

**Timothy J. Tautges, “Automatic Detail Reduction for Mesh Generation Applications”, Proceedings, 10<sup>th</sup> International Meshing Roundtable, SAND2001-2976C, Sandia National Laboratories, pp. 407-418, October 2001.**

# AUTOMATIC DETAIL REDUCTION FOR MESH GENERATION APPLICATIONS

Timothy J. Tautges

*Sandia National Laboratories<sup>1</sup>, Albuquerque, NM,  
University of Wisconsin-Madison, Madison, WI  
tjtautg@sandia.gov*

## ABSTRACT

Increasing resolution in computational simulation is enabling the use of original CAD design models as the basis of mesh generation. Because of varying resolution requirements, geometric detail removal is a critical part of this process. Three classes of geometry details are classified by their removal method: direct detail removal seeks to remove details directly by compositing them with neighboring topology; blend removal removes blends by first partitioning blends then compositing the pieces with neighbors; bridge removal requires more advanced decomposition, and is not described here. Key capabilities for all these methods include defining a size measure, and detecting geometric details based on those. Blend detection is also described. Results are given for automatic detail reduction applied to several real parts.

**Keywords:** mesh generation, computational geometry, detail reduction

## 1 INTRODUCTION

Computer hardware speed has been increasing at a remarkable pace over the last few decades, fulfilling a prediction known today as Moore's Law [1]. More recently, a focused effort on developing numerical modeling software which can take advantage of the fastest hardware has been sponsored by DOE's ASCI [2] and SciDAC [3] programs. The analysis codes developed under these programs are now capable of predicting component response at unprecedented levels of detail and complexity. In response to these new capabilities, the resolution of analyses being performed on the fastest computers has also increased over time. Although there have been many advances in mesh generation technology recently [4][5] these advances have not kept up with increases in computer speed and resolution. Therefore, mesh generation time (including both interactive and cpu time) for these models is still increasing relative to the analysis time for high-end problems.

For many analyses, an increase in resolution in the analysis model is often accompanied by an increase in detail in the domain model, or the geometry of the component being analyzed. On the spectrum between fully detailed geometry (i.e. the original CAD design model) and a geometry with almost no detail, typical resolutions for large models are

approaching the fully detailed end of the spectrum. For problems like this, starting the meshing process using the original CAD design model is becoming an option. However, design models often must be converted into models suitable for mesh generation and finite element analysis. This conversion process can be lengthy, and if not done carefully, must be repeated after even small changes to the original design model. This paper describes a solution for one part of this process, the removal of design details from CAD models.

Design models are usually constructed for the purposes of manufacturing, and therefore contain fine details that are part of the as-manufactured component. Examples of these details include fillets, which dissipate stress singularities, and rounds, which remove sharp edges, often for cosmetic purposes. These details can complicate the mesh generation process, especially for hex mesh generation but also for other types of meshes. Therefore, analysts often remove design details before meshing a geometry; that is, they coarsen the resolution of the geometric model so that it more closely matches the resolution of the target analysis. This simplification process reduces the subsequent mesh generation time, for little cost in terms of analysis accuracy.

Removal of geometric detail can be time-consuming, requiring manual manipulation of the CAD model. Automation of this process could significantly reduce the

---

<sup>1</sup> SANDIA IS A MULTIPROGRAM LABORATORY OPERATED BY SANDIA CORPORATION, A LOCKHEED MARTIN COMPANY, FOR THE UNITED STATES DEPARTMENT OF ENERGY UNDER CONTRACT DE-AC04-94AL85000.

overall mesh generation time. There have been various approaches to automatic detail removal; these are reviewed below.

### **1.1 Previous efforts**

Because the appearance of integrated design-analysis packages is relatively recent, there are not many integrated solutions for geometric detail removal delivered in the marketplace. However, there are several research efforts which have addressed this problem.

When starting with a design model in a CAD system, the most common approach to removing details is to remove them directly in the CAD system. This is often done by the analyst, or by the draftsman with the analyst looking over his shoulder. Obviously, this approach requires knowledge about the CAD system (hence the draftsman!). However, this approach can also introduce problems with consistency, where small changes to the initial design model after starting the model generation are not reflected in the analysis model. Also, this approach is difficult to reproduce after model changes resulting from analysis. Therefore, manually removing detail can serve as an impossible bottleneck in the design-analysis iterations often used in industry. Finally, changes made in the CAD system can be difficult to reverse, in cases where the analysis requires more resolution in a given area than initially thought.

One solution being discussed at Sandia National Laboratories and elsewhere is to use multi-level or layered design models. In this approach, a component is designed in progressively higher-resolution layers. Modern CAD systems like Pro/Engineer [6] and SolidWorks [7] have built-in support for layering detail in a model, and simplify the removal of details (by suppressing one or more layers). One advantage of this approach is that it reflects how design is usually done, starting with an abstract concept and moving toward a detailed design. However, there are also disadvantages when these models are used in analysis. If the desired resolution does not closely match that of a pre-defined layer, changes to the model are still required. Even worse, the layered approach does not work when the desired analysis resolution varies across the model; in this case, the fullest-detail layer must be used over the entire model, again requiring the removal of detail at other locations. Also, fine details in the model are often not the result of fine features in feature-based modelers, but rather the interaction between larger features; these details typically will not be removed by suppressing fine-resolution layers in the CAD system. Finally, we often do not have the luxury of designing a component from scratch; rather, we are working with models constructed and modified over the years. These models are often difficult to convert into layered models. While we feel that the layered approach has promise and should be incorporated into design practices, we recognize that it will be awhile before this change propagates down to the draftsmen actually constructing these models.

There have been several efforts aimed at removing details after an initial mesh has been generated, e.g. Ref. [8]. In this approach, details are removed by performing well-known mesh transitions, e.g. collapsing the face of a tetrahedron to

remove the tet element. While the process of removing elements from a mesh using primitive operations is well-understood, the problem remains of generating the initial mesh on the fine resolution of the model. This can be both time- and memory-intensive. Also, in cases where the desired analysis resolution is much coarser than the original model, this approach seems quite wasteful; these cases would probably still require some initial coarsening of the geometric model before meshing began. Sheffer has investigated the clustering of surface facets into groups having similar normal vectors or matching other geometric criteria [9]. While this solves the problem of generating an initial mesh, it relies heavily on geometric information, and seems to not account for topological details.

There have been several geometry-based solutions presented in the literature [10][11][12]; our approach is most similar to these. Armstrong et. al present a method based on the medial axis transform of an object [10]. While the MAT is quite useful for providing details about the non-local aspects of geometry, it typically does not contain information about “imprints” on the geometry, or groups of surfaces where the geometry is the same but the topology varies. Blacker [12] discusses approaches based on detecting small geometric features, and removing the features using “virtual topology” operations. Key issues that arise in all these efforts, and in the research reported in this paper, are the level of user interaction required, especially for the adjusting of multiple free parameters, and of course the overall robustness when applied to complicated models. It is not clear from the descriptions of [10][11][12] if these issues are addressed sufficiently to produce a usable capability.

### **1.2 Contribution**

While the solution described in this paper does not fully resolve the key issues of user interaction and robustness, it does contain slightly different approaches to some of them which improves on past work. A different size measure is introduced which more accurately reflects characteristic size. This paper also discusses cases where simple virtual topology operations are not sufficient for removing details, and provides solutions; this directly improves robustness. This paper presents a grouping of features into three classes, each addressed by a different method of removal, and elaborates on those methods. In particular, a new method for automatically identifying blends is introduced.

### **1.3 Organization**

This paper arranged as follows. Section 2 describes what we assume to be the initial conditions for detail removal, and describes our general approach. Section 3 discusses the important issue of characteristic size, and introduces our method for measuring it. Sections 4, 5 and 6 describe our approaches to direct detail removal, blend removal, and bridge removal, respectively. Section 7 contains a final discussion of this research and gives conclusions.

## 2 ASSUMPTIONS & GENERAL APPROACH

Before describing our research efforts in detail removal, it is useful to describe the starting conditions. The geometry which serves as the basis for finite element models usually comes from one of two sources: the analyst constructs the geometric model from the bottom-up, sometimes using an existing CAD model as a source of primitives; or, the analyst starts with a fully-detailed CAD model which has been constructed for other (i.e. non-CAE) purposes, e.g. a manufacturing CAD model. The latter source is becoming more common, at least for higher-end problems, and is the basis of this research.

### 2.1 Design and Analysis Models and the Analysis Process

It is rare that mesh generation can be done directly on the fully-detailed CAD model, which we refer to as the “design model”. More typically, changes are necessary to arrive at the geometry which is the basis of the mesh, which we refer to as the “analysis model”. These changes fall into three categories:

1. **Geometry Healing:** that is, the removal of geometric detail which does not reflect design intent and which usually appears as the result of translation between CAD systems. Although geometry healing can be an arduous process, commercial software solutions are beginning to be effective [13][14]. This issue is not addressed in this paper.
2. **Model Idealization:** where the various geometric components are made to “fit together”. This requires removing small gaps and overlaps in the model. Note that these details are not outside design intent. For example, we have observed “press fits”, where pins were designed larger than the hole they fit in, which appear as overlapping bodies in a model. Gaps in a model are more common, for example being used to represent bolts which are shorter than the holes they fit into. We assert that in most of these cases, these details represent the needs of downstream applications, e.g. manufacturing, and should not be part of the original design model, but added afterward.
3. **Detail Suppression:** This step is specialized for analysis model generation, where small details in the design model are removed to avoid resolving them in the mesh. This is done both to simplify mesh generation and to avoid large numbers of elements or abrupt transitions in mesh size required to resolve the small features.

For the purposes of this research, we assume that the starting point is a geometric model where all details which are outside design intent have been removed, and where the model has been idealized, such that the geometry defines a space common to most downstream applications. That is, we

assume that changes described in 1 and 2 above have been completed. This paper describes how to do step 3.

The steps above feed into an overall analysis process:

4. Generate mesh
5. Analyze & interpret results
6. Repeat 3-5 as needed, making changes in steps 3 or 4 to account for design changes or mesh resolution changes, respectively.

An important goal, one which is coming within reach, is to use analysis as the basis of product design activities. However, the efficiency of the approach above depends on the detail suppression step being:

- **Automatic**, to minimize the user interaction required per analysis iteration;
- **Reversible**, to allow user adjustment of automatic simplifications; and
- **Repeatable**, so that detail reduction can be performed on a model slightly different than the previous one (e.g. after local design model changes resulting from interpretation of analysis results).

The detail reduction approach described in this paper is distinguished from past efforts in all three categories.

### 2.2 Three Types of Details

There are two steps to our detail suppression approach, which we address separately. First, criteria must be chosen which determine the topological entities designated as “small” (we refer to these as “detail entities”). The other step is the removal of detail entities after being designated small. We identify three detail removal methods, distinguished by the types of detail entities they apply to as well as the removal method itself:

**Direct removal:** Certain geometric details can be directly removed from the model using “Virtual Topology” operations.

**Blend suppression:** Blends are commonly used in designs, but are often not resolved in analyses. Removal of these features usually requires a combination of virtual topology operations.

**Bridge feature removal:** Bridge features are parts of a geometric model which link distinct regions of a geometry with each other. These features can be either positive (a bridge) or negative (a through-hole); however, their removal is similar.

After discussing the first issue of defining what is small, the removal of each of these types of details is described.

## 3 CHARACTERISTIC LENGTH AND “SMALL” DEFINITION

Before determining whether a given feature is small, we must be able to measure its size,  $S$ . The simplest size measure for

an object is its extent; that is, the length, area, and volume of a curve, surface, or volume, respectively:

$$S_c = \text{Length}(c)$$

$$S_s = \text{Area}(s)$$

$$S_v = \text{Volume}(v)$$

This measure makes sense for curves, since it relates directly to mesh size. However, the measures for surface and volume should really be normalized to units of length, so that they can be compared to requested mesh size. The simplest normalization would be to assume a square or cubic shape, and compute a characteristic length based on that:

$$S_c = \text{Length}(c) \quad (1)$$

$$S_s = (\text{Area}(s))^{1/2} \quad (2)$$

$$S_v = (\text{Volume}(v))^{1/3} \quad (3)$$

The problem with these normalizations is that they tend to over-estimate the characteristic size, for example for long, thin surfaces. Therefore, we borrow a measure of characteristic size from fluid flow modeling, that of hydraulic diameter. The hydraulic diameter for surfaces and, by extension, volumes, are [15]:

$$S_s = D_h(s) = 4A/P \quad (4)$$

$$S_v = D_h(v) = 6V/A \quad (5)$$

where P is the surface perimeter, and A and V are area and volume, respectively.

**Error! Reference source not found.** shows the size measures computed by (2) and (4) for various surfaces in a real part. The hydraulic diameter is either much smaller than or about the same as the square root of the area. For some surfaces, the difference is significant (more than a factor of five), and tends to be larger for larger surfaces. The hydraulic diameter-based size measure tends to give a more accurate representation of characteristic size.

The notion of a feature which is “small” only makes sense when measured relative to something else. When applied to the mesh generation process, that something else should be the mesh size, since that is the quantity affected by detail size. The determination of what’s small compared to the local mesh size  $S_m$  is somewhat arbitrary; for the purpose of this research, we choose 1/3 local mesh size, assuming the smallest element in a mesh is no smaller than 1/3 the average mesh size.

Using the local mesh size as a relative measure of what is small has two primary advantages. First, it allows the size measure to vary across the model, rather than being constant for all bodies. This is more natural, since it fits more closely the way mesh sizes vary across a typical mesh. Also, assuming the detail removal process is reversible and repeatable, defining detail size using the local mesh size, features can be removed or restored simply by changing the local mesh size. This provides a more direct coupling between the geometric feature size and the local mesh size. That is, it provides more geometric detail in areas where the mesh is small enough to resolve that detail, and vica versa.

Figure 2-4 show a part resolved to several mesh sizes, with the appropriate level of detail in the geometric model resolved. Obviously, there are cases where this technique will not work, especially for hexahedral meshing; however, the general trend of more detail as mesh is refined is correct.

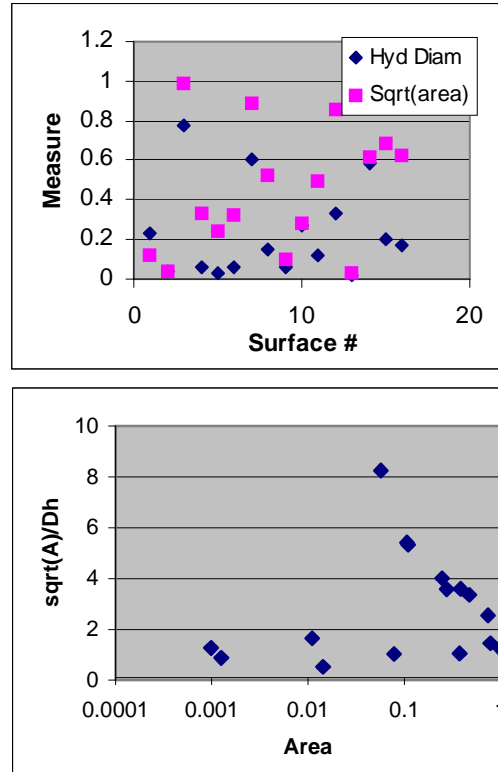


Figure 1: Size measures computed by various methods for various entities (top); ratio of sqrt(A)/Dh versus area (bottom).



Figure 2: A part resolved to a coarse mesh size, 0.5, including appropriate amount of geometric detail.

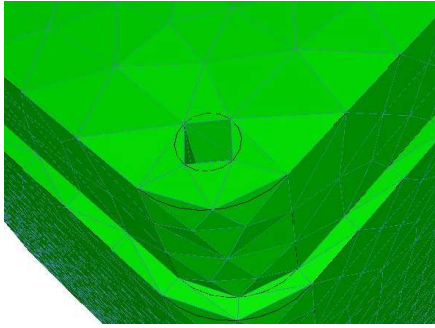


Figure 3: A part resolved to a coarse mesh size, 0.15, including appropriate amount of geometric detail.

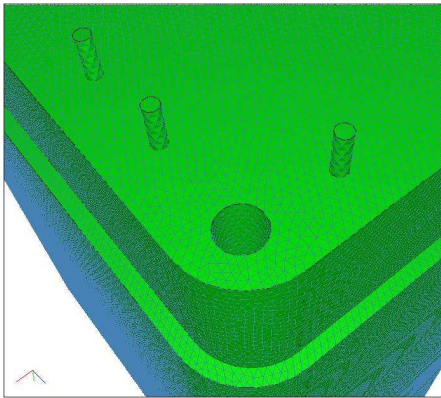


Figure 4: A part resolved to a coarse mesh size, 0.04, including appropriate amount of geometric detail.

#### 4 DIRECT REMOVAL

In the direct removal method of detail suppression, detail entities are removed using virtual topology “composite” operations [16]. The simplest cases of direct removal are shown in Figure 5; here, each detail entity has only one other entity with which to composite, and the small detail represented by the detail entity is removed by the operation. In many cases, however, there is more than one adjacent entity which could be composited with the detail entity. Some of these combinations yield topologically incorrect models, while others are topologically valid but do not remove the small detail represented by the detail entity. Several examples of these situations are shown in Figure 6. To choose the best pair of entities to composite together, we introduce the notion of topological and geometric composite-compatibility.

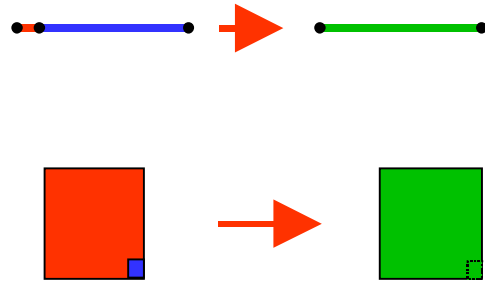


Figure 5: Simple cases of direct removal of detail curve (top), and surface(bottom).

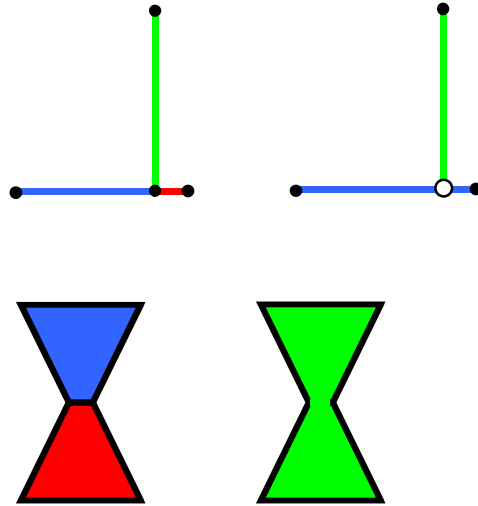


Figure 6: Examples of composite operations which produce topologically invalid results (top) or which do not remove the small detail (bottom).

Two entities  $N_i$  and  $N_j$  are said to be topologically composite-compatible if the composite operation leaves the topology in a valid configuration, i.e. where entities are bounded by lower-order topology (except in the case of periodic entities). Let  $N_i$  of dimension  $d$  have a set of adjacent entities  $N = \{N_j; j = 1, n\}$  of same dimension, where entities are adjacent if they share one or more entities  $n_k^{d-1} = \{N_i \cap N_j\}^{(d-1)}$  of dimension  $d-1$ . Then  $N_i$  and  $N_j$  are composite-compatible if and only if the parents of  $n_k^{d-1} = \{N_i, N_j\}$ . Proof:  $n_k^{d-1}$  is/are removed as part of the composite operation[16]. However, if  $\text{parent}(n_k^{d-1}) = \{N_i, N_j, N_k\}$ , removing  $n_k^{d-1}$  leaves  $N_k$  unbounded, i.e. produces topologically ill-defined entities, as in Figure 6 (left).  $\square$

In many cases there are several choices of neighboring entities which are topologically compatible with the detail entity  $N_i$ . In this situation,  $N_j$  is the entity which is closest in geometry to  $N_i$ , such that their composite will be as

continuous as possible. We denote this as geometric composite-compatibility. Specifically, we indicate geometric composite-compatibility  $C_g$  as how close to continuous two entities are:

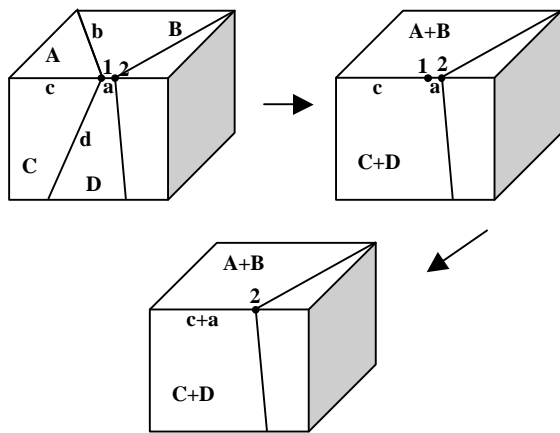
$$C_g = \text{abs}(t_i \bullet t_j)_{(nk)} \quad (\text{curves}) \quad (6)$$

$$= \text{abs}(n_i \bullet n_j)_{(nk)} \quad (\text{surfaces}) \quad (7)$$

where tangents or normals are measured local to the entity  $n_k$ .  $C_g$  close to unity are preferred.

A third situation which is quite common in practice is where there is no entity of equal dimension which is topologically composite-compatible with the detail entity  $N_i$ . Figure 7 shows an example where the detail entity (curve a) does not have a neighboring entity which is topologically composite-compatible. In situations like this, the detail entity cannot be removed simply by combining it with a neighboring entity; rather, entities of higher dimension must be composited. This has the net effect of reducing the valence of one of the entities  $n_k$ , eventually yielding a composite-compatible pair of entities  $N_i$  and  $N_j$ .

For example, in Figure 7, the detail curve, curve a, is bounded by two vertices of valence 4. Curve a cannot be directly combined with any of the curves adjacent to vertex 1, since that would leave the other curves connected to vertex 1 without a vertex on one end; the same problem exists with vertex 2. However, vertex 1 can be reduced to valence 2 by eliminating some of the incident curves; this is done by compositing surface pairs bounded by incident curves. For example, curve b is removed by compositing surfaces A and B. Curve d is removed by compositing surfaces C and D. After these composite operations, vertex 1 has valence 2, and curve a can be composited with curve c.



**Figure 7: Example of a detail entity (curve a) having no composite-compatible neighbors of the same dimension (top left). Composite surfaces A and B and surfaces C and D (top right). Valence of vertex 1 reduced so that curves a and c can be composited (bottom).**

In the above example, some choices of higher-dimension composites are better than others, due to geometric

compatibility. For example, compositing surfaces A and C to eliminate curve c, while possible, is clearly not as good a choice as compositing surfaces A and B. The best choice for compositing higher-dimension entities can be quantified by measuring geometric compatibility, and choosing entity pairs which have compatibility closest to unity. The algorithm used to perform direct curve removal is shown in Figure 8.

- **For small curve C:**
1. Choose surface S1 with smallest Dh incident on C; add to list A
  2. Find another surface S2 that:
    - Shares an edge with S1
    - Does not contain C
    - Is compatible with S1
    - S(S1 U S2) > eps
  3. Mark vertex V that shares S1, S2, C
  4. Put S2 on list A
  5. Repeat 2-4 until Valence(V) - size(A) + 1 = 2 or no new surfaces in A
  6. If Valence(V) > 2, repeat 2-4 with other surface incident on C, building list B
  7. If Valence(V) > 2, clear lists A & B, repeat 2-5 with other vertex of C
  8. If Valence(V) - size(A) - size(B) + 2 = 2, composite surfaces in list A together, same for B
  9. Composite C with curve adjacent to C and V

**Figure 8: Algorithm used to perform direct detail removal, including identification of topologically and geometrically composite-compatible entities.**

Note that the algorithm in Figure 8 allows two sets of composited entities, each set having geometric compatibility with other surfaces in the set. Note also that the set(s) of surfaces to be composited are identified before doing any of the actual compositing; this way, if the result still does not yield a pair of topologically composite-compatible entities, the operation is abandoned for the current lower-order entity  $n_k$ , and the algorithm moves on to another lower-order bounding entity  $n_k$ .

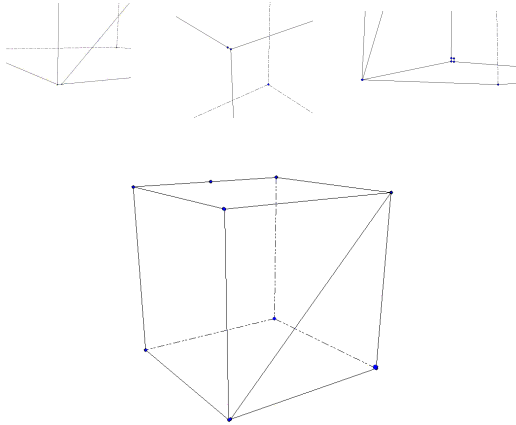
#### 4.1 Geometric versus Topological Compatibility

The notion of geometric compatibility has been described in other works, e.g. Ref. [9]. If geometric compatibility is enforced on an equal setting as topological compatibility, as in these works, the result can easily be the persistence of small details. For example, the cylindrical protrusions and the small ledge on the part in **Error! Reference source not found.** are geometrically incompatible with their neighbors, and by that measure would not be removed using direct removal. However, the surface are topologically composite-compatible with their neighbors. Moreover, we assert that if we are removing a small detail, geometric compatibility is a moot point and should be given lower priority.

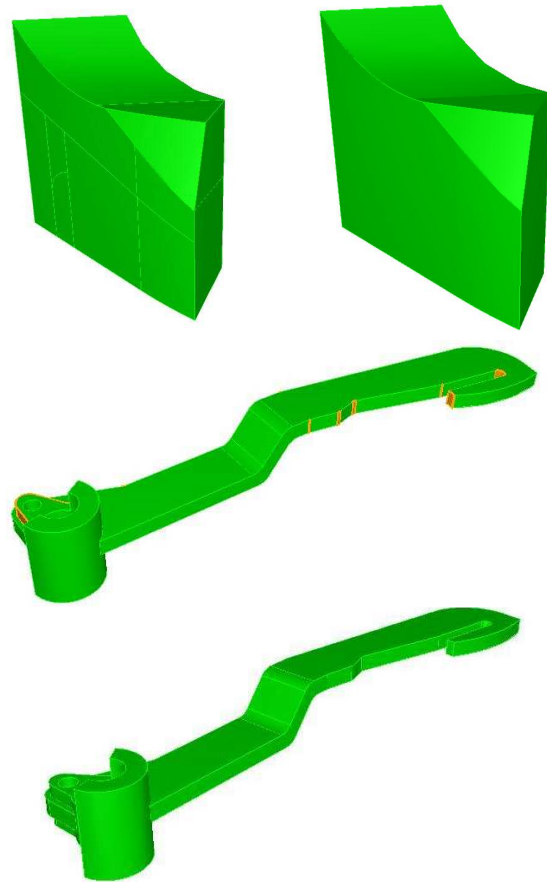
Lowering the priority of geometric compatibility has an advantageous side-effect of removing some of the free parameters described in [9]. This is very important to the automation of detail removal, and needs to be investigated further.

## 4.2 Examples

Figure 9 and Figure 10 show examples of direct removal, on a contrived part to illustrate various cases (Figure 9) and a few real parts to show robustness and applicability to real problems (Figure 10).



**Figure 9: Direct detail removal on a contrived part; detail vertex, two detail curves, and a detail surface. All detail entities are removed automatically to produce a brick with 8, 12 and 6 vertices, edges, and faces, respectively.**



**Figure 10: Direct detail removal for 2 parts, showing original part and part after detail removal.**

## 5 BLEND SUPPRESSION

Blends are design features included in a model for aesthetic purposes (e.g. rounding off sharp edges) or to avoid stress singularities (fillets). As their name implies, blends make smooth transitions between two or more surfaces. Blends are one of the most common design model details removed for analysis. Removing blends requires the automatic detection of blends; once blends are correctly identified, their removal is straightforward. Blend removal requires both partition and composite operations. Blend detection and removal is discussed briefly here; a more complete description will be given in a future paper.

### 5.1 Blend topology & nomenclature

Although blends are used in virtually every design system, they are often referred to using different nomenclature. Figure 11 shows an example of a brick with three surface pairs blended to form rounds; nomenclature for various parts of the blends are indicated in the figure, and briefly discussed below.

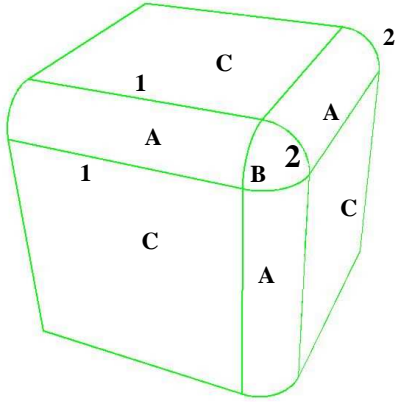


Figure 11: Blend topology, including blend sheets (A), blend blend (B), blended faces (C); and spring curves (1) and crossing curves (2). Only selected curves are labeled.

- **Blended face:** one of two surfaces between which the blend forms a smooth transition.
- **Blend sheet (BS):** the surface which transitions between a pair of blended faces.
- **Blend blend (BB):** a blend sheet which transitions between three or more blend sheets.
- **Spring curve (SC):** a topological curve separating a blend sheet from a blended face. Note that the transition between the surfaces at a spring curve is C1-continuous.
- **Crossing curve (XC):** a curve which cuts across a blend sheet and which joins two spring curves, one on each side of the blend sheet. Note that the transition between surfaces at a crossing curve may or may not be C1-continuous.

## 5.2 Automatic blend detection

At its simplest, a blend is characterized by a blend sheet which transitions between two blended surfaces, such that the surfaces are C1-continuous where they meet. If all blends were like this (and no other non-blend surfaces were), detecting them would be a simple matter of looking for curved surfaces with C1-continuous transitions to adjacent surfaces, with crossing curves at the end. However, blends often interact with each other, and with other features; for example, Figure 12 shows a few examples of more complicated blends. Hence, a simple topological check is not sufficient for detecting all but the simplest blends.

This section describes a heuristic method for finding blend sheets and blend blends, as well as the bounding spring and crossing curves. Once these surfaces and curves are identified, decomposing them to remove the details is a simple matter. The blend finding heuristic consists of local geometric and topological checks, along with non-local topological checks.

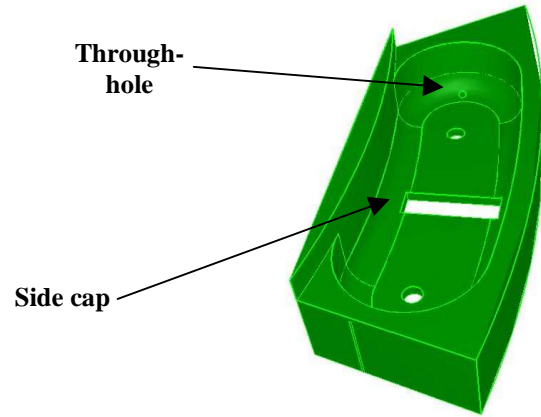


Figure 12: A few examples of more complicated blends.

### 5.2.1 Geometric criteria

Consider one of the blends shown in Figure 11. Finding a blend first requires identifying spring and crossing curves. Figure 13 shows two views of a blend sheet, viewing the blend from the top (top) and from the side (bottom).

Spring curves represent C1-continuous transitions between two surfaces; therefore, a curve cannot be a spring curve unless the normals on adjacent surfaces are within a small tolerance of each other:

$$\mathbf{n1}^+ \cdot \mathbf{n1}^- (1-\epsilon) \quad (\text{GC1})$$

Where  $\mathbf{n1}^+$  and  $\mathbf{n1}^-$  denote the normals to surfaces on either side of  $P1$  from the blend sheet.

Blend sheets form a C1-continuous transition between two surfaces; this is most easily done using a surface with constant radius of curvature in the plane orthogonal to the blend sheet and passing through the crossing curve  $c0$ . Assuming  $c0$  is a constant-radius curve, the radius of curvature can be computed from its end points and end point surface normals using basic trigonometry:

$$r = d \tan(\alpha/2) = c/\cos(\alpha/2) \quad (8)$$

$$\cos^2(\alpha/2) = 1/2(1-\cos(\beta))$$

$$\cos(\beta) = \mathbf{n1} \cdot \mathbf{n2}$$

$$C = 1/2 |\mathbf{P1P2}|$$

$$r = |\mathbf{P1P2}| / (8(1-\mathbf{n1} \cdot \mathbf{n2}))^{1/2} = \text{computed radius} \quad (8)$$

How does that radius compare with the actual radius of curvature of the surface in that location? This can be found by projecting the principal curvatures onto the vector joining the two crossing curve end points:

$$\mathbf{c1} = \mathbf{t1} \times \mathbf{n1}$$

$$\mathbf{c2} = -\mathbf{t2} \times \mathbf{n2}$$

$$\kappa1|_{c1} = \mathbf{c1} \cdot (\kappa_a + \kappa_b) |_{P1}$$

$$\kappa_2|_{c_2} = -c_2 \cdot (\kappa_a + \kappa_b) |_{P_2}$$

$$R = (\text{avg}(\kappa_1, \kappa_2))^{-1} = \text{measured radius} \quad (9)$$

where  $\kappa_a$  and  $\kappa_b$  are the principal curvatures on the surface.

We assert that if a curve is a crossing curve, its computed radius of curvature will be close to the measured radius at the end points:

$$r/R \quad (1 - \varepsilon) \quad (GC2)$$

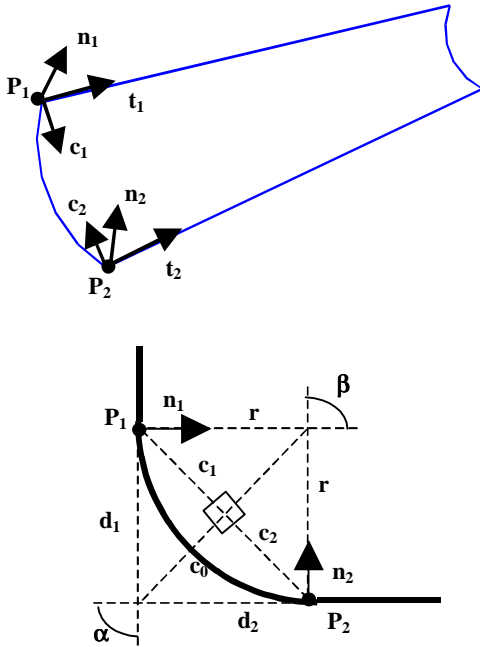
In order to prevent degenerate cases, i.e. when radius of curvature is infinite, and cases where surface curvature varies widely along a crossing curve, we add the following two criteria:

$$\text{abs}((\kappa_1 - \kappa_2)/\text{avg}(\kappa_1, \kappa_2)) < \varepsilon \quad (GC3)$$

$$\text{abs}(\kappa_1) > \varepsilon \quad (GC4a)$$

$$\text{abs}(\kappa_2) > \varepsilon \quad (GC4b)$$

These constraints enforce a certain amount of uniformity and smoothness across blend sheets, but does not require the radius of curvature to be constant. This flexibility is critical for robustness.



**Figure 13: Identification of crossing curves (top) and spring curves (bottom) using geometric criteria GC1-4.**

Note that for crossing curves (GC2-GC4), there are no criteria involving the end-point normals as measured for the blended surfaces. This means that a crossing curve also can satisfy the spring curve criteria, although this is not required; this will be important for evaluating non-local topological criteria. In the geometric criteria stage of blend finding, curves satisfying the crossing curve constraints GC2-4 are

marked as crossing curves; the remaining curves satisfying GC1 are marked as spring curves.

Geometric criteria GC1-4 identify curves which *can* be spring or crossing curves. However, these criteria are necessary, but not sufficient, for determining these curves. Stated another way, there can be C1-continuous curves or other curves satisfying GC1-4 in a model which do not bound blend sheets. Obviously, other criteria are needed to correctly identify blend sheets and blends.

### 5.2.2 Local topological criteria

A blend sheet is distinguished from a surface bounded by an arbitrary collection of spring and crossing curves by the specific arrangement of the curves on the surface. We observe that blend sheets typically contain:

- 2 chains of spring curves, where each chain is a series of spring curves arranged end-to-end with tangents on either side of interior vertices approximately parallel,
- 2 chains of crossing curves
- No other curves

This leads to the following local topology criteria for blend sheets:

$$\# \text{ chains of SC's} == 2 \quad (LTC1a)$$

$$\# \text{ chains of XC's} == 2 \quad (LTC1b)$$

Blend blends are defined as surfaces which transition between 3 or more blend sheets. Since a smooth transition is typical, the blend blend will meet each blend sheet at a curve representing a C1-continuous transition between the surfaces. However, blend sheets meet blend blends only at their ends, since the spring curves separate the blend sheets from the blended faces. Therefore, blend blends and blend sheets are separated by crossing curves.

Blend blends sometimes have “offsets”, where the crossing curves do not join directly but rather are separated by offset curves. These offset curves are still C1-continuous transitions, but they transition between the blend blend and some of the blended faces associated with the blend sheets.

Based on these observations, we use the following local topology criteria for blend blends:

$$\# \text{ XC chains} \geq 3 \quad (LTC2a)$$

$$\# \text{ SC chains} == 0 \parallel == \# \text{ XC chains} \quad (LTC2b)$$

For each SC chain, the surface opposite the blend blend is shared by the blend sheets corresponding to the XC chains on either side of the SC chain (LTC2c)

Topological criteria LTC1 and LTC2, although quite restrictive in the allowed arrangements of curves on blend sheets and blends, still are not sufficient for robustly identifying blends in arbitrary models.

### 5.2.3 Non-local topological criteria

The problem with the above algorithm, as it stands now, is that certain curves can be both spring or crossing curves, since a subset of the crossing curves can also satisfy spring curve constraints. Thus, a subset of curves can be changed from crossing curves to spring curves. By LTC1 and LTC2, this curve classification affects the classification of adjacent surfaces. Therefore, the identification of both curves and surfaces associated with blends is coupled, and must be done concurrently. For this reason, we observe that there is a non-local component to the blend detection process.

Solving the non-local part of this process can be done using either a heuristic or an optimization technique. The latter approach is outside the scope of this paper, and will be described in a different paper.

The most challenging part of the blend detection algorithm is to choose non-local topological constraints that force the algorithm to converge to a reasonable solution. We choose these constraints by observing the following: in typical models, the surfaces adjacent to spring and crossing curves can only be classified in a few possible ways. These arrangements are shown in Figure 14 for crossing curves (top), indicating non-local constraint NTC1, and spring curves (bottom), indicating non-local constraint NTC2.

After classifying curves and surfaces using GC1-2 and LTC1-2, each crossing and spring curve is checked against the arrangements in Figure 14. A certain number of curves will violate NTC1-2; how do we decide which curves to change? Curves which are identified as SC's and which violate NTC1 are reset to no classification, while XC's which violate NTC2 and which satisfy GC1 are re-classified as SC's (otherwise they are reset to no classification). After re-classifying curves, the surfaces bounding these curves are re-evaluated against LTC1-2, and the process is repeated. Since these iterations monotonically decrease the classification of curves and surfaces towards no classification, this process will converge eventually (although it is possible to converge on a solution consisting of no blends).

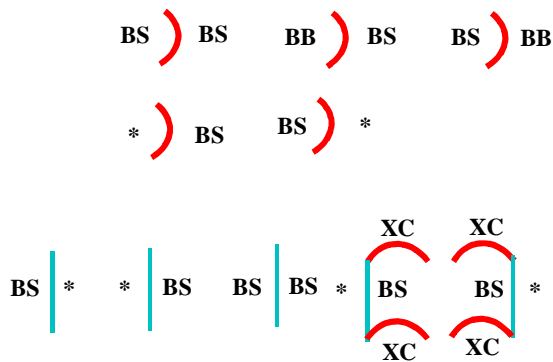


Figure 14: Allowed arrangements of surfaces adjacent to crossing curves (top) and spring curves (bottom).

### 5.2.4 Blend decomposition

Describing the process of decomposing blend sheets and blends and compositing the pieces into adjacent surfaces is omitted for brevity. However, since SC's and XC's are already identified for each blend, this process is straightforward, and follows the process normally done interactively.

### 5.3 Algorithm

The algorithm described above is summarized in Figure 15.

- For each curve, evaluate GC1-2, classify accordingly
- For each surface adjacent to spring/crossing curve, evaluate LTC1-2, classify accordingly
- For each spring/crossing curve, evaluate NTC1-2, mark invalid curve(s)
- For each invalid crossing curve:
  - If GC1 is satisfied, mark as spring curve, else mark as no classification
  - Mark adjacent surfaces
- For each invalid spring curve, mark as no classification & mark adjacent surfaces
- Re-classify all marked surfaces

Figure 15: Overall algorithm for finding and suppressing blends.

### 5.4 Examples

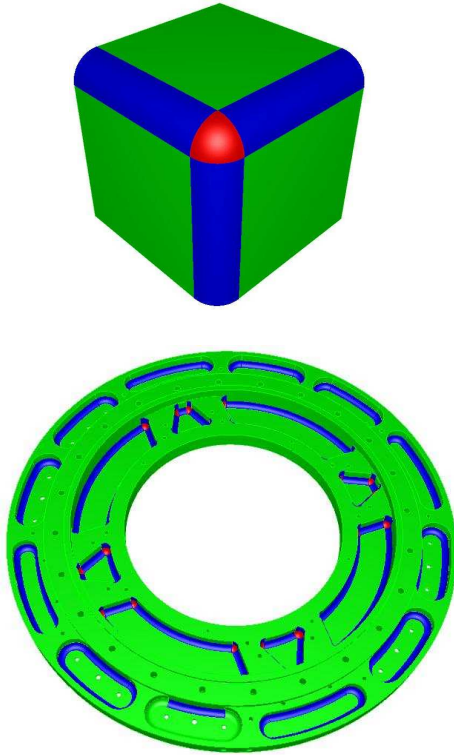
The algorithm in Figure 15 has been implemented in the CUBIT mesh generation toolkit from Sandia National Laboratories[17]. Implementing these algorithms in a production-quality meshing toolkit allows testing directly on "real" analysis problems. Several examples of blend detection are shown in Figure 16. These blends were detected fully automatically.

### 5.5 The Fine Print

There are many topological arrangements of blends found in practice which aren't handled by the algorithm above. The natural question then is whether this algorithm will work with more general geometry containing these types of blends. We discuss below several of the more common cases that appear, and how we postulate they could be handled with our algorithm.

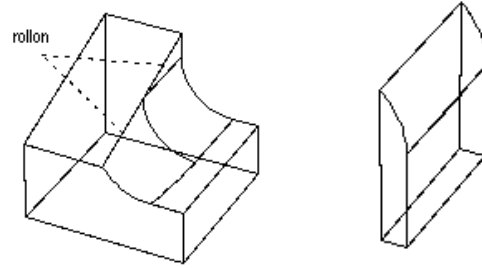
One arrangement that is quite common is surfaces containing two external loops, where the curves on each of the loops form a single chain of spring curves. This is found for example when a fillet is added at the base of a cylindrical boss. Currently, since a blend sheet must contain both spring and crossing curves, our algorithm will not detect these types of blends. This can be corrected in several ways. The easiest solution is to split periodic surfaces; this will usually split these types of blend sheets into two blend sheets, which are then easily detected by our algorithm. The other way to handle two-loop surfaces is to join the two loops with two

virtual curves, each of minimal length. These curves will pass the crossing curve test, and again the blend sheets will be detected easily.



**Figure 16: Examples of blend detection and removal. Blend sheets are shaded dark, and blend blends are shaded medium**

Another common arrangement is to have blend sheets which are cut with through-holes or blind holes; an example of this arrangement is shown in Figure 12. Simple cases, for example when the hole intersects one chain of spring or crossing curves entirely, can be handled if the curves bounding the hole are detected and considered part of the adjacent chain. Preliminary work has been done to detect these situations; however, this work has not progressed enough to report at this time. Another simple case, where the hole forms its own loop on the interior of the blend sheet or blend, could be handled by ignoring these curves in the blend determination step (although they would have to be accounted for during the partitioning step).



**Figure 17: Examples of blends not yet handled by this algorithm; roll-on blends (left), and face-edge blends (right).**

Some more complex blend configurations contain blend sheets which intersect the boundary of a blended surface (sometimes referred to as a “roll-on” blend), or where a surface is blended with an edge in the model; these configurations are shown in Figure 17. Although our algorithm has not yet been implemented to handle these configurations, our general approach is amenable to handling these. Roll-on blends can be handled by detecting the roll-on point (automatically or with user guidance) and adding the roll-on edges to the adjacent spring curve chain. Face-edge blends are probably difficult to handle without additional user input.

## 6 BRIDGE FEATURE REMOVAL

The third class of features to be handled by automatic feature suppression are referred to as bridge features. Bridge features are characterized by a material bridge with cross-section small enough to satisfy the definition of “small”. This bridge can be a positive feature, e.g. a thin-walled section, or a negative feature, e.g. a through-hole.

Algorithms for detecting and suppressing these types of details are outside the scope of this paper; they will be described in another paper.

## 7 DISCUSSION, CONCLUSIONS

The mesh generation process currently dominates the overall analysis time; most of this time is spent in interactive manipulations of the model. This is the time which must be reduced to eliminate the mesh generation bottleneck.

A key part of reducing the meshing bottleneck is using a geometry-centric process, where meshing is focused on the geometric model. In this way, much of the data related to meshing can be assigned to geometry, and can persist across multiple meshing operations. This approach is gaining in popularity, indicated by the number of different integrated packages for meshing/analysis/post-processor, e.g. SDRC Ideas [17], Ansys [19], and Pro/Engineer [6] suites of tools.

The natural outcome of this approach is to start the meshing process from the design model inside a CAD system. Assuming the starting model is well-defined, according to the

definition in Section 3, the details in the model which must be removed for a given analysis are not geometry problems, per se, but are details not relevant to that analysis. Whether a detail is important to an analysis will change between analysis types and even within an analysis. Thus, detail reduction is a meshing-time decision, rather than a part of the CAD design process. Furthermore, since fine details complicate the meshing process, it works best if these details are removed from the model before the start of meshing. Finally, because resolution requirements often vary over the course of a given analysis, detail reduction must be reversible, so that details can be recovered if they are relevant after all.

The algorithms described in this paper meet the requirements described above, largely through their use of virtual topology. The algorithms also are relatively efficient, since they rely mostly on local criteria and minimize the amount of global searching required to detect small features. The algorithms are robust, as demonstrated by the examples in Sections 4.2 and 5.4. Finally, since these algorithms do not depend on information particular to any one solid modeling system, they are portable across solid modelers.

There are several areas where this work must be extended or completed in order to provide a fully capable automatic detail suppression capability. Removal of small volume details has not yet been implemented, although detection of these details works the same as surfaces. In blend removal, more complex topology like roll-on blends and face-edge blends have yet to be handled; these configurations are not observed often in real parts, so this issue is not as critical. Finally, bridge removal has not been addressed; this is an important part of any detail suppression technique, because of the abundance of through-holes in typical design models. This capability will be implemented soon.

## 8 REFERENCES

- [1] Gordon E. Moore, "Cramming more components onto integrated circuits", *Electronics*, Volume 38, Number 8 (1965).
- [2] "Accelerated Strategic Computing Initiative", U.S. Department of Energy, <http://www.asci.doe.gov/index.htm>.
- [3] "Scientific Discovery through Advanced Computing", Office of Science, U.S. Department of Energy, March 24, 2000.
- [4] Timothy J. Tautges, "The Generation of Hexahedral Meshes for Assembly Geometry: Survey and Progress", *Int. J. Numer. Meth. Engng.*, 50: 2617-2642 (2001).
- [5] Steven J. Owen, "A Survey of Unstructured Mesh Generation Technology", *Proc. 7th International Meshing Roundtable*, SAND98-2250, Sandia National Laboratories, Oct. 1998.
- [6] Parametric Technology Corporation, <http://www.ptc.com>.
- [7] SolidWorks, SolidWorks Corp., <http://www.solidworks.com>.
- [8] M. S. Shephard, M. W. Beall, R. M. O'Bara, "Revisiting the Elimination of the Adverse Effects of Small Model Features in Automatically Generated Meshes", *Proc. 7th International Meshing Roundtable*, Sandia National Laboratories, SAND98-2250, pp. 347-364 (1998).
- [9] A. Sheffer, "Model Simplification for Meshing Using Face Clustering", to appear, *Computer Aided Design*.
- [10] C. G. Armstrong, S.J. Bridgett, R.I. Donaghy, R.W. McCune, R.M. McKeag and D.J. Robinson, Techniques for Interactive and Automatic Idealisation of CAD, in *proc. Numerical Grid Generation in Computational Field Simulations*, Ed. M. Cross., B. K. Soni, J. F. Thompson, J. Hauser, P. R. Eiseman, *Proceedings of the 6th International Conference*, held at the University of Greenwich, pp.643-662, July 1998.
- [11] Mobley, Anton V., Michael P. Carroll, and Scott A. Canann, "An Object Oriented Approach to Geometry Defeaturing for Finite Element Meshing", *7th International Meshing Roundtable*, Sandia National Labs, pp.547-563, October 1998.
- [12] Ted Blacker, Alla Sheffer, Jan Clements, Michel Bercovier, "Using Virtual Topology to Simplify the Mesh Generation Process", *Trends in Unstructured Mesh Generation*, ASME Applied Mechanics Division, Volume AMD-Vol 220, 1997.
- [13] Barequet Gill and Subodh Kumar, "Repairing CAD Models", *Proceeding IEEE Visualization*, Phoenix, AZ, IEEE, pp.363-370, October 1997.
- [14] ACIS Healing Husk, Spatial Technology Inc., <http://www.spatial.com>.
- [15] R. B. Bird, W. E. Stewart, E. N. Lightfoot, "Transport Phenomena", Wiley & Sons (1960).
- [16] Jason A. Kraftcheck, "Virtual Geometry: A Mechanism for Modification of CAD Model Topology for Improved Meshability", Master's Thesis, University of Wisconsin-Madison, 2000.
- [17] T. D. Blacker et al., 'CUBIT mesh generation environment, Vol. 1: User's manual', SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, May 1994.
- [18] SDRC Ideas Master Series, Structural Dynamics Research Corp., <http://www.sdrc.com>.
- [19] ANSYS, ANSYS, Inc., <http://www.ansys.com>.

Changes:

- remove section on blend detection
- change `edge_indirect_face_composite` to account for “best” vertex removal
- add bridge removal description, examples