Recognizing Geometric Patterns for Beautification of Reconstructed Solid Models

F. C. Langbein   B. I. Mills   A. D. Marshall   R. R. Martin

Department of Computer Science, Cardiff University
PO Box 916, 5 The Parade, Cardiff, CF24 3XF, UK

Abstract

Boundary representation models reconstructed from 3D range data suffer from various inaccuracies caused by noise in the data and the model building software. The quality of such models can be improved in a beautification step, which finds regular geometric patterns approximately present in the model and imposes a maximal consistent subset of constraints deduced from these patterns on the model. This paper presents analysis methods seeking geometric patterns defined by similarities. Their specific types are derived from a part survey estimating the frequencies of the patterns in simple mechanical components. The methods seek clusters of similar objects which describe properties of faces, loops, edges and vertices, try to find special values representing the clusters, and seek approximate symmetries of the model. Experiments show that the patterns detected appear to be suitable for the subsequent beautification steps.

1. Introduction

Reverse engineering a physical object is the extraction of information from the object that is sufficient for a particular purpose like reproduction or redesign. Depending on the application, different representations of the object are needed. For an overview see [13]. We are interested in reconstructing a boundary representation (B-rep) model of a particular engineering part from 3D range data, which has all the desired geometric properties present in the original, ideal design. Our ultimate goal is an automated, intelligent 3D scanning system suitable for naive users and non-engineering applications as well as engineers.

We intend to reconstruct models with accurate geometric properties for engineering parts with only planar, spherical, cylindrical, conical and toroidal surfaces that either intersect at sharp edges or are connected by fixed radius rolling ball blends. Valid B-rep models approximating these objects can be generated by current reverse engineering systems [3]. However, the generated models suffer from various inaccuracies created by sensing errors from the data acquisition phase as well as approximation and numerical errors arising from the reconstruction process. Improving the precision of the sensing techniques and the reconstruction methods could reduce the errors, but some errors will always remain. As our intention is to recreate an ideal model for a physical object, we also have to consider additional errors introduced by possible wear of the object and the particular manufacturing method used to make it. To ensure that certain regular geometric patterns, like aligned cylinder axes or orthogonal planes, are present, they have to be enforced at some stage of the reverse engineering process.

Previous approaches augment the surface fitting step by constraint solving methods [14] such that, for instance, two planes are fitted simultaneously under the constraint that they are orthogonal. Another approach is to identify features like slots and pockets whose approximate location and type is provided by a human and use this information to improve the results of the segmentation and surface fitting phase [12].

In our approach we attempt to improve the B-rep model initially created by the model building software in a separate step which we call *beautification*. Improving the model without further reference to the point data avoids the computational expense of constrained fitting. An analyser is employed to find a set of regular geometric patterns as constraints that are approximately present in the initially created model. We aim to find a large set of possible constraints, which may contain inconsistencies. A maximal consistent subset of likely constraints has to be enforced by subsequent beautification steps to create an ideal model.

In this paper we discuss analysis methods to find common regular geometric patterns. We introduce the notion of similarity as the fundamental concept for the patterns and outline the results of a part survey identifying common patterns. Then we present methods to find similarities related to parameters, edge loops, directions, axes, positions and surface types and a method to find approximate symmetries of point sets derived from the model. Finally we discuss the results of some experiments.

2. Geometric patterns as similarities

We use similarity as a concept to recognize certain geometric patterns that are approximately present in a B-rep model. From a *global* point of view this leads to approximate symmetries, which are discussed in Section 9, as similarity between subsets of the model or between isometric images of the model. In the following sections we focus on *local* similarities between properties of B-rep model elements like faces, loops, edges and vertices, and similarity between a property present in the model and a predefined special value for this property. For instance, we find a set of cylinders with similar radii, and detect if the average value is close to a special value such as an integer.

We extract properties of B-rep model elements or groups of these elements as typed *feature objects* and detect similarities between them. The *type* of a feature object is defined by the property it represents. For instance, the axis of a cone or cylinder forms a feature object of the type *axis*. Note that a single B-rep element can generate several different feature objects of various types. Furthermore, additional feature objects can be created from groups of feature objects or groups containing feature objects and B-rep model elements. For example, axes may generate intersection points as further feature objects.

Similarities between elements of a set of feature objects $X = \{x_l\}$ of the same type are described by a similarity measure $\delta : X \times X \to \mathbb{R}_0^+$ which depends on the type, where δ is symmetric, non-negative, and $x_1 = x_2$ implies $\delta(x_1, x_2) = 0$. We call two objects $x_1, x_2 \in X$ ε-similar if $\delta(x_1, x_2) < \varepsilon$, $(\varepsilon \in \mathbb{R}^+)$.

To find similar feature objects of the same type we use a hierarchical clustering algorithm. The hierarchical structure can be used in the subsequent steps to decide which constraints should be accepted for the ideal model. Note that for clustering the similarity measure should be a metric. We want to create a partition $\{C_k\}$ of X where each C_k is represented by a feature object c_k and a tolerance t_k such that $\delta(c_k, x_l) < t_k \leq t_a$ for all $x_l \in C_k$, where t_a is a tolerance value limiting the maximum size of the clusters and depends on the feature object type. Additionally we require C_k to be maximal, i.e. there is no $x_l \in X \setminus C_k$ that is t_a-similar to c_k. Each C_k is partitioned further into sub-clusters S_j in the same way as X. The sub-clusters have to be sufficiently distinct from each other, i.e. for each pair $S_{j_1}, S_{j_2}, (j_1 \neq j_2)$, the condition $\delta(s_{j_1}, s_{j_2}) - t_{j_1} - t_{j_2} \geq t_d$ has to be fulfilled for some tolerance t_d with $0 < t_d \leq t_a$. The sub-clusters are again partitioned if we can find further subsets fulfilling a similar condition.

The feature object representing a cluster is computed by an averaging method `avg` depending on the feature object type. Given two ε-similar feature objects x_1, x_2 of the same type representing clusters with ω_1 and ω_2 elements, the average $x_{\mathrm{avg}} = \mathtt{avg}(x_1, \omega_1, x_2, \omega_2)$ represents the union of the two clusters such that $\delta(x_{\mathrm{avg}}, x_l) < \varepsilon$, $(l = 1, 2)$.

A variety of solutions to the clustering problem exists [8, 16]. The most common approach is to start with the smallest value of $\delta(c_l, c_k)$ and combine the two elements with respect to the hierarchical structure to give a new element $\hat{c}$ which replaces c_l and c_k. This is repeated until only one element remains or the distance between the elements is too large. The brute force algorithm for this can be improved by using a dynamic closest pair data structure which partially sorts the distance values such that the clustering can be done in $O(n^2)$ time and space [7].

3. Part survey and common patterns

We surveyed about 600 mechanical components to determine common geometric patterns which are suitable for beautification [11]. Various objects like small engine parts, fittings and brackets for optical systems, plastic fittings, caps and connectors, sliding fittings for cupboards and a general selection of CAD display models from online repositories and company catalogs, and other part surveys were reviewed. The parts chosen had low to medium complexity, i.e. less than about 200 faces and their geometric properties were significant for their application. They were also physically small enough to be put on a typical 3D scanner, which means that they must fit inside a 50cm cube, and be light enough to man-handle onto the scanner bed. The features were large enough (bigger than about 5mm) to provide sufficient data to be able to properly fit surfaces, and there were no deep cavities that could not be probed by the scanner.

About 97% of the parts exhibited important geometric patterns which could be classified using our similarity concept. This justifies our approach trying to exploit such patterns to improve the quality of reconstructed models. We list common patterns for which we present analysis methods in Table 1. The number in the last column indicates how common the particular geometric pattern is from 5 for being nearly always present to 1 for rare as determined manually.

Our algorithm looks for similar parameters from surfaces and edges such as lengths and angles, and tries to find special values for those parameters, like integers. Special values might also depend on manufacturing and functional purposes. We do not consider the former for the ideal model. If a value is not a simple rational number and depends on a functional purpose, this value is usually subject to a tolerance, which should be larger than the scanner tolerance $(\sim 20\mu\mathrm{m})$. If we choose a value within that tolerance, the model should be usable. For applications where this is not enough, specific methods have to be developed.

Further, we present a method to find similar edge loops independent of a scaling factor. We also consider directions, like plane normals and axis directions. We seek par-

Parameters	Equal parameters.	5
	Special parameter values.	3
Loops	Equal loops independent of scaling.	4
Directions	Parallel directions.	5
	Directions which have the same angle relative to a special direction.	4
	Symmetrical arrangements of directions on planes.	3
	Symmetrical arrangements of directions on cones.	4
Axes	Aligned axes.	3
	Axes intersecting in a point.	3
Positions	Equal positions.	2
	Equal positions under projection.	3
Surface Types	Surface is approximately a plane or a cylinder.	—
Complete Model	Finite symmetry groups of special points.	4

Table 1. B-rep model elements with related geometric patterns and their estimated frequency.
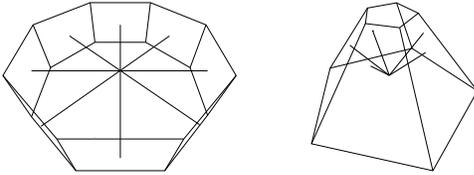
Figure 1. Symmetrically arranged directions.

allel directions and look for special directions with respect to which all directions in a set have the same angle forming a plane or a cone. The directions in these sets might also be arranged symmetrically (see Figure 1).

Associating directions with positions leads to axes. The positions may be obtained from vertices, apices of cones, and centres of spheres and tori, etc. We look for aligned axes and their common intersection points. For positions alone we seek equal positions and positions which are equal when projected onto a special plane or line derived from the main directions in the model.

We also consider changing surface types. For example, a large radius cylindrical face may be similar to a planar face and thus be classified incorrectly as planar. As this is a problem of the model building software and not a regular pattern of a real part, no frequency has been determined.

The majority of interesting global aspects turn out to be strongly related to cubes, or at least to rectangular prisms. The major symmetries involved are prismatic and pyramidal of orders 2, 4 and 6, all of which can be produced by decorating cubes. The only common exceptions are conic and cylindrical symmetry. We extract special points like vertices, surface centres, etc. and determine the finite symmetry groups of these point sets in an approximate sense. This means we look for isometries mapping the original set approximately onto itself.

4. Parameters and simple fractions

Parameters listed in Table 2(a) describe aspects of B-rep model elements independently of their location and orientation. To avoid comparing obviously unrelated parameters like the semi-angle of a cone with the radius of a cylinder, a type is assigned to each parameter (Table 2(b)). We cluster parameters of the same type using the hierarchical clustering algorithm. Since the values for angle and length parameters are on different scales, the clustering tolerances depend on the parameter type. We use t_{la} and t_{ld} for lengths and t_{aa} and t_{ad} for angles.

The resulting hierarchical clusters of similar parameters of the same type are represented by average parameter values which should perhaps be replaced by special parameter values. In the following we present a method to find simple fractions close to a given parameter value.

Let w be a real number for which we search simple fractions in the open interval $(w - t, w + t)$ with the tolerance t. We assume that integer values are always special for w and that w is non-negative. Note that recognizing numerical constants [2] is a well studied related problem, but assumes that a rather exact approximation is required.

If $1/(2t) < m$, then more than one n/m for a fixed m could be in the interval $(w - t, w + t)$. To avoid this ambiguity, we set the maximum allowed value for m to $M_0 = \text{floor}(1/(2t))$, which should not be larger than some M_{max}, say 10. Also note that if t is larger than $1/2$, we would actually ignore the integers. Hence, we set a minimum M_{min} for M_0, which can optionally be larger than 1 if we have to work with large t.

If M_0 is small, we could simply multiply w by each $m \leq M_0$ and check if $|w - n/m| < t$ with $n = \text{round}(wm)$ to find special fractions. However, for large M_0, this becomes expensive as we have to check many fractions for a large

Element	(a) Parameter	(b) Type	(c) Direction
plane	—	—	normal
sphere	radius	length	—
cylinder	radius	length	axis direction
cone	semi-angle	angle	axis direction
torus	major radius	length	axis direction
	minor radius	length	
straight edge	distance between end points	length	direction of the line between the end-points
circular edge	radius	length	normal of plane in which
	angle of the circle segment	angle	the circle lies
elliptical edge	—	—	normal of plane in which the ellipse lies

Table 2. Parameters and directions.

set of real numbers w. We give a more efficient method for larger M_0 by combining the simple method with continued fractions in Algorithm 1. We start by approximating w by $a_0 + x_0$ where $a_0 = \text{round}(w)$. The error x_0 is approximated by using the simple method recursively. We call the method initially by $\texttt{rec\_frac}(x_0, a_0, 1, (a_0 > w), M_0)$. Starting with $b_l = 1$ in recursion level l, we get $x_l = b_l/a_l + x_{l+1}$ for $a_l = \text{round}(b_l/x_l)$ and error x_{l+1}. Additional approximations are obtained by increasing b_l by one, until a_l is larger than some limit M. The process is repeated recursively for each error x_{l+1} until the error is smaller than some tolerance t_{rmin}. Note that as functions of the type $b/x_0 - \text{round}(b/x_0)$ are more likely to generate small denominators for $x_0 \in (0.5, 1)$, we start the algorithm with $1 - x_0$ if $x_0 < 0.5$. Table 3 shows an example where the special values generated are marked with an arrow.

To apply this method to an average parameter v of a parameter cluster with a tolerance t_a (t_{la} or t_{aa}), we have to map the range of v on an appropriate range to get w. We choose a family of functions $f_l : \mathbb{R} \to \mathbb{R}$ which represents the scales on which we look for simple fractions. For angle parameters we use the functions $f_\pi : v \mapsto v\pi$ and $f_t : v \mapsto \arctan(v)$. For length parameters the family is defined by $f_{K_l} : v \mapsto vK_l$ where the K_l are base units for length measurements like 1.0, 0.1 or 2.54 (cm to inch conversion). We then apply the algorithm to $w_l = f_l^{-1}(v)$ for each f_l within a tolerance t. For linear $f_l : v \mapsto vK_l$ we use the tolerance $t = t_a/K_l$. For f_t, we use t_{aa} as tolerance with the condition $||v| - \pi/2| > t_{\text{aa}}$ assuming that $v \in [-\pi, \pi]$.

5. Similar edge loops

We seek approximately equal edge loops in the B-rep model using the hierarchical clustering algorithm. Our current method only considers flat loops with straight edges. A more sophisticated system handling all loops as polygons formed from the vertices but considering the geometries of the edges could be built on this system. For each loop we compute the first n Fourier coefficients of a function f representing the polygon as defined below [4], which gives us a complex vector u of Fourier coefficients with d components u_j. We compare two of these vectors u, v with the similarity measure $\delta(u, v) = \sum_{j=1}^{d} ||u_j| - |v_j||$. The averaging method is the usual weighted average of complex vectors. To cluster the vectors we require two tolerances t_{loa} and t_{lod}. This method is an efficient and simple way to compare polygons and provides a natural similarity measure. Note that by using the Fourier transform of f, we compare the loops independently of scale.

There are various ways to define a function f representing a polygon. One basic idea is to define a curvature of a polygon. Let α_k be the angle at the k-th vertex v_k of the polygon with m vertices v_k such that $v_0 = v_m$ and let l_k be the length of the line segments from v_0 to v_k. In our particular approach we use a *distribution* f of the form $f = \sum_{k=0}^{m-1} \alpha_k \delta_k$. δ_k is the Dirac distribution at $2\pi l_k/l_m$ such that $\langle \delta_k, \phi \rangle = \phi(2\pi l_k/l_m)$ for $\phi \in \mathbf{C}^\infty(\mathbb{T})$ where $\mathbf{C}^\infty(\mathbb{T})$ is the space of infinitely differentiable functions on the unit circle $\mathbb{T}$ in the complex plane identified with the infinitely differentiable, 2π periodic, complex functions on $\mathbb{R}$. For details see [5]. The Fourier coefficient of order j, ($j \in \mathbb{Z}$), of the distribution f is

$$u_j = \frac{1}{2\pi} \langle f, \exp(-ij\cdot) \rangle = \frac{1}{2\pi} \sum_{k=0}^{m-1} \alpha_k \exp\left(-ij2\pi \frac{l_k}{l_m}\right).$$

(1)

Let p be the minimum of $(l_{k+1} - l_k)/l_m$, i.e. the smallest ratio between an edge length and the circumference of the polygon. If P is the minimum of the p of all relevant polygons, then at least $\text{ceil}(1/P)$ Fourier coefficients have

I. The function has been called as `rec_frac(x,n,m,neg_frac,M)` where x is the value which has to be approximated by a fraction, n, m are the two integers representing the fraction n/m found so far, `neg_frac` indicates if x has to be added or subtracted from n/m and M is the maximum denominator allowed at this recursion level.

II. Let $b = 1$.

III. While the denominator $a = \mathrm{round}\,(b/x)$ is not larger than M:

    1. If $a > b$:

        A. Let $r = x - b/a$.

        B. If r is negative, set `neg_r` to `true` and $r = |r|$. Otherwise `neg_r` is `false`.

        C. If `neg_frac` is `true`, then the new numerator is $p = na - mb$, otherwise $p = na + mb$.

        D. The new denominator is $q = ma$.

        E. If $r < t$, then add p/q to the list of special values, if it is not already in it.

        F. If $r > t_{\mathrm{rmin}}$ and p/q was not already in the list of special values, call `rec_frac(r,p,q,neg_r,`M_0M`)`.

    2. Let $b = b + 1$.

Algorithm 1. Recursive algorithm `rec_frac` for finding simple fractions.

$$
\begin{array}{llll}
0.59 & = 1/2 & + & \underbrace{0.09} \\
 & & = 1/11 & [\to \mathbf{13/22}] \quad - \quad 0.000909 \\
 & & = 2/22 & \phantom{[\to \mathbf{13/22}]} \quad - \quad 0.000909 \\
 & = 2/3 & - & \underbrace{0.076667} \\
 & & = 1/13 & [\to \mathbf{23/39}] \quad - \quad 0.000256 \\
 & = 3/5 & [\to \mathbf{3/5}] \quad - \quad 0.01 &
\end{array}
$$

Table 3. Finding special values for 0.59 with $\mathbf{t} = 0.05$, $\mathbf{M_0} = 5$ and $\mathbf{t_{rmin}} = 0.01$.

to be computed to ensure that all relevant frequencies are considered. As the Fourier coefficients of f can be computed exactly, only a small number of coefficients is sufficient to characterize the shape of the polygon. Typically we use the first 10 coefficients.

For an n-sided, regular polygon, the only non-zero coefficients are those of order kn, ($k \in \mathbb{Z}$). In the approximate case the values of these coefficients are sufficiently larger than the others, so that they can be used to identify special polygons. It appears that this can also be used to find polygons which are based on an n-sided, regular polygon, with minor modifications like an additional or missing vertex.

6. Regularly arranged directions

We consider regular patterns of directions, like normals of planes and directions of cylinder axes, arising from the B-rep model elements (see Table 2(c)), as unit vectors and identify opposite directions. This direction space is the real projective plane P_2, which can be represented by a sphere with antipodal points identified.

Directions on a great circle of the sphere represent directions that are orthogonal to another direction and thus lie in a plane, which we call an *axis plane*. Directions that are on a small circle of the sphere represent directions on a cone, which we call an *axis cone*. The arrangement of the directions in a plane or a cone may also be symmetric. We call these *planar* or *conical angle-regular* (see Figure 1).

Directions that are approximately parallel correspond to approximately equal points in P_2. They can be detected using the hierarchical clustering algorithm. The small angle between two directions d_1 and d_2 defined by $\angle(d_1, d_2) = \arccos(|d_1{}^t d_2|)$ is a suitable similarity measure for clustering. The average of two directions d_1 and d_2 representing clusters of the size ω_1 and ω_2 is $\left(\omega_1 d_1 + \mathrm{sign}(d_1{}^t d_2)\omega_2 d_2\right) / \left(\omega_1 + \omega_2\right)$. The two tolerance values used for clustering are t_{aa} and t_{ad} defined earlier. Clustering the directions as points in P_2 yields a hierarchical structure of parallel directions. Even if the number of directions extracted is large, we expect to find only a limited number of *different* directions.

6.1. Directions in planes and cones

A set of directions $\{d_l\}$ on a circle in P_2 satisfies the equation system $|d_l{}^t x| = a$, where $x \in P_2$ is the centre of the circle. If $a = 0$, we have an axis plane with normal x. If $a \in (0, 1)$, we have an axis cone with axis x. Note that for a plane, taking the absolute value of $d_l{}^t x$ is not required, and we can drop it for a cone if we ensure that all directions have the same orientation relative to x.

To find the sets of directions d_l lying in an axis plane,

I. Compute the angles α_{lk} between the directions d_l and d_k for $l < k < m$, where m is the number of directions d_l.

II. For all reference directions d_{j_0} with $j_0 < m$ and for all angles α_{lj_0} with $j_0 < l < m$:

    1. Find all candidate base angles $\beta_{j_0 n_0} = \pi/n_0$ for $n_0 \in \mathbb{N}$ such that $|\beta_{j_0 n_0} - \alpha_{lj_0}| < t_{aa}$. For each candidate base angle do:

        a. Add $\beta_{j_0 n_0}$ to the list of candidates for d_{j_0} if there is no integer multiple $\beta_{j_0 n}$ of $\beta_{j_0 n_0}$ already in the list.

        b. If $\beta_{j_0 n_0}$ has been added to the list, remove any $\beta_{j_0 n}$ from the list which is an integer multiple of $\beta_{j_0 n_0}$.

III. For each reference direction d_{j_0} with $j_0 < m - 1$, consider the subset $\{d_{j_0}, \ldots, d_m\}$ and for each candidate $\beta_{j_0 n}$ with $n = 1, \ldots, N_{j_0}$:

    1. For each d_j with $j_0 < j \leq m$:

        A. If for $f = \alpha_{lj_0}/\beta_{j_0 n}$, we have $|\operatorname{round}(f)\beta_{j_0 n} - \alpha_{lj_0}| < t_a$, then do:

            a. Record d_j to be a $\operatorname{round}(f)$ multiple of the base angle. It is stored as a multiple $\operatorname{round}(f)$ under the following conditions.

            b. If there is already a direction D as the multiple $\operatorname{round}(f)$ and D is the parent of d_j, then only replace D by d_j if d_j is sufficiently closer to the multiple as indicated by t_{pd}.

            c. If there is already a direction D as the multiple $\operatorname{round}(f)$ and d_j is the parent of D, then only replace D by d_j if D is not sufficiently closer to the multiple as indicated by t_{pd}.

            d. If there is already a direction D as the multiple $\operatorname{round}(f)$ and d_j is not related to it in the hierarchical structure, then replace D by d_j if d_j is closer to the multiple.

            e. If there is no direction D as the multiple $\operatorname{round}(f)$, then make d_j this direction.

    2. If the arrangement in the angle-regular subset for $\beta_{j_0 n}$ is regular (see text), note it. In addition remove integer multiples of $\beta_{j_0 n}$ from the base angle candidate lists for the reference directions d_l which are in the current regular subset.

Algorithm 2. Algorithm to find angle regularities.

we cluster the normals representing all axis planes generated from each pair of linearly independent directions. The clustering is done in the same way as the clustering of parallel directions, but we employ the clustered parallel directions instead of *all* directions. This will not only reduce the number of normals generated, but also avoids cases where the directions are approximately linearly dependent.

A similar method is used to find axis cones. For each triple d_1, d_2, d_3 of linearly independent directions representing parallel direction clusters, we generate an axis cone with axis in direction c and angle α by solving the linear equation system $d_l{}^t x = 1$, ($l = 1, 2, 3$). From this we get the cone parameters as $\alpha = \arccos(|x^t d_1|/\|x\|_2)$ and $c = \alpha x$. As the d_l are not exact, we avoid finding flat axis cones which are actually axis planes, or axis cones which represent parallel lines, by rejecting cones for which $\alpha < t_{aa}$ or $|\pi/2 - \alpha| < t_{aa}$.

The axis cones represented by pairs (c_l, α_l) are clustered. For two cones (c_l, α_l), ($l = 1, 2$), the similarity measure for clustering is defined by $\sqrt{\angle(c_1, c_2)^2 + (\alpha_1 - \alpha_2)^2}$. The averaging method for two cones generates an average cone with the weighted average of the two directions as axis and the weighted average of the angles as angle. The tolerances used are t_{aa} and t_{ad} defined earlier.

Once we know the set of directions forming a regular arrangement, the quality of the approximation can be improved by solving the equation system $|d_l{}^t x| = a$ in a least squares sense for the direction set.

6.2. Symmetrically arranged directions

Exploring axis planes and cones in more detail may reveal symmetries in the arrangements of the directions. Given a set of directions in a plane or a cone, we look for subsets such that the angles between the directions in a subset are integer multiples of a base angle β. The subsets may be incomplete in the sense that not all multiples of β are present. The main problem is to identify appropriate subsets of multiples that approximately satisfy this condition.

Let $\{d_j\}$ be a set of directions in an axis plane and let α_{lk} be the angle between d_l and d_k. We call the directions d_j (approximately) *planar angle-regular* if there is an angle $\beta = \pi/n$, ($n \in \mathbb{N}$), such that there is an $\alpha \in \{\alpha_{lk}\}$ with $|\alpha - \beta| < t_{aa}$ and the α_{lk} are (approximate) integer multiples of β. Based on which multiples of β are present we decide whether an angle-regular set represents a regular pattern. To avoid too small base angles β, we set a maximum N_{max} for n, where N_{max} should be smaller than $\pi/(2t_{aa})$.

Given a set of m directions d_j, we look for a minimum number of subsets which are approximately planar angle-regular. The algorithm for this consists of three main steps (see Algorithm 2). First we generate all angles α_{lk} ($l < k < m$) between the directions d_l and d_k where a column k in that matrix with elements below the diagonal represents the angles to d_k used as a reference direction for the set $\{d_k, \ldots, d_m\}$. We have to take care to choose angles consistently to the right of the reference direction d_k

6

with respect to the normal q of the axis plane (see Figure 2). With $v = q \times d_k$ we have

$$\alpha_{lk} = \begin{cases} \arccos(d_k{}^t d_l) & \text{if } v^t d_l \geq 0, \\ \pi - \arccos(d_k{}^t d_l) & \text{if } v^t d_l < 0. \end{cases} \quad (2)$$

This also allows us to identify which of the $k\pi/n$, $k = 0, \ldots, n-1$, directions is occupied by a particular direction, for some $n \in \mathbb{N}$.

In the second step a set of possible base angles $\beta_{j_0 n}$ for each reference direction d_{j_0} is derived from the α_{lk}. As we look for approximate angle-regular arrangements, a single $\alpha_{l j_0}$ can generate more than one candidate β depending on the tolerances. Base angle candidates which are multiples of a smaller base angle candidate are eliminated.

In the third step we try to find planar angle-regular subsets by checking the angles $\alpha_{l j_0}$ for each reference object d_{j_0} and all base angle candidates $\beta_{j_0 n}$. Before we accept an angle-regular subset of directions as regular we also check which multiples of $\beta_{j_0 n}$ are actually present. We accept it if either all multiples, or at least every second one for $n > 4$ is present, or at least three consecutive directions are occupied.

Finding conical angle-regular sets is very similar to finding planar angle-regular sets. Before we can employ Algorithm 2, the directions on the cone have to be projected onto a plane orthogonal to the cone axis. Due to this projection, opposite directions in the plane cannot be considered equal anymore and the base angles are $2\pi/n$. Note that an orthogonal system is a conical angle regularity.

7. Axes and positions

In this section we discuss regular geometric patterns related to axes and general positions in the B-rep model. For directions associated with a position, i.e. which represent axes, we seek approximate intersections of axes, and aligned axes. For cylinders, cones and tori we use the root point of the surface as the associated position. The root point of a cone is its apex and the root point of a torus is its centre. The root point of a cylinder is an arbitrary point on the central axis. For elliptical edges we choose the centre of the ellipse and for straight edges we choose an arbitrary point on the edge as the associated position.

Planar faces do not have an obvious root point, but we can define such a root point by considering the boundary loops. Suitable root points are the average of the vertex positions for each loop and the centre of the convex hull of each loop. Note that this defines multiple axes for a single planar face. Other possibilities exist for defining root points of planar surfaces and more general types of edges.

To seek intersecting axes, we compute the minimum distance between them. If it is smaller than t_{la}, we say the axes
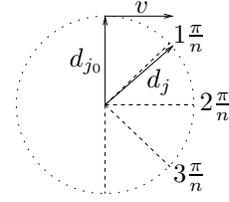
Figure 2. Planar angle-regular.

intersect and use the midpoint of the shortest line between them as the approximate intersection point. These intersection points are clustered to find sets of axes intersecting approximately in a point. For this we use actual axes and not direction clusters. We only intersect axes that belong to different direction clusters to avoid trying to intersect approximately parallel axes.

To detect approximately equal positions, we cluster positions of vertices, root points of surfaces and axis intersection points with tolerances t_{la} and t_{ld}, the Euclidean distance as similarity measure, and the weighted average between points. In addition, we project the points on special planes and lines through the origin to find partially equal positions. The planes and lines are determined by the main directions present in the model. If the model has one or more orthogonal systems, we use planes orthogonal to the directions in these systems and lines defined by the orthogonal directions. If the model has a main axis without an orthogonal system, we use the line through the origin parallel to the main axis and the plane orthogonal to this direction. Partially equal positions are found by clustering the projected points.

8. Surface types

For some surfaces, the surface fitting software might not have found the correct geometric type. For example, a cone with a small angle could be interpreted as a cylinder.

To determine if a surface geometry might need to be changed, we have to check whether the piece of surface present in the model could be well approximated by another surface type. To derive a measure for this we use the curvature of the surface, which is also used for determining the surface type in our model building software [10]. This measure is only valid if the piece of the surface is sufficiently small. For planar, cylindrical, spherical and toroidal surfaces the principal curvatures k_1, k_2 are constant. If $\sqrt{k_1^2 + k_2^2}$ is approximately zero as indicated by some tolerance t_c, it is approximately a plane. If only one k_i is approximately zero, the surface is approximately a cylinder with radius $1/k_j$, where k_j is the second curvature.

We have to take special care for a cone, as one of the principal curvatures is not constant. Let $k_2(p)$ be the non-

constant principal curvature at a point p on a cone. Conical faces for which the surface type can be changed do not contain the apex and they are always finite, so k_2 has a minimum and maximum. If $k_2(p_{\min}) - k_2(p_{\max})$ is approximately zero with respect to t_c, then the cone is approximately a cylinder or a plane. It is a approximately a plane if $K = (k_2(p_{\min}) + k_2(p_{\max}))/2$ is approximately zero, otherwise it is approximately a cylinder with radius $1/K$.

Note that a cylinder might also be approximately a cone, or a plane might be approximately a sphere, etc., but for such cases, we do not have any indication of parameter values to use for the alternative surface. By setting up the surface fitting software such that spheres, cones and tori are preferred, these cases become less likely.

However, it is not enough to check only the curvature to decide if a surface could be of a different type. It has to be possible to change the surface within the topology of the model. For this we check if all the boundary loops of the surface are also approximately on the alternative surface.

9. Detecting finite symmetry groups

Finite symmetry groups formed by isometries mapping the model onto itself are also symmetries of certain point sets derived from the model. A finite model with an *infinite* number of symmetries has the symmetries of a sphere, a cylinder or a cone. For the type of models we consider, these symmetries can be detected as equal and partially equal positions, aligned axes and orthogonal directions. Here we outline a method to find approximate *finite* symmetry groups of a point set. We can detect approximate symmetries of the model with this by extracting appropriate points like vertices, centres of spheres, cones and tori from the model, and verifying if the symmetries found also preserve the type of each point, the adjacency, and the orientation information of the model.

See [1, 9, 15] for related approaches to detect approximate symmetries. The difficult cases in previous work are related to problems in determining the existence of precisely the symmetry being looked for. Our method seeks maximal symmetry and the associated tolerance without predetermined choice of tolerances or symmetries.

In the first step of our algorithm all consistent clusterings of the point sets are generated. In a second step each of the clusterings is examined for symmetries.

We start by taking each point as a separate cluster with a tolerance level of 0. All point pairs are stored in a priority queue based on distance between the points. In order of increasing distance, point pairs are consecutively removed from the queue until the queue is empty. When removing a pair from the queue, the two clusters to which the points belong are combined into a single cluster. The distance between the points represents the current tolerance level. Whenever combining two clusters yields a consistent clustering, we check for symmetries of the centroids of the clusters in the second step. We call a clustering *consistent* if the distances between the remaining points on the queue are larger than the current tolerance level, and for each cluster, all distances between the points in the cluster have been considered and removed from the queue.

In the second step we aim to determine symmetries of the central points of each cluster. The set of symmetries of these points is determined as permutations that preserve the distances between them within their tolerance level. First we compute a non-degenerate tetrahedron whose vertices are the centroid of the point set, and three points on its convex hull, chosen such that they are as far apart as possible from the centroid and the other selected points. Instead of an expensive check for all possible permutations representing the isometries, we do a limited depth-first search over a tree of permutations.

The tree of permutations is formed by *partial injections*. A partial injection is a list of point pairs where each point appears at most once as first and at most once as second element of the pair. A pair indicates that the partial injection maps the first point on the second point. A partial injection is a permutation if each point is present exactly once as first and exactly once as second element. The root of the permutation tree is an empty list as partial injection. The children of a partial injection are those obtained by appending one more pair to the list of pairs. The leaves of this tree form the permutations.

A partial injection is *approximately distance preserving* if, for each pair of points in the domain of the partial injection, the absolute difference between the distance of the two points and the distance between their images under the partial injection is less than the tolerance. Only approximately distance preserving partial injections, which we call *proto-symmetries*, are candidates for valid permutations representing symmetries. Their number is typically much smaller than the complete tree of partial injections.

First a depth-first search to depth level four is performed on the permutation tree, backtracking whenever a node is not a proto-symmetry. Once a partial injection maps three points in general position in a distance preserving fashion, the fourth point can only be mapped to two possible locations and all subsequent points can only be mapped to one. Using the centroid as the first point means that only two points need to be mapped, since the centroid must be mapped to itself. Thus, we only check partial injections involving the three points of the tetrahedron on the convex hull for being a proto-symmetry. This applies to zero tolerance levels and non-zero tolerance levels if the point set is not too elongated. In order to find the unique match for the rest of the points, each of the points is tried in turn for each match in a direct approach.
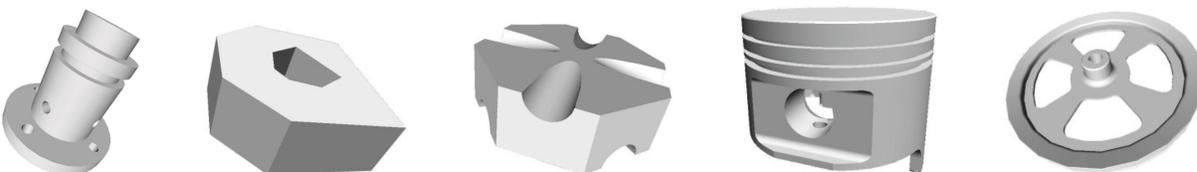
	(a)	(b)	(c)	(d)	(e)
P:	21	55	78	79	93
U:	1	8	9	5	18
M:	0	3	7	0	5

Figure 3. Test objects.

If the tolerance is non-zero, then the approximate distance to each of the first four points does not always identify a point uniquely. This problem occurs when the distance between two of the four points is small compared to the distance from those points to the point being identified. To avoid this problem the first four points are chosen to be the centroid and three points on the convex hull as far from the centroid and each other as possible. In this way no point in the collection is further from the first four than they are from each other.

By scanning only a depth of four points, a long collection of points is seen as an approximately linear arrangement. Such a point set can be modified by magnifying the distance of each point radially from the central line. As this does not affect rotational or mirror symmetries referred to that line, the radially magnified point set is then analysed for approximate symmetry in a second pass.

The algorithm has been tested on point sets. Experiments with approximately symmetric point sets show that the algorithm runs in a few minutes for examples of the complexity arising in reverse engineering.

10. Experiments

The methods presented in this paper have been tested with various engineering objects. By rotating and translating faces of exact models and generating a point cloud from these perturbed objects we simulated noisy reverse engineering data. From the point cloud we used reverse engineering software provided by CADMUS to obtain reconstructed models, which were then analysed by our algorithms implemented on a GNU/Linux platform.

Some of our test objects are shown in Figure 3 ((d) and (e) were obtained from [6]). They were all perturbed by 3 degrees and 0.3 length units. For (d) only the exterior visible to a 3D scanner was considered. As the symmetry detection algorithm has only been implemented for point sets, it was not part of these tests. The tolerance values for testing the methods were set to 5 degrees and 0.5 length

units. **P** is the total number of patterns found. **U** is the number of unwanted patterns detected, where an unwanted pattern is one that is not part of the design and conflicts with the intended patterns. **M** is the number of important patterns missed as identified by a human.

Analysing (a) resulted in a number of equal cylinder radii with special values. As some of the radii in the exact model were only 0.5 length units apart, they were unwantedly considered to be equal. The sets of axes formed by the main cylinders and the holes in the bottom were detected to be parallel. The axes of the opposite holes in the sides of one cylinder were also found to be parallel and to be all in a plane. The planar angle-regular algorithm detected their symmetrical arrangement. The conical angle-regular algorithm detected the orthogonal system created by the main axes and the axes of the side holes. Using the information about parallel axes, all the aligned axes were detected as well. Further, the intersection points of the axes of the side holes and the axes of these holes with the bottom holes were found. Projecting the positions onto the plane defined by the main axis revealed a number of partially equal positions.

Similarly most of the regular geometric patterns in the other objects were detected. This includes the planar angle-regular direction sets in (b) and (c) and the two conical angle-regular sets of cylinder axes in (c). However, some of the planar faces building the two angle-regular arrangements in (b) were unwantedly considered to be approximately parallel, as the difference between the angles was within the angular tolerance. In (c) this caused the detection of some unwanted conical angle-regular directions. In (b) and (c) the intersection point of the axes generated from the centre of the convex hulls of the side faces was not detected correctly, as the perturbation of the side faces moved some of the axes too far apart.

The main axes of (d) and (e) were found as parallel directions and aligned axes. Some parameter values of object (d) were unwantedly considered to be equal. In object (e) the algorithms detected the two planar angle regularities of the faces orthogonal to the main axis with the base angle $\pi/2$

9

and the angle $\pi/6$ between them. Furthermore, a number of close, but different, positions were considered to be equal.

The experiments showed that by choosing small tolerance values, only a few very accurate and thus also very likely patterns are found and some patterns are missed. By increasing the tolerance values more patterns are detected, but at the same time the number of unwanted patterns increases. While the tolerance values could be determined by carefully analysing the errors introduced by the reverse engineering process, the algorithms will still find unwanted patterns if the exact model contains these approximately.

Further experiments showed that the number of unwanted patterns could be reduced by careful tolerance value selection. At present, however, we aim to generate a large number of possible patterns and leave the problem of choosing a set of consistent constraints, which determine the original design, to the subsequent beautification steps.

11. Conclusion

We have presented methods to find regular geometric patterns in inaccurately reconstructed solid models based on similarities. Tests with various perturbed objects were satisfactory in the sense that important geometric patterns were found in the parts and appear to be suitable for the subsequent beautification steps.

In future work we will develop a system which tries to find a maximal, consistent set of constraints containing the main patterns to generate an ideal model.

Acknowledgements

References

[1] H. Alt, K. Mehlhorn, H. Wagner, and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237-256, 1988.

[2] D. H. Bailey and S. Plouffe. Recognizing numerical constants. In *Proc. Organic Mathematics Workshop*. Simon Fraser University, December 1995.

[3] P. Benkő, R. R. Martin, and T. Várady. Algorithms for reverse engineering boundary representation models. To appear in *Computer-Aided Design*, 2001.

[4] S. Berchtold, D. A. Keim, and H.-P. Kriegel. Using extended feature objects for partial similarity retrieval. *The VLDB Journal*, 6:333-348, 1997.

[5] R. Dautray and J.-L. Lions. *Mathematical Analysis and Numerical Methods for Science and Technology, Functional and Variational Methods*, volume 2, pages 4-17. Springer, Berlin, Heidelberg, New York, 1988.

[6] Drexel University. National design repository. <uri: http://edge.mcs.drexel.edu/repository/>.

[7] D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. In *Proc. 9th ACM-SIAM Symp. Discrete Algorithms*, pages 619-628, January 1998.

[8] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[9] S. Iwanowski. Testing approximate symmetry in the plane is NP-hard. *Theoretical Computer Science*, 80:227-262, 1991.

[10] G. Lukács, R. R. Martin, and A. D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In H. Budkhadj and B. Neumann, editors, *Proc. 5th European Conf. Computer Vision*, volume 1, pages 671-686, Freiburg, Germany, June 1998.

[11] B. I. Mills, F. C. Langbein, A. D. Marshall, and R. R. Martin. Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components. Submitted to *Computer-Aided Design*, 2000.

[12] W. B. Thompson, J. C. Owen, J. de St. Germain, S. R. Stark, and T. C. Henderson. Feature-based reverse engineering of mechanical parts. *IEEE Trans. on Robotics and Automation*, 15(1):57-66, 1999.

[13] T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models – an introduction. *Computer-Aided Design*, 29(4):255-268, 1997.

[14] N. Werghi, R. Fisher, C. Robertson, and A. Ashbrook. Object reconstruction by incorporating geometric constraints in reverse engineering. *Computer-Aided Design*, 31(6):363-399, 1999.

[15] H. Zabrodski and D. Avnir. Measuring symmetry in structural chemistry. *Advances in Molecular Structure Research*, 1:1-31, 1995.

[16] J. Zupan. *Clustering of Large Data Sets*. Research Studies Press, Letchworth, Hertfordshire, England, 1982.