Boolean operations for faceted solids

Ashish L. Shete
Student DACAD
ashish_shete85@rediffmail.com

Sandip N. Jadhav
Program Director (CAD)
sandip@cctech.co.in

Centre for Computational Technologies (CCTech), Pune.

Abstract:

Industrial CAD designer at various stages of product design come across faceted definition of solids due to CAD import from STL, DXF and VRML files. Most of the times the data provided by the clients or third party is sufficient for the referencing purposes. This data cannot be edited or processed to perform further geometric operations essential for downward application. This is main hindering block in main stream CAD modeling software. We are addressing this problem through faceted Boolean operations for such specific cases.

In this paper we present an algorithm to perform interactive Boolean operations on free-from faceted solids. We are developing fast surface-surface intersection algorithm for triangulated surfaces. In next stage different combinations of possible resultant solids produce from intersection operations would be displayed to user for desire Boolean selection. This enables us to add, subtract and intersect complex solids at interactive rates.

The result of an boolean operation is a set of triangles that determines the boundary of the intersection between two meshes. These triangles could be not one of the original ones. To calculate this new ones, this module search pairs of triangles that intersect and lie on the boundary. The intersection determines how to subdivide these two triangles to obtains the needed ones.

Keywords:

Boolean, solid modeling, faceted solids, triangulated surface, reverse engineering.

1. Introduction

Triangulated Faceted geometries are surface patches which are defined by points and triangles. These geometries are formed by tessellation of well defined surfaces like NURBS or B Spline surfaces. These are mainly used for visualization and rendering purposes. Due to there tessellated nature it is hard to find out there intersections. Also it is difficult to determine whether the point in space is inside the volume bounded by these surfaces, on the surfaces or outside the bounding volume. The above processes are intensively used for Boolean operations performed on solid bodies. Fast intersection and in/out searches are required in order to perform various solid parts generations.

Present work aims at developing a CAD functionality which can generate results by performing Boolean operations on faceted geometries.

2. Boolean Operations

The basic Boolean operations are analogous to mathematical set operations:

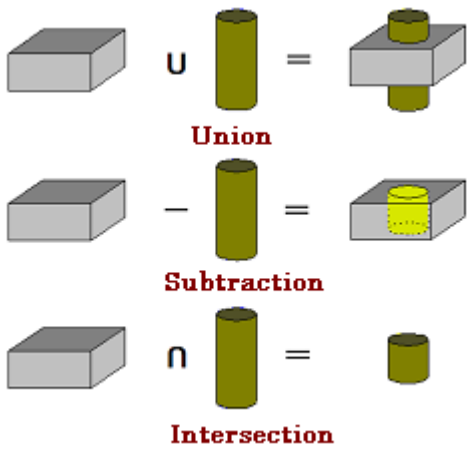


Figure 2-1 Basic Boolean operations

2.1. Global Booleans

A global Boolean involves comparing all face pairs from the target and tool bodies.

Global Booleans are performed which requires one target body, one or more tool bodies and an options structure. If more than one tool body is supplied, the union of overlapping tools is computed first and then the Boolean between the target and the tools is performed.

2.2. Local Booleans

A local Boolean involves comparing selected face pairs in the target and tool bodies. This form of Boolean operation is quicker than a global Boolean but does not guarantee topological consistency on the resulting bodies.

2.3. Boundary regions

In order to perform Boolean operations, edges are imprinted on the target and tool to denote the intersecting parts of the two bodies. These imprinted edges divide the boundaries of the bodies into boundary regions.

Suppose the bodies shown are used to perform a global and local Boolean respectively.

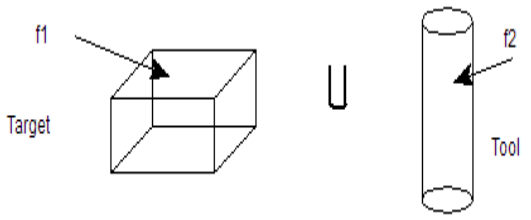


Figure 2.3.1 Bodies used in a simple union

For global Booleans, all the boundary regions of the tool bodies lie either completely inside or completely outside of the target body, as shown in Fig.

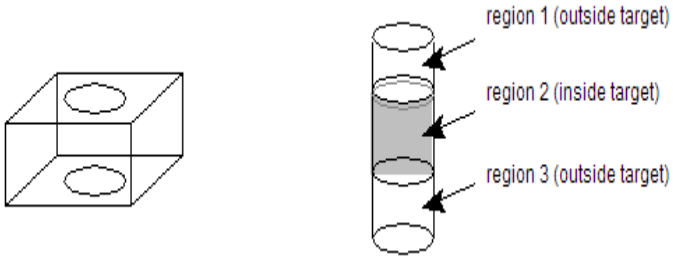


Figure 2.3.2 Regions after the boolean

For local Booleans, because not all faces of both bodies are used to compute the imprinted edges, boundary regions of the tool body are classified as inside if they are locally inside the target body near a loop of imprinted edges, and are classified as outside if they are locally outside near the loop of imprinted edges, as shown in Fig. Options are provided to select which regions are to be excluded or included in a local Boolean.

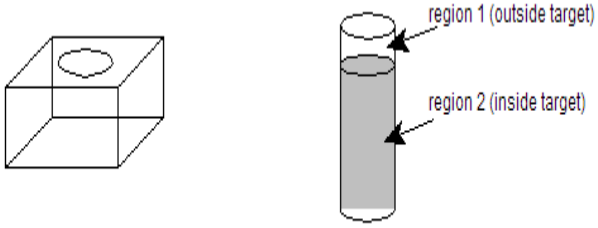


Fig. 2.3.3 Boundary regions for local Booleans

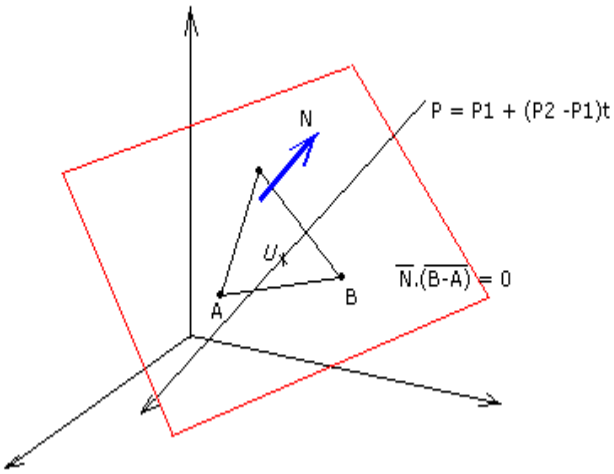
3. Imprinting

In our algorithm imprinting takes place by method explained below:

The input faceted geometries are the set of triangles. If 1st surface has *M number* of triangles and second surface has *N* triangles, each triangle in first set will try to find whether there is any intersection

between it and the second surface. This could be found out by considering each triangle of 1st surface as a set of 3 segments. These segments are actually sides of the 1st triangle. These sides will be sent as rays to the 2nd surface and the intersecting points are calculated. Different checks are applied to see the exact position of the intersecting point on the second surface. In this way M X N iterations are performed.

3.1. Ray Triangle intersection algorithm:



A plane, in its vector form, is specified by a based point B and its normal vector N. For an arbitrary point, or position vector, X on the plane, the direction vector from the base point B to X, X-B, must be perpendicular to the normal vector N. Therefore, we have (X-B).N must be zero. From (X-B).N=0, we have the equation of a plane specified with a base point and its normal vector:

X.N- B.N = 0

Given the vector notation of lines and planes, it is very easy to compute the intersection point of a line and a plane. Let the given line be A+td. Let the plane be defined with a base point B and its normal vector n. Then, this plane has equation X.n=B.n. If the line intersects the plane, there must be a value of t such that the corresponding point lies on the plane. That is, there must be a t such that the point corresponding to this t would satisfy the plane equation. Since a point on the line is A+td, plugging A+td into the plane equation yields

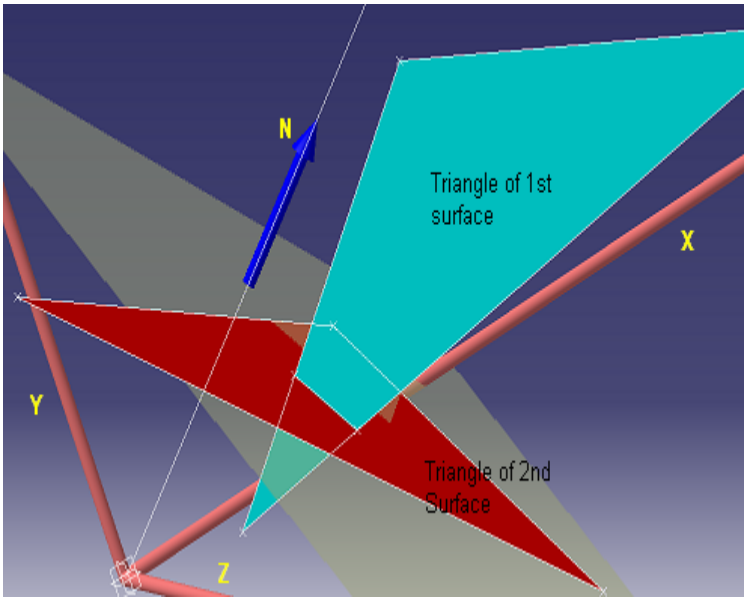
(A+td).N - B.N = 0

Rearranging the terms and solving for t yields

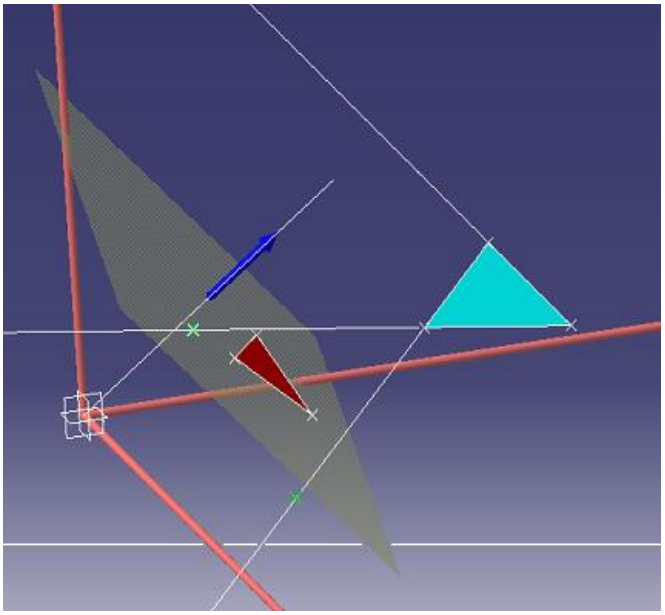
t = (B-A).N / d.N

Therefore, plugging this t into the line equation yields the intersection point.

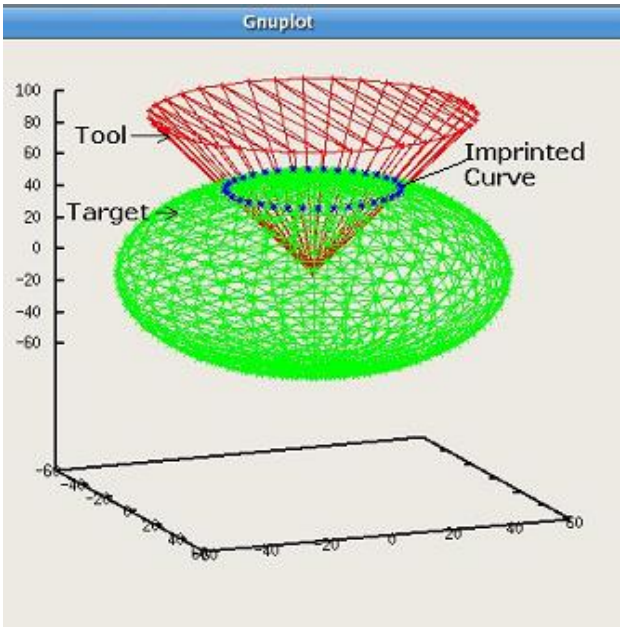
In the above, if d.N is zero, t cannot be solved and consequently no intersection point exists. The meaning of d.N = 0 is that d and n are perpendicular to each other. Since n is the normal vector of a plane and d is perpendicular to n, d must be parallel to the plane. If the line is parallel to the plane, no intersection point exists.



In some cases it may happen that the two triangles are not intersecting at all, here parameter t determines whether to consider the point of intersection or not. Point in polygon search methods are used to determine whether the point is in inside or outside of the concerned triangles.



After performing the intersection, points on surfaces are displayed which form actual boundaries, for face, which separates the surfaces from each other. The faces of the target and the tool bodies are intersected with each other to produce new edges where they meet. These edges divide the faces of each body into facesets which are either inside, outside, or on the boundary of, the other bodies.



4. Gluing

The resulting sets of faces are joined together into a single intermediate body. Triangulation is generated on these boundary points to redefine the separated facesets.

5. Selection

The parts of the model which are to be kept or rejected are selected, according to the type of Boolean being performed and the options supplied, using information gained in the earlier phases. Selection could be done by selecting a point, face, or an edge of triangle for a particular region.

We are planning to save these operated parts in separate STL files.

6. Implementation Details:

The code has been written using ANSI C[6] on Linux platform. CAD translator developed for VRML read all important geometric entities. Non triangulated entities are converted to the triangulated surfaces.

For implementation of imprinting algorithm, geometric tool library is created to support the intersection algorithm. It consists of vector algebraic operations such as vector cross product, subtraction, dot product, vector normal and unit vector. Code uses the parametric as well as implicit form of equations for various geometric entities like segments, planes, curves, triangles etc. To find the point of intersection and gluing operations are time consuming and tedious. Therefore we make use of spatial search method to increase to increase the speed of ray tracing algorithm in surface intersection. For results display purposes GNUPlot, Qt, OpenGL is used.

Spatial Searching Framework

Sequential searching of geometry is not the fastest method. Instead, division of the whole geometry domain into a grid or

voxels helps search faster as the search gets 'localized'. Voxel method of searching has its own limitations when it comes to non-uniform distribution of geometry objects in a domain. The quadtree/octree approach of domain division comes in handy to overcome this limitation of voxels as the division of the domain (and hence the problem of searching) adapts itself to local complexity of the problem.

Principle:
The grid approach to searching sub-divides the domain into rows and columns of a specified size (can be different in x, y and z directions) and marks individual voxels (cells) thus formed with the geometric objects that lie within (this local domain). The original problem, therefore, gets sub-divided into smaller problems equal in number to the number of voxels formed. When a search query is fired, the concerned voxel (the concerned sub-problem) is first identified which, if searched through, will surely obtain a solution. Only those geometric objects that have been marked earlier as lying in this voxel are searched and the solution obtained. Thus, the problem of searching all geometric objects within the entire domain gets reduced to searching only those objects that lie within the concerned local sub-domain – which is obviously faster.

However, this search may not be as fast if the distribution of objects in the original domain is non-uniform. This happens since the number of objects lying within a voxel (cell occupancy) varies across the grid with some voxels having very high occupancy and others having very low (if the query requires searching a voxel with high occupancy, the search is not sufficiently fast).

The quadtree approach starts out with dividing the domain into four (octree approach divides into eight in 3D) sub-domains. Only that sub-domain is further sub-divided into four, which has higher-than-desired cell occupancy, leaving out the subdivision of those sub-domains that need not be redundantly sub-divided further as they already have optimum occupancy. With each subdivision the quadtree/octree is said to have penetrated a level deeper. Very deep trees can render the search slower. Therefore, each subdivision decision takes into account not only the desired cell occupancy, but also the tree depth and strikes a balance between the two factors. Thus, more uniform cell occupancy is achieved although the distribution of geometry objects across the full domain may not be uniform. The quadtree division, when complete, captures the complexity of the problem in a way.

References:

[1] O'Rourke, J., "*Computational Geometry in C",* Cambridge University Press,2nd Ed 2001.

[2] Farin, G.E., "*Curves and Surfaces in CAGD*", Academic Press Inc., 4th Ed.,199

[3] Hamies, R., Aftosmis, M., "On *Generation High Quality Water-Tight Triangulation directly fro CAD"*, *"Numerical Grid Generation in Computational Field Simulations"*, The International Society of Grid Generation, 2002, pp. 27-46.

[4] Ritche, Kernigham, "*The C Programming Language*", Princeton-Hall Inc., 2nd Ed, 2000.

[5] Schneider, J.P., Eberly, H.D., "*Geometric Tools for Computer Graphic*", Morgan Kauffmann, 2003.

[6] Roger, Adams, "*Mathematical Computation for Computer Graphics*", 2nd Ed., 2001.

[7] Mezentsev, A., Woehler, T., "*Methods and Algorithm of Automated CAD Repair for Incremental Surface Meshing*", In Proceedings, *10th International Meshing Roundtable*,

Sandia National Laboraories, 2001, pp.353 – 362.

[8] 3D Systems,Inc., CA, *Stereolithography Interface Specification (STL)*, http://www.3dsystems.com

[9] AutoCAD Reference Manual, AutoDesk, Inc., *Data Exchange file format (DXF),* http://www.autodesk.com.

[10] IITZeus Preprocessor, http://www.aero.iitb.ac.in/~iitzeus