A Software Demonstration of 'rap': Preparing CAD Geometries for Overlapping Grid Generation

N. Anders Petersson

Center for Applied Scienti£c Computing,
Lawrence Livermore National Laboratory,
Livermore, California, 94551 USA.
andersp@llnl.gov

Abstract

We demonstrate the application code **rap** which is part of the **Overture** library. A CAD geometry imported from an IGES £le is £rst cleaned up and simpli£ed to suit the needs of mesh generation. Thereafter, the topology of the model is computed and a water-tight surface triangulation is created on the CAD surface. This triangulation is used to speed up the projection of points onto the CAD surface during the generation of overlapping surface grids. From each surface grid, volume grids are grown into the domain using a hyperbolic marching procedure. The £nal step is to £ll any remaining parts of the interior with background meshes.

Introduction

We present the software tool **rap** for preparing geometries for mesh generation and for creating overlapping meshes. The geometries are obtained from IGES £les translated from the internal representation of a Computer Aided Design (CAD) package. Preparation of these geometries includes identi£cation and correction of translation errors as well as the removal of details regarded unnecessary for the intended analysis. A geometry is considered ready for mesh generation when it describes a watertight, consistent description of the modeled object. Our approach consists of a collection of error detection utilities in combination with an interactive interface for altering the geometry. Once the geometry has been prepared, we grow a set of overlapping surface grids starting from edges or feature curves on the geometry. Figure 1 depicts the overlapping surface grids over part of an engine manifold. The three-dimensional computational domain is £lled by growing volume grids starting from the surface grids, and any remaining void is £lled with simple (Cartesian) background grids.

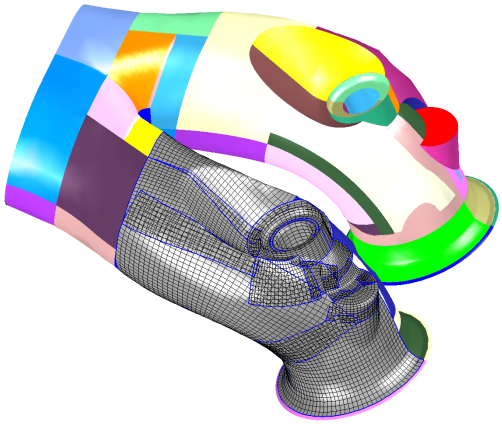

Figure 1: CAD geometry and overlapping surface grids on an engine manifold

Generating the surface meshes requires fast and accurate evaluation of the underlying surface. If the CAD representation contains errors, the mesh generator will not only re¤ect these errors, but in many cases will likely fail to generate meshes at all. While the tools are developed within the **Overture** overlapping grid framework, they are general enough to be useful for a variety of applications. In particular, they are also currently being used to generate embedded boundary (Cartesian) grids, where the surfaces cut through the computational grid in an arbitrary manner. The goal of our work is to provide a robust geometry model and ef£cient supporting software for mesh generation, to be used both during pre-processing and for moving body and adaptive mesh simulations.

Researchers have proposed several approaches to preparing CAD models for grid generation. Some have suggested generic means to represent the geometry, sometimes interfacing directly with the CAD packages themselves [11, 5]. Others attempt to deal with the representations provided by neutral £le formats generated by CAD software [10, 1, 6]. The current work falls into the latter category since we desire to eliminate the dependence on proprietary software and to reduce memory requirements enabling the CAD model to be in memory during a moving or adaptive mesh re£nement computation. While much work has been accomplished to automatically correct many problems encountered with these £les [10, 1, 6], we still £nd many signi£cant geometric de£ciencies that require user input for resolution.

In our work the geometry is described as a boundary-representation (B-REP) consisting of a network of surface patches. Currently, the surfaces are imported

using the IGES format. The quality of the geometry description in these £les vary widely depending on their source. Errors span the range of simple problems such as slight mismatches between neighboring surface patches, intermediate problems such as trim-curves that do not close on themselves, to gross geometry errors such as warped or missing surface patches.

One class of geometry errors, that is straight forward to detect, arises from problems with trimming curves. Most trimming curve problems are easily understood once the curves are inspected, and we provide a user interface for correcting these problems. Another need for user intervention arises when making meshes on a CAD model intended for production. Here models often contain parts irrelevant for the analysis and have more detail than is feasible to resolve on a mesh. Furthermore, models can sometimes be divided along symmetry planes to decrease the overall size of the problem, or to guarantee a perfectly symmetric mesh.

During the £nal stage of geometry preparation, the topology is determined and a global triangulation is created on the corrected network of surfaces. Henshaw [4] has developed an edge matching algorithm within **Overture** to perform these steps. The edge matching algorithm has also been proven to be a valuable tool for detecting any remaining gaps, overlaps, or irregular surfaces in the model.

Boundary representation

We represent CAD geometries as a patchwork of surfaces, referred to as a composite surface. A composite surface consists of several component surfaces; component surfaces are represented with different shades in Figure 1. Each component surface is described as a mapping from the unit square in R^2 (the parameter plane) to the physical space in R^3. Complex geometries often contain trimmed surfaces in their composite surface. Trimmed surfaces are comprised of parametric surfaces divided into "active" and "inactive" regions using curves in the parameter plane, as illustrated in Figure 2. Active pieces of the surface belong to the geometry description, while inactive regions should be ignored. Active and inactive regions are maintained by the orientations of the trim curves in the parameter plane.

Trim curves are given a parametric representation, $\boldsymbol{r} = \boldsymbol{r}_c(t)$, $0 \leq t \leq 1$, where $\boldsymbol{r} = (r, s)$ are coordinates in the parameter plane of the underlying surface. These curves are represented using Non-Uniform Rational B-Splines (NURBS) [9]. Each of the trim curves curves may be assembled by joining a number of curves into one continuous curve, see Petersson & Chand [8] for details.

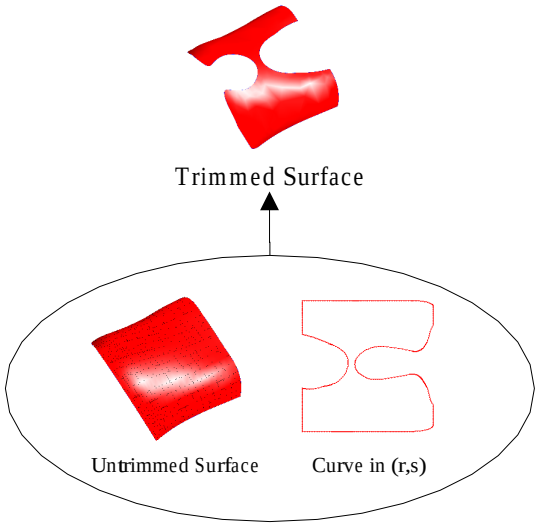

Figure 2: The trimming of a parametric surface using curves in the parameter plane

Preparing the geometry for grid generation

Some examples of common trimming curve errors are shown in Figure 3. Trim curves are manipulated in the parameter plane to avoid dif£culties dealing with surface singularities and extremely narrow surfaces, where different parts of the trim curve otherwise could be plotted very closely together. The most important tools for editing trim curves in **rap** are:

- Hide and show sub-curves. Hidden sub-curves are not considered during the assembly of the trim curve.
- Join endpoints of sub-curves with a line segment.
- Split a sub-curve into two pieces.
- Move the endpoint of a sub-curve. If the new endpoint projects onto the existing sub-curve, the sub-curve is £rst split and the remaining part of the sub-curve is discarded. Otherwise, the last control point is simply moved to the new end point.
- Truncate or extend two sub-curves to their intersection.
- Automatic assembly of sub-curves into a continuous, periodic trim curve.

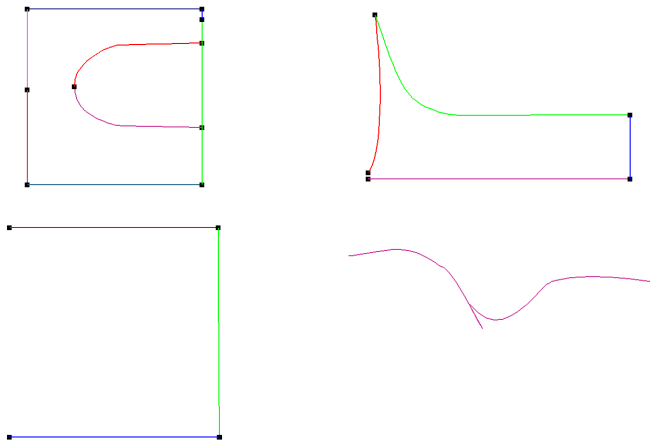

Figure 3: Examples of errors in trimming curves. The different colors represent sub-curves and the black squares show the end points of the sub-curves.

- Creation and deletion of complete trim curves. New trim curves can be created by projecting edges of other surfaces onto the trimmed surface, or by a surface-surface intersection.

CAD geometries often contain more details than are necessary or practical for a simulation. In the current work, unwanted details may be removed by deleting the surfaces that represent the unnecessary feature. Any holes that were part of the detail can be removed by deleting trim curves de£ning the hole, or by adding new surfaces covering the hole. For example, the junction between the manifold and the valve in Figure 4 was simpli£ed by removing some patches around the junction, extending the valve stem, and re-trimming the surfaces on the manifold. Note that this modi£cation is relatively large compared to the size of the model and would therefore be hard to automate completely.

Calculating the surface connectivity

The connectivity between the surface patches in a CAD geometry is not speci£ed in the IGES £les, so it must be determined before a surface mesh can be generated across patch boundaries. To determine the connectivity, we use an edge matching algorithm [4], which attempts to match each boundary edge with a boundary edge from a neighboring patch. The algorithm begins by building three-dimensional NURBS curves on the boundaries of all surface patches by mapping all sub-curves

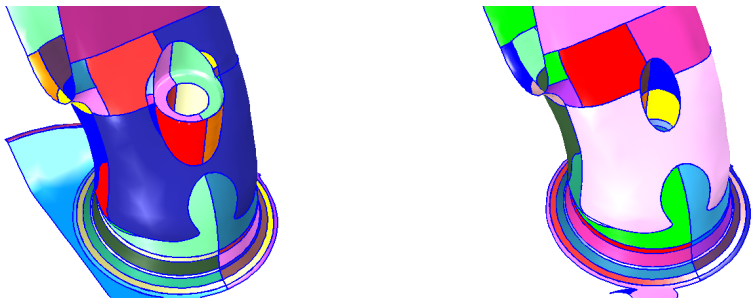

Figure 4: Removing unwanted details and simplifying the engine geometry.

in the trimming curves to R^3 (the trimming curves are de£ned in parameter space). It then attempts to identify where an boundary edge from one patch matches the boundary edge of a neighboring patch. To accomplish this, it is usually necessary to split the boundary edge at appropriate locations. When the closest neighboring boundary edge is within a tolerance of the original boundary edge, we declare that the edges have been merged and choose one of the boundary edges to be the master edge. Thereafter, the master edge de£nes the boundary segment for both patches. In this way, we effectively remove any gaps or overlaps present in the original representation.

If there are missing or duplicated surface patches in the model, the connectivity algorithm will fail to £nd matching boundary edges for those patches. The algorithm will also fail if the gap or overlap between neighboring patches exceeds the tolerance threshold. Hence, the connectivity algorithm serves as an ef£cient detector of any remaining irregularities in the model. For example, consider the detail of the engine manifold in Figure 5.

Hyperbolic grid generation

After all boundary edges have been matched, the model is triangulated such that each triangle belongs to exactly one surface patch, see Henshaw [4] for details. The triangulation is therefore well suited to provide an initial guess for projecting points onto the CAD surface during the surface grid generation. An example of a triangulation is shown in Figure 6.

The triangulation also forms the basis for the generation of embedded boundary (Cartesian) grids. Here the triangulation is used to remove grid cells outside of the domain, and to calculate volume fractions for the cells that are intersected by the triangulation. Details of this approach will be reported elsewhere.

Surface grids are generated on the geometry starting from edge or feature

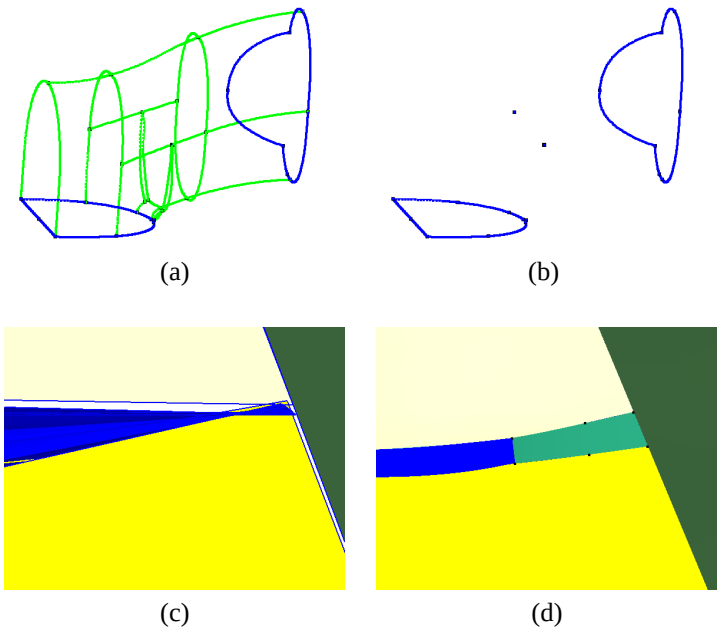

Figure 5: The edge matching algorithm colors matched edges in green and unmatched edges in blue (a). The algorithm detected two problem areas, shown as blue dots in (b). A closeup near the left dot revealed the blue warped surface (c). The warped part of the surface was removed, and a new patch was build to cover the hole (d).

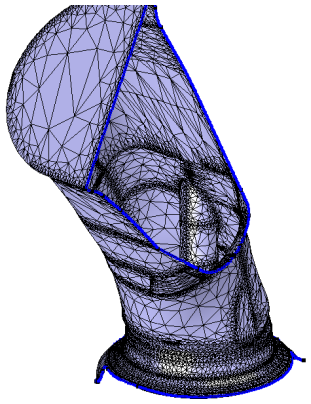

Figure 6: A triangulation of the geometry in Figure 4.

curves on the surface, see Figure 7. The starting curve forms the £rst grid line, which is marched along the surface using a hyperbolic grid generation algorithm, see Chan and Buning [2]. To ensure that the grid resides on the CAD geometry, each new grid line is projected back onto the surface. An example of a complete overlapping surface grid is shown in Figure 7.

A similar procedure is used to generate volume grids. Here the grid generation starts from a surface grid, and proceeds by marching it into the computational domain using a hyperbolic volume grid generation algorithm. When grids are grown out from a corner in the geometry we can optionally project some grid lines onto the geometry during the marching procedure. In this way, a volume grid can have up to three faces that reside on the geometry. The volume grid generation is repeated for all the surface grids in the model. Any voids that remain in the computational domain are £lled with simple, often Cartesian, background grids.

The £nal step in the overlapping grid process is to connect all component grids. This can, for instance, be accomplished by the automatic algorithms in **xCog/Chalmesh** [7] or **ogen** [3]. Our work will be made available through the next release of the **Overture** software, whose current release may be found at `http://www.llnl.gov/casc/Overture`.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

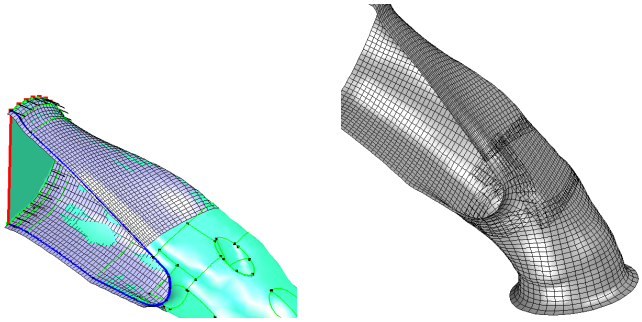

Figure 7: A surface grid is grown from the edge curve marked with green and red diamonds (left). A complete overlapping surface grid for the manifold part of the geometry in Figure 4 (right).

References

[1] G. BUTLIN AND C. STOPS, *CAD data repair*, in Proceedings, 5th International Meshing Round Table, 1996.

[2] W. CHAN AND P. BUNING, *A hyperbolic surface grid generation scheme and its applications*, paper 94-2208, AIAA, 1994.

[3] W. D. HENSHAW, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[4] W. D. HENSHAW, *An algorithm for projecting points onto a patched CAD model*, in Proceedings, 10th International Meshing Round Table, 2001.

[5] S. MERAZZI, E. A. GERTEISEN, AND A. MEZENTSEV, *A generic CAD-mesh interface*, in Proceedings, 9th International Meshing Round Table, 2000.

[6] A. A. MEZENTSEV AND T. WOEHLER, *Methods and algorithms of automated CAD repair for incremental surface meshing*, in Proceedings, 8th International Meshing Round Table, 1999.

[7] N. A. PETERSSON, *Hole-cutting for three-dimensional overlapping grids*, SIAM J. Sci. Comp., 21 (1999), pp. 646–665.

[8] N. A. PETERSSON AND K. K. CHAND, *Detecting translation errors in cad surfaces and preparing geometries for mesh generation*, in Proceedings, 10th International Meshing Round Table, 2001.

[9] L. PIEGL AND W. TILLER, *The NURBS Book*, Springer, 2nd ed., 1997.

[10] J. P. STIENBRENNER, N. J. WYMAN, AND J. R. CHAWNER, *Fast surface meshing on imperfect CAD models*, in Proceedings, 9th International Meshing Round Table, 2000.

[11] T. J. TAUTGES, *The common geometry module (CGM): A generic extensible geometry interface*, in Proceedings, 9th International Meshing Round Table, 2000.