

Topological and Geometric Beautification of Reverse Engineered Geometric Models

F. C. Langbein¹, C. H. Gao^{1,2}, B. I. Mills^{1,3}, A. D. Marshall¹, R. R. Martin¹

¹School of Computer Science, Cardiff University, PO Box 916, Cardiff, CF24 3XF, UK

²School of Manufacturing Science and Engineering, Sichuan University, Chengdu, China

³Institute of Information and Mathematical Sciences, Massey University, Auckland, Albany, NZ

Abstract

Boundary representation models reverse engineered from 3D range data suffer from various inaccuracies caused by noise in the measured data and the model building software. Beautification aims to improve such models in a post-processing step solely working with the boundary representation model. The improved model should exhibit topological and geometric regularities representing the original, ideal design intent. This paper gives an overview of algorithms for a complete beautification system suitable for improving the topology and the geometry of low to medium complexity reverse engineered models.

Categories and Subject Descriptors (according to ACM CCS): J.6 [Computer-Aided Engineering]: Computer-Aided Design

1. Introduction

Reverse engineering extracts sufficient information from physical objects to reconstruct CAD models for a particular purpose like re-design, reproduction or quality control. Ideally, for applications like redesign, the reconstructed model should exhibit exactly the same geometric properties present in the original, ideal design. Rather than trying to create a description of the exact measured physical object, suitable for creating an identical copy or for inspection purposes, we are interested in reverse engineering the shape of an engineering object such that the description represents the original design intent. For this purpose we use a state-of-the-art reverse engineering system which can create a boundary representation (B-rep) model of the object's natural surfaces from dense 3D range data. However, due to inaccuracies in the measured data, approximation and numerical errors during the reconstruction process, and possible wear of the scanned object, the model is approximate in the sense that it only approximately exhibits intended regularities such as symmetries. We present a system to automatically improve such models in a post-processing step, which we call *beautification*.

[VM02] gives an overview of reverse engineering systems. Here we consider objects bounded only by planes, spheres, cylinders, cones and tori. Such faces may be connected by fixed-radius rolling ball blends. There are reliable surface fitting methods available for these surfaces [BMV01] and many realistic engineering objects can be described using only these surface types [MLM*01a]. In this paper we ignore blends. Fixed-radius rolling ball blends can be detected in the point data [KMV00], but rather than inserting the blends immediately, we treat them as edge and vertex attributes, and construct the blends after beautification.

Alternative approaches exist based on simultaneous surface fitting. Thompson et al. [TOG*99] consider feature-based reverse

engineering of mechanical parts. In their system a human identifies features like pockets in the point set interactively. This information is used to reconstruct the model by fitting parametric feature models instead of simple surfaces to the 3D point set, which improves the accuracy of the generated models. It is also possible to fit multiple surfaces to 3D point sets under geometric constraints [BKV*02, WFR*02]. Thus, rather than fitting surfaces individually, they are fitted simultaneously using the constraints as a set of conditions which the surface parameters have to fulfil in addition to providing a good fit to the 3D point data.

Such approaches often require human interaction because low-level information about an object's surface represented as a point set is not directly sufficient to make decisions about higher-level design intent. We try to avoid the necessity for human interaction by first extracting a higher-level B-rep model with analytic, natural surfaces. From this representation, further information about the actual design intent can be derived automatically. By trying to improve the B-rep model without further reference to the point data we also significantly reduce the computing time.

In the following we give an overview of our complete beautification system, and outline the main algorithms for topological and geometric beautification. As experiments have shown, our system is able to improve models of low to medium complexity.

2. Beautification

Beautification has to address defects in the *topology* as well as adjusting the *geometry*. E.g., if we reverse engineer a four-sided pyramid, we expect all four sloping faces to meet at a single vertex at the top. Equally important is that the sloping faces are arranged symmetrically to form a regular pyramid. We address both types

of problem. We assume that the input model is valid, although it may not have a closed shell if there are gaps in the point set. Note that we aim to change the model by a relatively small amount—just enough to impose approximate geometric and topological regularities on the model which are present within a small tolerance.

Firstly, we detect *topological defects* and decide how to adjust the topology to remedy them. Doing so results in a list of geometric elements and geometric constraints describing the geometric relations between the elements as required by the topology. If the constraint system is solvable, at least one object with the specified topology exists. Note that usually there are many more, and we must still further specify the geometry. For this we seek potential approximate *geometric regularities* in the second step. Typically, a large number are found, not all of which need be mutually consistent. Thus, a consistent subset of these regularities, likely to represent the original design intent and describe the complete improved model, is selected. During selection, we have to maintain the *solvability* of the constraint system describing the model while also selecting *likely* regularities. Finally, the constraint system is solved and a new model is rebuilt. Thus, overall our beautification system executes the following main steps in sequence:

- I. **Detecting topological defects:** small faces, sliver faces, short edges, gaps in the model, etc. are identified.
- II. **Adjusting the topology:** isolated small faces and short edges are replaced by a single vertex and surrounding topology is adjusted to meet it; existing faces are extended to cover gaps left by missing data by removing the edges and loops bounding gaps; etc. The realisability of these changes is tested by verifying the solvability of a constraint system.
- III. **Detecting approximate geometric regularities:** symmetric arrangements of faces, other regular arrangements, etc. which are approximately present in the geometry are detected. Exact conditions for approximate regularities are used rather than arbitrary tolerances. The methods aim to detect sufficient regularities to determine the improved model.
- IV. **Selecting geometric regularities:** a consistent set of geometric regularities likely to describe the model's design is selected. To do this, regularities are expressed by constraints. Methods are used to determine the solvability of constraint systems, and the likelihood of a regularity being part of the ideal design.
- V. **Rebuilding an improved model:** an improved model is rebuilt from the solution to the constraint system. (Checks must be made for remaining topological defects; these must either be repaired immediately or the process restarted at step II.)

Using this approach, the topological changes desired may not be geometrically realisable. While we check for realisability using a constraint system, we cannot include general constraints, e.g. requiring two surfaces to intersect, without further specifying the relation, as this requires the use of inequality constraints. Thus, when adjusting the geometry, which includes finding exact relations between intersecting surfaces, we may detect that the topology cannot be realised. In this case we can *either* try to repair the model during rebuilding, e.g. fill holes with additional faces, *or* return to the topological beautification phase and choose an alternative topology.

For typical reverse engineered objects, topological defects are localised in the sense that interaction between multiple topological defects is limited to local faces rather than the global structure. This allows us to repair topological defects of different types in a well-

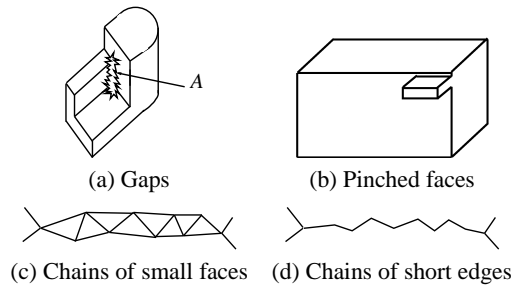


Figure 1: Some Problems for Topological Beautification

defined sequence, and limits the possible inconsistencies between topological and geometric adjustments.

3. Topological Beautification

We first detect and repair topological defects of the types listed below. They do not represent a list of all possible topological defects, only problems which are likely to arise during reverse engineering of models. Note that we start with a valid model (although it may be incomplete, i.e. it may have no closed shell) and these defects merely represent undesirable topology rather than incorrect topology. In our approach the problems are repaired in the order listed below—for details see [GLM*03a].

1. **Removing gaps in a single face:** A loop of half-edges may exist in the interior of a face, with nothing on the other side of the loop. Such cases may arise, e.g., where the scanner did not collect any data from within a deep concavity. The loop of half-edges should be removed, extending the face.
2. **Removing gaps crossing an edge:** A loop of half-edges may span two faces, with nothing on the other side of the loop. The edge between the faces is divided into two pieces by the gap. The gap should be removed by extending the faces and inserting a new edge section.
3. **Removing gaps spanning multiple faces:** A loop of half-edges may span multiple faces, with nothing on the other side of the loop. If the surrounding edges intersect approximately in one, two, or more vertices, the gap is respectively replaced by a vertex or an edge, or a new face is inserted to close the gap.
4. **Adjusting pinched faces:** If a face narrows to a very thin part it is *pinched* (see Figure 1(b)). The thinning is removed by joining the edges close to each other by a vertex or an edge sequence. This may split a face into two faces.
5. **Removing chains of small faces:** Two faces should intersect in an edge, but instead a chain of small faces may separate them. This chain should be replaced by an edge (see Figure 1(c), where the first step is to reduce a face chain to an edge chain).
6. **Removing sliver faces:** Two faces should meet in an edge, but instead a long very thin (*sliver*) face may separate them. The sliver face should be replaced by an edge. We only consider isolated sliver faces which can be replaced by a single edge.
7. **Removing chains of short edges:** Several consecutive short edges may need to be replaced by a single long edge (see Figure 1(d)). This problem may result, *inter alia*, from repairing some of the other problems listed.
8. **Merging adjacent faces with the same geometry:** Two adjacent faces may share the same geometry across a contiguous edge sequence. Edges and vertices as appropriate are removed,

and the faces merged.

9. **Removing isolated small faces:** Several edges should meet in a single vertex, but instead they meet at several distinct vertices, joined by multiple short edges surrounding a small face. The small face should be replaced by a vertex.
10. **Merging edges:** Once faces have been merged, related edges should also be merged. As each edge should be the complete, connected intersection of two adjacent faces, we merge each edge pair connected by a vertex which is attached to no more than two edges.
11. **Removing isolated short edges:** If several edges should meet at a single vertex, but instead they meet at several distinct vertices, joined by one or more short edges, these short edges should be replaced by a single vertex.

Topological beautification may involve the local addition or removal of faces, edges and vertices, and other updates to nearby topology to ensure that a correct, valid model results. We only add and remove elements and change the boundary relations. The geometry is found from the original geometry using only simple computations (e.g. computing an average position). For example, edges may need to be disconnected from an existing vertex, and connected to a new vertex. In addition, constraints must be generated and imposed on the geometry attached to the topological elements to ensure that in the final model, the geometry and topology are consistent (e.g. a vertex has to lie on an edge it bounds).

In a *raw* reverse engineered model, multiple topological defects of multiple types will usually coexist. To efficiently resolve these problems we need to detect and modify the defects in the right order. E.g. removing a gap spanning multiple faces may produce a sliver face, but removing a sliver face can never produce a gap. Given the particular ordering above, certain defects are known not to be present at each stage, having been repaired earlier. Thus, certain potential complex interactions between multiple defect types can be ignored.

All the defects listed depend on a notion of “small”, e.g. we intend to remove only “small” spurious faces as determined by a tolerance. This tolerance can be detected automatically by a consistent clustering method (see Section 4). It could also be provided by the user based on the magnitude of errors expected in the model. When choosing a tolerance, we should be careful that small but significant parts of the model are not deleted—the tolerance should clearly distinguish between the size of any small face or short edge which is to be deleted and any part of the model which is to be retained. For simplicity, we assume here that a single *global* tolerance value is used. However, if, for example, different regions of the object were scanned at different resolutions, a more sophisticated approach using an adaptive tolerance might be needed.

Topological beautification has some similarities to, but also some differences from, CAD model healing [PC03]. Healing also aims to adjust (and repair) topology, but for an input model with more general topological defects. Thus, healing may have to cope with physically impossible geometry, incorrectly oriented surfaces, faces with no defined geometry, self-intersecting edges, faces whose boundary is not a closed loop, and incomplete topology even though all individual faces are present. Such problems of *validity* are *not* expected to occur in beautification.

Detecting topological defects can in principle be done by a linear scan of the faces and/or edges in the model by testing simple condi-

tions which depend on the defect type. Note that this only applies to *local* topological defects, which are typical for reverse engineered models. Defects relating to the global structure, e.g. chains of small faces along *all* natural edges of the model, cannot be handled in this way. Such cases arise due to inadequate range data, and must be resolved by obtaining more accurate data. *Local* defects can be handled by making *local* changes to the topological structure.

4. Geometric Beautification

After topological beautification we have to make the geometry agree with the updated topology. Furthermore, we also wish to adjust the geometry so that the model exhibits exact intended geometric regularities which may only be approximately present in the raw model. We do this by imposing geometric constraints which both enforce the required topological structure and the desired geometry—the constraint system describes the complete, beautified model for rebuilding.

Our methodology for approximate geometric regularities is based on an exact notion of approximate symmetry. As we aim to detect many regularities, it is likely that they are not all mutually consistent. Hence, we have to select a set of consistent regularities which completely describes the likely design intent of the model. In our current approach we add regularities in *priority* order to the constraint system. A regularity is only *selected* and kept if the constraint system remains solvable.

Regularities are described in terms of symmetries of *features*. By features we mean properties of B-rep elements (vertices, edges, and faces, called *cells* in the following) which change in a similar way to the element itself under isometric transformations. We consider *positional* features such as vertex positions; *directional* features such as plane normals; *axis* features such as cylinder axes; *length* and *angle* features such as edge lengths and cone semi-angles. We require that relations between features are preserved when transformed by the same isometry. E.g. while isometries may change the directions of individual axes, isometries do not change the angle between directions.

Following Klein’s Erlanger Program, a geometric property of a cell is any property which remains invariant under isometric transformations. For a straight edge, its length is such a property. Consider two orthogonal planes. The $\pi/2$ angle between the plane normals does not change when we transform both planes by an isometry. However, as we intend to detect such arrangements for all direction features, we would have to generate features for each pair of cells with suitable directions. Instead, we choose to define features for individual cells. These are not invariant under isometries, but they change in the same way as the cell does when transformed by an isometry. Thus, relations between features are preserved, and we can use such features to determine regular arrangements between cells (e.g. symmetrically arranged directions where the angles between the directions are integer multiples of some angle π/n).

Table 1 lists the types of common regularities which we determine using feature symmetries. The regularities are mainly distinguished by the type of symmetry involved. Note that we also take into account special values (e.g. a length exactly equal to an integer) not related directly to symmetries.

The simplest regularity type is formed by cells with *identical* features, e.g. *parallel* directions—such features remain invariant under

Feature Type	Regularity	Symmetries
Direction	Parallel directions	Identity
	Symmetries of directions	Isometries
	Rotational symmetries of directions like in regular prisms and pyramids	Rotations
Axis	Aligned axes	Identity
	Parallel axes arranged equi-spaced along lines and grids	Translations
	Parallel axes arranged symmetrically on cylinders	Rotations
	Axes intersecting in a point	Identity
Position	Equal positions	Identity
	Point set symmetries	Isometries
	Equi-spaced positions arranged on a line or a grid	Translations
	Positions arranged symmetrically on a circle	Rotations
	Equal positions when projected on a special line or plane	Identity
Length /	Equal scalar parameters	Identity
Angle	Special scalar parameter values	(special value)
	Simple integer relations	(special value)

Table 1: Some Common Geometric Regularities

the *identity* transformation. More generally, a feature set which remains invariant under all isometries or a sub-group of isometries represents a regularity. Often, it is not the whole feature set which remains invariant, so we have to find appropriate maximal subsets. Thus, we distinguish between *global* and *partial* symmetries.

Let $F = \{f_1, \dots, f_n\}$ be a set of n mutually distinct features which remains invariant under a group G of isometries. The isometries are associated with permutations of the features. Each $g \in G$ induces a permutation σ of the labels $1, \dots, n$: for $g(f_k) = f_l$ we get $\sigma(k) = l$. Note that if the features are not mutually different, g induces more than one permutation. Hence, identity regularities are a special case, and must be detected first. We can then replace identical features by a single feature in order to detect non-trivial symmetries as permutations. For instance, for n points p_1, \dots, p_n arranged in sequence symmetrically around a circle in \mathbb{E}^2 , the permutation $\sigma: l \mapsto (l+1) \bmod n$ is induced by a $2\pi/n$ rotation around an appropriate point. By detecting all distance preserving permutations of the points we find these rotations.

For beautification, we require a concept of *approximate* geometric regularity. In the approximate case, features only match approximately, which yields ambiguous situations where global information is required to find a proper symmetry. We seek conditions such that local information is sufficient. In the exact case, symmetries relate to distance preserving permutations which can be detected by checking whether features match locally. We define approximate symmetries such that this behaviour is retained for the features in question. We detect approximately distance-preserving permutations at tolerance levels where a local match implies a global match as defined exactly in the following paragraph.

Let $s =_\epsilon t$ iff $|s - t| < \epsilon$ for $s, t \in F$ and let $D(F) = \{d(r_l, r_k) : l, k \in L\}$ for $L = \{1, \dots, n\}$. A permutation σ is an approximate symmetry of F at tolerance level ϵ , if $=_\epsilon$ is an equivalence relation on $D(F)$, and $|d(f_l, f_k) - d(f_{\sigma(l)}, f_{\sigma(k)})| < \epsilon$ for all $l, k \in L$. So we can look locally for approximately matching distances between

features, while $=_\epsilon$ being an equivalence on $D(F)$ ensures that we get a global match with respect to the elements of F .

Based on this general approach, regularity detection starts by *clustering* the features. The clusters have to be *consistent* in the sense that all distances between elements of a cluster are smaller than a tolerance and the distances to features in other clusters are larger than this tolerance. This also identifies appropriate tolerances automatically. We then seek approximately distance-preserving permutations of the clusters. As the permutations correspond to isometries we only have to check a small number of features: in 3D Euclidean space mapping a tetrahedron onto another tetrahedron completely determines an isometry. Thus, we only have to consider four points to find an isometry and check whether the remaining points are mapped onto each other by the isometry. For partial regularities, we also have to identify appropriate cluster subsets: considering all subsets is too expensive.

For more details on regularity detection see [Lan03]. Our symmetry concept is described in [ML03]. Various algorithms for detecting partial and global regularities are given in [GLM*03, LMM*01, LMM*01a, LMM*01b, MLM*01].

Regularity detection determines a large set of approximate regularities present in the raw model at various tolerance levels. Constraints can be used to impose these on the model. However, the constraints are unlikely to be mutually consistent. As we do not use a strict tolerance limit, we get some regularities which only exist at rather large tolerances. Moreover, the detection methods seek multiple relations between the same features. Hence, we have to select *mutually consistent* regularities, which are likely to represent the original, ideal design intent.

We use a sequential selection method which tries to add regularities in order of a *priority*. This is suitable as a means for improving low to medium complexity models. We build a constraint system in sequence by adding constraints describing the regularities. As we add each constraint, we check if the system remains solvable. If not, the regularity it describes is rejected and all corresponding constraints are removed from the constraint system. This way we select in preference regularities with high priorities, while at most considering each regularity once.

The priority of a regularity is computed by taking a weighted average of: a measure of the numerical accuracy to which the regularity's constraints are satisfied in the raw model, a figure of merit expressing the quality or desirability of the regularity depending on specific arrangements and constants involved, and a constant describing a minimum desirability for each regularity type. This average is weighted by a merit function which indicates how common the regularity is (determined by surveying a range of engineering components). A detailed description of the priority computation is given in [Lan03, LMM03]. We note that a more sophisticated decision process considering more complex relations between regularities and the model, globally, could improve regularity selection.

5. Solvability of Constraint Systems

We require an efficient solvability test for constraint systems, as solvability is tested many times during beautification. Interpreting constraints in a topological context leads to a method similar to the usual degrees-of-freedom analysis [HLS97] where the degrees of

freedom become the topological dimensions of the involved spaces. While it is desirable to ensure that we have a unique solution or at most a discrete set of solutions, we only check whether at least one solution exists, and for efficiency we do not compute a solution. In particular we accept cases where there are infinitely many solutions. We can seek some solution for the geometry of the *final* model close to that of the raw model. The large number of regularities we detect makes under-constrained systems very unlikely.

Our solvability test is explained in detail in [Lan03, LMM03]. The basic idea is illustrated by the following example of distance constraints between points. An unconstrained point in \mathbb{E}^3 can be anywhere in space, i.e. its parameter domain is \mathbb{R}^3 . A distance constraint between two points v_1, v_2 limits the allowed values the two points can have. One way of enforcing this is to parameterise v_1 by an arbitrary value in \mathbb{R}^3 and to require that v_2 is on a sphere of fixed radius with centre v_1 . Thus, v_2 is described by a parameter on the unit sphere \mathbb{S}^2 in combination with the position of v_1 , and so for v_2 we select a (lower-dimensional) sub-manifold of the parameter manifold \mathbb{R}^3 , which is essentially \mathbb{S}^2 . (Obviously, the roles of v_1 and v_2 can be exchanged.) When a 3D point is constrained by two such distance constraints, it has to lie in the intersection of two spheres. In general this intersection may be empty, a point, a circle or a sphere—without solving equation systems we cannot determine which. However, we may assume that usually we have the generic case of two spheres intersecting in a circle.

Dimension reduction as above does not directly give the solvability properties of a constraint system. For this we have to consider the remaining dimensions of the spaces involved. They cannot be reduced to less than 0. In fact, as we do not determine a global orientation or position of the elements involved, we have to have at least 6 dimensions left for each rigid sub-structure in the constraint system. This can be verified by considering the dependencies between the geometric elements as induced by the constraints.

6. Model Rebuilding

The regularity selection process results in a constraint system describing the model’s geometry. As the final beautification step we have to rebuild a beautified model from this constraint system.

Initially we find a solution to the constraint system using a numerical solver (a robust quasi-Newton method minimising least-squares error). Note that for beautification the geometric model may not always be described completely by our constraints (but it is highly likely as we detect many regularities), so a symbolic approach is unsuitable. Furthermore, not all constraint systems can be readily solved symbolically.

An improved model is rebuilt using the beautified topological information derived from the raw model with the geometric feature values obtained from the numerical solution. We create new faces and re-intersect them to obtain the complete model. Sharp edges are found using a standard surface-surface intersection algorithm. As we only consider certain analytic surface types, smooth intersection edges can be directly handled as special cases.

In our beautification system we make decisions about adjusting the *topology* before we adjust the *geometry* of the model. Because the topology is enforced by special constraints, the resulting constraint system cannot be inconsistent unless it is necessary to consider non-generic cases, which are unlikely for usual engineering

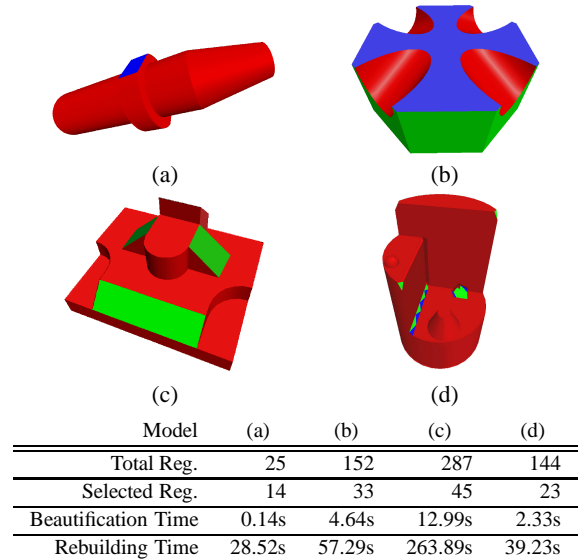


Figure 2: Beautification Examples

objects. In order to avoid selecting invalid topologies, we already check the topological constraints during topological beautification. However, note that we cannot use general constraints requiring two surfaces to simply intersect without specifying the desired relation.

A potential problem arising from separating topological and geometric beautification is that we do not consider whether the topological and geometric changes are consistent with respect to design intent. Topologically, we may decide to remove a small face, but this small face may be required in order to realise a complicated geometric regularity. If we remove the face first, then we cannot realise this regularity later. Such issues rarely arise in practice; combining topological and geometric decisions is left as future work.

7. Experiments and Discussion

The system presented here has been tested using a variety of low to medium complexity reverse engineered models. Our experiments show that it is able to improve raw models with respect to design intent. As the raw models are approximate, there is always some uncertainty about the actual design intent. Depending on the tolerance settings, specific parameter values and minor regularities are not always reconstructed according to the original design. However, major regularities, like global symmetries, major orthogonal systems, etc. representing the global structure of the model, are imposed exactly on the improved model.

Figure 2 shows some of our models used to test beautification. It lists the numbers of regularities detected, how many were selected for beautification, the times for detecting and selecting constraints, and the times for solving the constraint system and rebuilding the model. Further results are given in [GLM*03a, Lan03, LMM*01, LMM03].

Our methods are able to improve reverse engineered models, but are limited by the ambiguities caused by the fact that we only have approximate models. Major regularities of the model can be handled quite robustly and are usually exactly enforced in the improved model, but the minor regularities selected do not always represent

the intended design. Here, by major regularities we mean regularities which involve a large number of faces of the model and usually relate to a highly symmetric arrangement (with respect to the features). Minor regularities involve only a few faces in the model and usually have less symmetry.

As we have an approximate model, we have to work with tolerance levels. If the tolerance level is small enough that the features of interest are sufficiently distinct, we are able to identify them precisely. However, to beautify reverse engineered models, we usually have to work at tolerance levels where the features cannot be clearly distinguished in such a way. We try to reduce the ambiguities by looking for tolerance levels at which certain properties of the features are present unambiguously in a local sense. But when there are inconsistencies between these regularities, it is not always possible to make a clear decision between them, as there are always cases of inconsistencies between regularities which are all more-or-less equally desirable. This applies in particular to minor regularities, e.g. multiple special angle values between two planar faces. This is a fundamental property of approximate models, and while our methods were designed to take this into account, such ambiguities cannot be avoided.

Our system is able to detect approximate regularities for which clear, unambiguous evidence is present in the raw model. It reports the regularities at tolerance levels at which there is no ambiguous interpretation of the data. Most intended regularities are detected in the raw models. As no maximum tolerance value is used, and the tolerance levels for the regularities are detected automatically, differences in the tolerances of intended regularities can be handled. However, this also results in a larger number of regularities which have to be considered for selection. Trying to devise detection algorithms which only detect *intended* approximate regularities appears to be considerably harder. While we seek unambiguous evidence in the raw model for the presence of a regularity, we cannot make a decision about whether the regularity is intended without having additional information about the model, such as other regularities, consistency with respect to design intent, and solvability of the related constraint system.

8. Conclusion

We have presented an approach to beautification of reverse engineered models as a post-processing step which can improve a raw reverse engineered model using only that model as input. Our system works reasonably well for low to medium complexity objects. The topology is adjusted appropriately and major regularities are detected and selected correctly. In future work we will address more complex models. This will include decomposing models into suitable sub-parts, expanding regularity detection, and improved selection methods considering combinations of regularities.

Acknowledgements

This project was supported by UK EPSRC Grant GR/M78267 and NUF-NAL Grant 00638/G. We wish to thank T. Várady and P. Benkő of the Hungarian Academy of Sciences and CADMUS Ltd. for providing reverse engineering software and helpful discussions.

References

- [BKV*02] BENKÓ P., KÓS G., VÁRADY T., ANDOR L., MARTIN R.R.: Constrained fitting in reverse engineering. *Computer-Aided Geometric Design* 19:3 (2002) 173–205.
- [BMV01] BENKÓ P., MARTIN R.R., VÁRADY T.: Algorithms for reverse engineering boundary representation models. *Computer-Aided Design* 33:11 (2001) 839–851.
- [GLM*03] GAO C.H., LANGBEIN F.C., MARSHALL A.D., MARTIN R.R.: Approximate congruence detection of model features for reverse engineering. In *Proc. Int. Conf. Shape Modelling and Appl.* (2003) 69–77.
- [GLM*03a] GAO C.H., LANGBEIN F.C., MARSHALL A.D., MARTIN R.R.: Local Topological Beautification of Reverse Engineered Models. *Computer-Aided Design* (2004) to appear.
- [HLS97] SITHARAM M., HOFFMANN C., LOMONOSOV A.: Finding dense subgraphs of constraint graphs. In *Constraint Programming* (1997) 463–478.
- [KMV00] KÓS G., MARTIN R.R., VÁRADY T.: Methods to recover constant radius rolling ball blends in reverse engineering. *Computer-Aided Geometric Design* 17:2 (1999) 127–160.
- [Lan03] LANGBEIN F.C.: Beautification of Reverse Engineered Geometric Models. PhD thesis, Dept. Computer Science, Cardiff University (2003). <http://www.langbein.org/research/BoRG/beautification.pdf>.
- [LMM*01] LANGBEIN F.C., MILLS B.I., MARSHALL A.D., MARTIN R.R.: Approximate geometric regularities. *Int. J. Shape Modeling* 7:2 (2001) 129–162.
- [LMM*01a] LANGBEIN F.C., MILLS B.I., MARSHALL A.D., MARTIN R.R.: Finding approximate shape regularities in reverse engineered solid models bounded by simple surfaces. In *Proc. 6th ACM Symp. Solid Modelling and Appl.* (2001) 206–215.
- [LMM*01b] LANGBEIN F.C., MILLS B.I., MARSHALL A.D., MARTIN R.R.: Recognizing geometric patterns for beautification of reconstructed solid models. In *Proc. Int. Conf. Shape Modelling and Appl.* (2001) 10–19.
- [LMM03] LANGBEIN F.C., MARSHALL A.D., MARTIN R.R.: Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design* 36:3 (2003) 261–278.
- [ML03] MILLS B.I., LANGBEIN F.C.: Determination of approximate symmetry in geometric models — an exact approach. (2003) submitted.
- [MLM*01] MILLS B.I., LANGBEIN F.C., MARSHALL A.D., MARTIN R.R.: Approximate symmetry detection for reverse engineering. In *Proc. 6th ACM Symp. Solid Modelling and Appl.* (2001) 241–248.
- [MLM*01a] MILLS B.I., LANGBEIN F.C., MARSHALL A.D., MARTIN R.R.: Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components. Tech. Report GVG 2001-1, Geometry and Vision Group, Cardiff University (2001). <http://ralph.cs.cf.ac.uk/papers/Geometry/survey.pdf>.
- [PC03] PARK J.C., CHUNG Y.C.: A tolerant approach to reconstruct topology from unorganized trimmed surfaces. *Computer-Aided Design* 35:9 (2003) 807–812.
- [TOG*99] THOMPSON W.B., OWEN J.C., DE ST. GERMAIN J., STARK S.R., HENDERSON T.C.: Feature-based reverse engineering of mechanical parts. *IEEE Trans. Robotics and Automation* 15:1 (1999) 57–66.
- [VM02] VÁRADY T., MARTIN R.R.: Ch. 26: Reverse Engineering. In *Handbook of CAGD*, Elsevier, 2002.
- [WFR*02] WERGI N., FISHER R.B., ROBERTSON C., ASHBROOK A.: Shape reconstruction incorporating multiple non-linear geometric constraints. *Constraints* 7:2 (2002) 117–149.