engineering regular objects: simple segmentation and surface fitting procedures", *Int. J. of Shape Modeling*, Vol 4 (1998), pp. 127–141

[41] N. Werghi, R. Fisher, C. Robertson and A. Ashbrook, "Object reconstruction by incorporating geometric constraints in reverse engineering", *Computer-Aided Design*, Vol 31, 1999, pp. 363–399