A Simple Algorithm for 2D-Mesh Generation by Domain Composition

R. Joan-Arinyo, L. Pérez-Vidal and J. Vilaplana-Pastó
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Av. Diagonal 647, 8a, E–08028 Barcelona

email: [robert, lpv, josep]@lsi.upc.es

November 11, 1998
Working Paper

Abstract

An algorithm to yield a grid meshing of the polygon resulting from the domain composition of two previously meshed polygons is presented. The algorithm is based in the step-by-step advance of one front at each side of the region where the two poligons overlap. Examples of results are also presented to illustrate the method.

Keywords: CAD, CAGD, CAM, geometric modelling, solid modelling, grid, superposition, intersection, polygon.

1 Introduction

The problem of meshing a geometric domain is central in many scientific and technical fields. Just to name some applications we have dealt with: Finite Element Method grids for subsequent computations of structures, fluid mechanics, electrical networks, height fields for Digital Elevation Models embedded in Geographical Information Systems, and surface interpolation.

In all these areas, the problem may arise of the need to use a grid that meshes a domain which can be seen as the result of some boolean operation, intersection, union and difference, of preexisting, already meshed domains.

In this paper we report on a simple but effective algorithm that generates a triangulation for a 2D domain defined as the union of two already triangulated 2D simple polygonal domains. The resulting mesh is built from triangles belonging to the initial meshes plus a small number of new triangles. Therefore, the resulting triangulation exhibits properties that are basically those of the initial triangulations.

2 The Domain Composition Approach

Mesh generation by domain composition is tightly related to the Constructive Solid Geometry (CSG) in which a 2D domain is constructed using regularized Boolean operations, [5], and the shape primitives are, for example, box, circle and wedge. In the approach, a domain is thought of as a Boolean combination of primitive shapes, each easily meshed. Then a mesh for the compound domain is computed as a combination of a number of overlapping meshes each belonging to a different primitive.

One of the basic ideas behind this approach is the fact that, for simple domains, consisting only of one shape primitive, meshes can be best defined in a natural coordinate system that simplifies both generating the mesh and solving physical problems. For example, cartesian coordinates are appropiate for box shaped domains, whereas polar coordinates are better suited for circular domains.

The approach appears to be simple and appealing but the literature on mesh generation by domain composition is scarce. Cox, [3], reports on a work oriented to generate meshes used in physical problem solving like finite elements and finite difference methods. Given a domain expressed as a CSG model, for each primitive shape a mesh is defined separately using the discretization that is most natural and the physical problem is formulated accordingly. Along the overlapped boundaries of the subdomains, compatibility is enforced by coupling equations and requiring that the solutions on the subdomains agree.

Henswah ...

In [4], Finke and Hinrichs report on an algorithm that computes the overlay of two simply connected planar meshes. The overlay consists of all non-empty sets generated by pairwise intersecting cells from each mesh. First, the algorithm computes a set of trapezoids that cover the compound domain. Then an intersection mesh is computed. The mesh nodes are the set of all intersections between pairs of edges, with each edge belonging to a different initial mesh, and an edge is stablished between two intersection points whenever they lie in two adjacent trapezoids. Finally, the initial meshes and the intersection mesh are merged.

Boender in [1] and Boender *et al.* in [2] describe an approach to automatic finite element mesh generation for CSG representations of solids with curved surfaces. Although the work subject is closely related to domain composition, the approach does not fall in this category. The method first evaluates a boundary for the CSG model. Then a tetrahedral mesh is derived from the boundary representation. It is argued that a finite element mesh derived directly from the CSG model would, in general, not be a proper discretization of the solid object. Only the availability of the domain boundary allows to precisely position vertices and edges of the cells and determine whether the cells intersect the boundary.

3 The Problem

Let P_a and P_b be two simple polygons with no holes in them that have been previously triangulated with meshes T_a and T_b, respectively. We will develop an algorithm to yield a triangulation T_u that spans the polygon P_u resulting from the boolean operation $P_a \cup P_b$.

To exclude irrelevant cases, we assume that the intersection of P_a and P_b is non-void and that neither P_a is completely included in P_b, nor viceversa. Furthermore, we assume that the intersection of the two polygons is a single connected, simple polygonal region. The intersection is bounded by the union of a connected subset of the boundary of P_a and a connected subset of the boundary of P_b. These subsets share two points, p_1 and p_2.

Under these assumptions, if $V(T)$ is the set of vertices of triangulation T, then the set of vertices of the computed triangulation will be

$$V(T_u) = V(T_s) \cup V(T_b) \cup V'$$

where V' is either $\{p_1\}$ if p_2 is a vertex of T_a and T_b, $\{p_2\}$ if p_1 is a vertex of T_a and T_b, or $\{p_1, p_2\}$ if no p_1 nor p_2 belong to the triangulations.

It is assumed that the initial meshes T_a and T_b are of a similar density. Density can be directly measured by the amount of vertices per unit area or, indirectly, by the area of the triangles. Note the intentional use of *similar* instead of *same*.

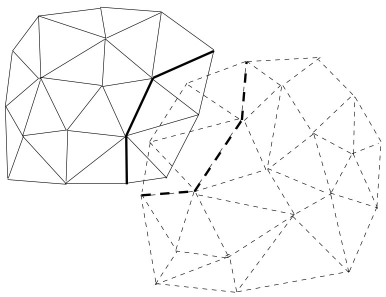

Figure 1: Maximal fronts.

4 Definitions

In this Section we give some definitions that will be used later on.

Definition 4.1 A front of edges, a front in short, in a triangulated domain is a connected sequence of edges of the triangulation such that every edge vertex is shared at most by two edges and the first and last vertices in the sequence belong to the domain boundary.

Definition 4.2 Given two triangulated domains that intersect in a single connected region, a maximal front of edges in the triangulation of a domain is a front such that separates the set of triangles covering the domain into two subsets: Those triangles whose intersection with the overlapping area is empty, called single colored triangles, and the rest. See Figure 1.

Given two triangulated domains that intersect in a single connected region, there are always two pairs of edges, each edge belonging to a different domain boundary, such that they intersect in a point. Since these edges belong to the domain boundary, they belong to only one triangle in the corresponding triangulation. We will refer to these triangles as corner triangles.

Let t_a and t_b be two intersecting corner triangles belonging to triangulations T_a and T_b, respectively. Let e_a and e_b be the edges of t_a and t_b that intersect at point p. Then, t_a and t_b can always be split into two triangles by adding the edge defined by point p and the vertex opposite to e_a and e_b, respectively. These new triangles do not belong to the initial triangulations and will be refered to as splitting triangles. Now we can state the following definitions, illustrated in Figure 2.

Definition 4.3 Given two triangulated domains that intersect in a single connected region, an augmented triangulation of one of the intersecting do-

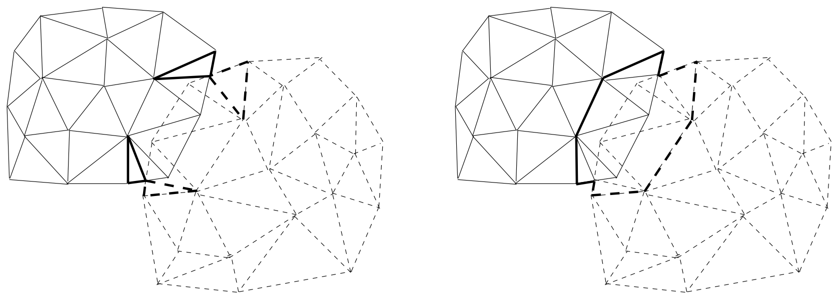

Figure 2: Left: Augmented triangulations. Right: Augmented maximal fronts.

mains is the triangulation resulting from replacing each corner triangle in the initial triangulation with the two splitting triangles.

Definition 4.4 Given two triangulated domains that intersect in a single connected region, an augmented maximal front of edges in a domain is a maximal front corresponding to an augmented triangulation.

5 The Algorithm

The algorithm is fed with two triangulated simple polygons. First the algorithm computes for each polygon its augmented maximal front. Then, triangles that have a non empty intersection with the overlapping region are included in the final triangulation as the fronts advance, one step in turn, as long as they do not collide. Finally, the seam polygon defined by the final fronts is triangulated.

5.1 Computing the Augmented Maximal Fronts

Computing an augmented maximal front starts by building a maximal front. This is performed by searching for edges belonging to triangles in a polygon two of whose vertices are strictly outside the opposing polygon and the third vertex is inside. The maximal front form a continuous frontier separating colored triangles, triangles that are completely outside, from triangles that are partly or completely inside the opposing polygon.

The augmented triangulations are computed as follows. First the corner triangles in each triangulation are determined. A corner triangle is a triangle

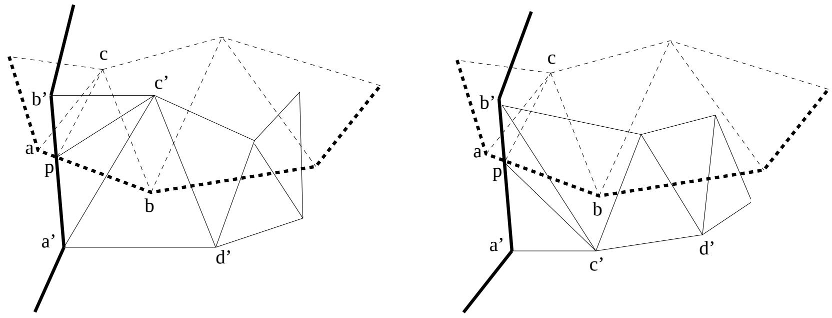

Figure 3: The two possible configurations of corner triangles.

that has one edge belonging to the outer boundary of the polygon, and, for the two vertices of this edge, one is outside, and the other inside the other polygon. That is, this edge is crossed by the outer boundary of the other polygon. Each maximal front has two corner triangles, one at each end of the chain.

For each pair of corner triangles, the algorithm computes the point where two boundary edges, one from each polygon, intersect. For each corner triangle, two new triangles, the splitting triangles, are created and each corner triangle is replaced with its splitting triangles.

Finally, to effectively build an augmented maximal front two different situations need to be considered depending on the configuration of the corner triangles. First consider the situation depicted on the left side of Figure 3. One triangulation is drawn in dotted lines, the other is drawn in solid lines and we want to compute the augmented maximal front of the triangulation in solid lines. Bold lines are the domain boundaries. The corner triangles considered here are labeled abc and $a'b'c'$. Boundary edges ab and $a'b'$ intersect at point p. Splitting triangles of corner triangle abc are apc and bpc. Analogously, for triangle $a'b'c'$, the splitting triangles are $a'p'c'$ and $b'pc'$. Note that the new edge pc' that defines the splitting triangles for triangle $a'b'c'$ has a non empty intersection with the interior of the triangle abc. It is clear that the two first edges of the desired augmented maximal front should be pa' and $a'd'$.

Now consider the situation depicted on the right side of Figure 3. Here, the new edge pc' that defines the splitting triangles for the corner triangle $a'b'c'$ has an empty intersection with the interior of the corner triangle abc and the two first edges of the desired augmented maximal front should be

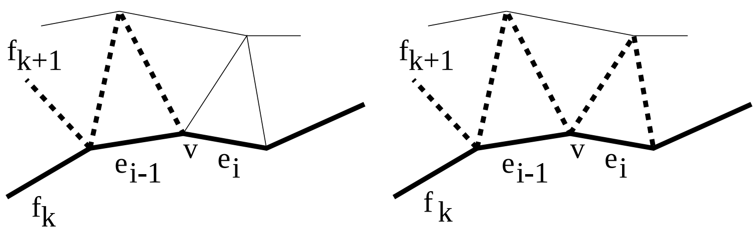

Figure 4: Vertex-sharing configuration.

now pc' and $c'd'$.

5.2 Advancing the Fronts

The strategy used to advance the fronts is simple. It consists in, for each front to advance one step in turn, so that the collision takes place more or less halfway between the positions of the two fronts at the beginning.

Let f and g be the two fronts each belonging to a different triangulated domain. Denote by f_k and g_k the two valid fronts computed in the advancing step k and assume that f_k is the front to advance next. The algorithm computes the front f_{k+1} by processing each edge in f_k sequentially. Just for the sake of clarity, assume that edges are processed from left to right and that e_i, with $i > 1$, denotes the current edge in f_k. A set of candidate edges is computed. If the intersection of the candidate edges with the region bounded by the front g_k is empty, candidate edges are added to the front f_{k+1}.

With every edge e_i of a front f_k, we associate the triangle t_i that belongs to the triangulation where f is defined and such that the vertex in t_i different from those that bound e_i points toward the current opposing front g_l. We will refer to this triangle as the advancing triangle of e_i.

Three different cases need to be distinguished according to the local configuration of the advancing triangles, t_{i-1} and t_i, associated with edges e_{i-1} and e_i in f_k, respectively. If triangles t_{i-1} and t_i just share a vertex, see Figure 4 left, the two edges of t_i different from e_i, are added to f_{k+1}, as illustrated in Figure 4 right.

If advancing triangles t_{i-1} and t_i just share an edge, e^* in Figure 5 left, the common edge is removed and the edge of t_i wich is different from e_i and from e^* is added to f_{k+1}. The updated front is shown in Figure 5 right.

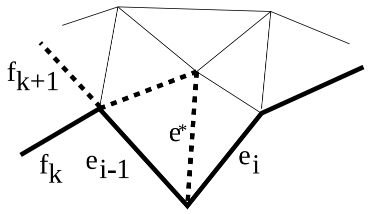

Figure 5: Edge-sharing configuration.

Let us consider now the case where t_{i-1} and t_i are coincident advancing triangles. First note that it is not possible for t_{i-2} and t_{i-1} to be also coincident because this would mean that the front f_k has three coincident consecutive advancing triangles and therefore the front would form a closed triangle. However, edge e_i has been included in f_{k+1} both when t_{i-2} and t_{i-1} share a vertex and when they share an edge. Hence, e_i is removed from f_{k+1}. See Figure 6.

Finally, the algorithm deals with the first edge, e_1, in the front f_k just considering that the advancing triangle t_1 has a vertex-sharing configuration.

After some advancing steps, the position of the fronts will be such that each front can not advance without trespassing behind the opposite front. We have, then, two chains that share their ending vertices; if we consider the circular union of these two chains, they form a polygon, called seam polygon. Figure 7 shows the fronts for the case depicted in Figure 1 and Figure 2 at their maximal forward position. The seam polygon is shown in bold lines.

5.3 Triangulating the Residual Polygon

The last step is the inner triangulation of the seam polygon. This is done with Seidel's algorithm [6].

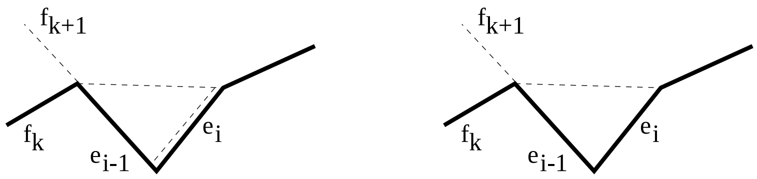

Figure 6: Triangle-sharing configuration.

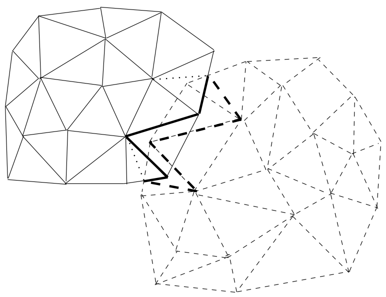

Figure 7: Final configuration of the fronts.

6 Results

To illustrate how the algorithm works, we have processed some examples. In all the examples the domain is defined as the union of a square and a polygon that approximates a circle. The meshes defined on each primitive shape are always triangulations.

6.1 Regular Triangulations

Here each subdomain has been partitioned with a regular triangulation. The two polygons with their triangles at their initial state can be seen on the left of Figure 8. On the right, the single colored triangles of each polygon, that is, the triangles that do not belong to any overlapping submesh, are shown.

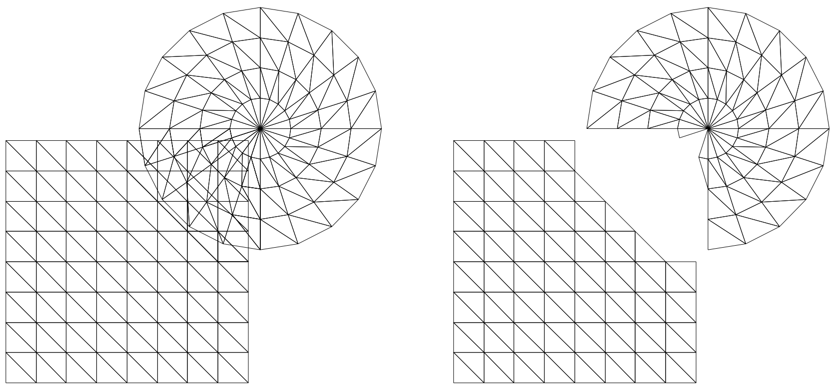

Figure 8: Initial state of the polygons and single colored triangles.

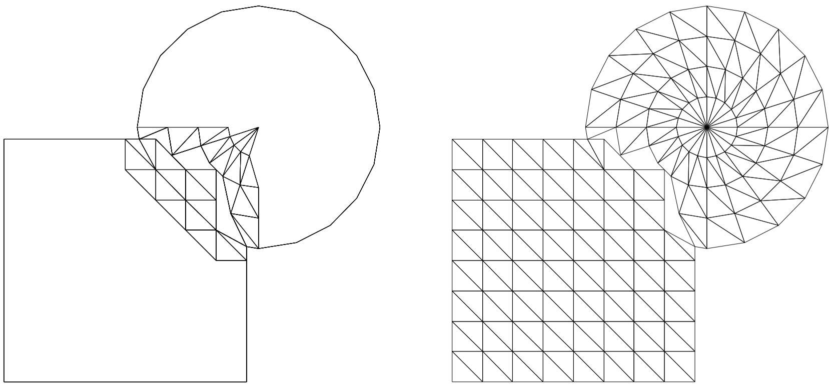

Figure 9: Triangles selected to remain and final configuration of the fronts.

Figure 9 shows on the left side the subset of overlapping triangles selected by the algorithm to be on the final triangulation. Figure 9 shows on the right side the set of triangles when the fronts cannot advance any more. The empty region bounded by the final fronts defines the seam polygon to be triangulated.

On the left, Figure 10 shows the seam polygon and the triangulation performed on it. For the sake of clarity, it is shown at a larger scale. The final triangulation is shown on the right side of Figure 10.

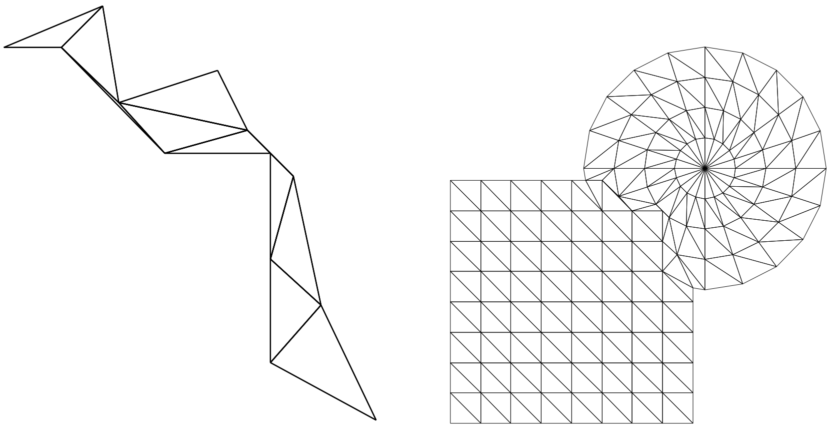

Figure 10: Triangulation of the seam polygon and final result.

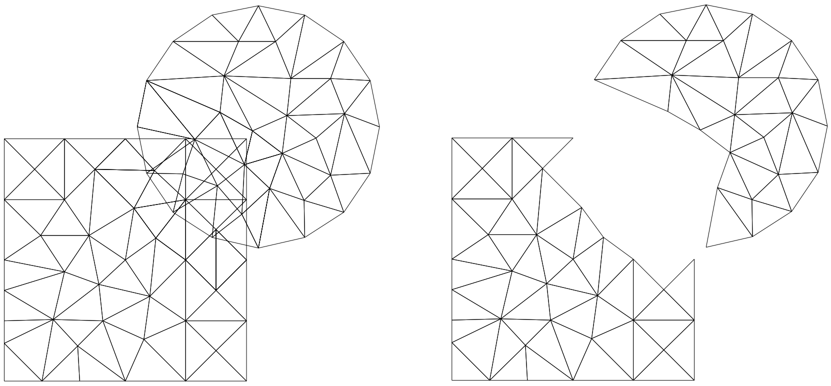

Figure 11: Initial state of the polygons and single colored triangles.

6.2 Irregular Triangulations

Althoug one of the initial motivations to develop our approach was the fact that meshes for primitive shapes can be best defined in a natural coordinate system, the resulting algorithm applies to any kind of triangular mesh. Figure 11 shows on the left the two polygons of the first example but now partitioned with irregular triangulations built by triangulating with Shewchuck's method, [7], a set of points randomly generated in each polygonal shape. The single colored triangles of each polygon are shown on the right side of Figure 11.

Figure 12 and Figure 13 show the situations corresponding to those depicted in Figure 9 and Figure 10 for the regular triangulation.

7 Conclusions

This paper introduces an algorithm to compute a triangulation to cover the polygon resulting from the union of two data polygons. Each one of the two data polygons is also previously triangulated, and the method takes advantage of the preexisting meshes. The resulting algorithm is conceptually simple, easy to implement and applies to any kind of simple polygonal domain with no holes in it.

The algorithm has been implemented to deal with triangulations but the generalization to other kind of meshes is straightforward.

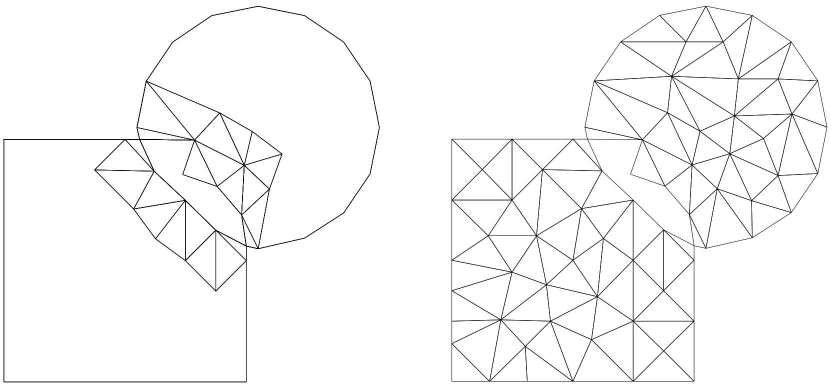

Figure 12: Triangles selected to remain and final configuration of the fronts.

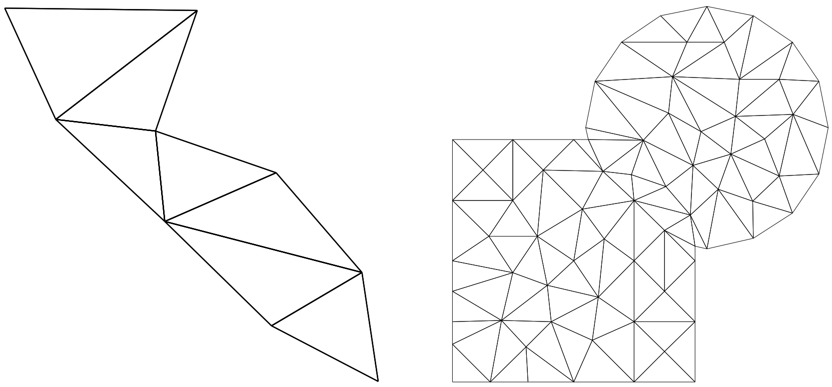

Figure 13: Triangulation of the seam polygon and final result.

Except for splitting triangles and seam polygons, the final triangulation is made up from triangles already existing in the given triangulations. Therefore, the quality of the computed triangulation is basically the same as those of the input triangulations. Splitting triangles can be very ill conditioned. This problem could be alleviated by applying a correction procedure to locally rearrange the set of edges that define the corner triangles.

No special strategy is currently applied to triangulate seam polygons. Since these polygons can be very ill conditioned, see for example the seam polygon in Figure 10, a convenient partition of seam polygons and a subsequent triangulation of each partition, could improve the quality of the final mesh.

8 Acknowledgements

This work has been partially supported by CICYT, the Spanish Research Agency, under grant number TIC 95-0630-C05-04.

References

[1] E. Boender. *Finite Element Mesh generation from CSG models.* PhD thesis, Tecnische Universiteit Delft, Dept. of Technical Mathematics and Informatics, September 1992.

[2] E. Boender, W.F. Bronsvoort, and F.H. Prost. Finite-element mesh generation from constructive-solid-geometry models. *Computer-Aided Design*, 26(5):379–392, May 1994.

[3] J. Cox. *Domain Composition Methods for Combining Geometric and Continuum Field Models.* PhD thesis, Purdue University, Dept. of Mechanical Engineering, 1991.

[4] U. Finke and K. Hinrichs. Overlaying simply connected planar subdivisions in linear time. In *Proc. 11th Annual ACM Symposioum on Computational Geometry*, pages 119–126, Vancouver, Canada, 1995.

[5] A. Requicha and H. Voelcker. Constructive solid geometry. Technical Report Memo 25, University of Rochester, Production Automation Project, 1977.

[6] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

[7] J. R. Shewchuk. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry*. Association for Computing Machinery, May 1996.