3-Level Geometric Modeler: CSG, Subdivision Trees and Boundary Representation

Kai Cao, Zhongdi Luo

New York University
kcao@cs.nyu.edu zl562@nyu.edu

Abstract

The problem of computing geometry modeling in $\mathbb{R}^3$ is fundamental to computer graphics, path finding, motion planning and numerous branches of computer science. Given a polyhedron P defined by constructive solid geometry (CSG) model, we raise a new idea to present the boundary features (vertices, edges and faces) of P using subdivision method; it has the following steps: 1) define P in CSG model, 2) subdivide a given box B_0 for P and construct the adequate subdivision tree T, 3) represent the boundary features based on the subdivision tree T. We could also compute the boolean operations of 2 subdivision trees. Since the great generality and scalability of subdivision method, this idea can also be expanded to support more complex shape with higher order surfaces in $\mathbb{R}^3$.

I. Introduction

In this paper, we focus on the geometric modeling of solids (especially polyhedra) in $\mathbb{R}^3$. The traditional definition of a polyhedron in $\mathbb{R}^3$ is the boolean union of half spaces (CSG). The faces of a polyhedron P are the plains which bounds the half spaces. Meanwhile the edges are the intersection of two unparallel plains and the vertices are the intersection of three. Faces, edges and vertices constitute the features of a polyhedron. By CSG definition, these features are not obvious. To present them more efficiently and make them more operable, we keep subdividing a box B_0 containing P until the boundary features are kept in certain subboxes. Then we can compute the features by manipulating the subdivision tree instead.

II. Regularized set

A set $S \subset \mathbb{R}^3$ is a **regularized set** if $S = \overline{int(S)}$.

A **solid** is defined as a regularized subset of $\mathbb{R}^3$. Obviously a polyhedron P is a finite solid.

Suppose A and B are regularized sets, we define the boolean operations:

- $A \cup^* B = \overline{int(A \cup B)}$
- $A \cap^* B = \overline{int(A \cap B)}$
- $A \setminus^* B = \overline{int(A \setminus B)}$
- $A^{c*}B = \overline{int(A^c)}$

III. CSG primitives

A **constructive solid geometry** (CSG) model describes a polyhedron as regularized set operation on primitives which are the simplest solid objects used for the representation.

Primitives in CSG models are all semi-algebraic sets as follows:

1. **Half Space**: $H = \{(x, y, z) | a * x + b * y + c * z + d \geq 0\}$.
2. **Sphere** : $S = \{(x, y, z) | (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 \leq 0\}$.

3. **Cylinder** : $C = \{(x,y,z)|A = -x\sin(\theta)+y\cos(\theta)\cos(\phi)+z\cos(\theta)\sin(\phi), B = -y\sin(\phi)+z\cos(\phi), A^2+B^2 \leq R^2\}$.

4. **Torus** : $T = \{(x,y,z)|(R-\sqrt{x^2+y^2})^2 + z^2 \leq r^2\}$.

A polyhedron is limited by the intersection of half spaces.

IV. Subdivision tree

Subdivision tree is an octree which is very efficient to maintain the features of a polyhedron in a hierarchical structure.

4.1 Box

B is a **box** if $B \subset \mathbb{R}^3$, $B = \{(x,y,z)|x_1 \leq x \leq x_2, y_1 \leq y \leq y_2, z_1 \leq z \leq z_2\}$, or simplified as $B = ([x_1,x_2],[y_1,y_2],[z_1,z_2])$, which is a regularized subset of $\mathbb{R}^3$.

4.2 Subdivision on box

Suppose $B = ([x_1,x_2],[y_1,y_2],[z_1,z_2])$, let $\bar{x} = \frac{x_1+x_2}{2}$, $\bar{y} = \frac{y_1+y_2}{2}$, $\bar{z} = \frac{z_1+z_2}{2}$. The subdivision on B is to subdivide B into eight subboxes by splitting at $x = \bar{x}$, $y = \bar{y}$ and $z = \bar{z}$.

4.3 Features

For a polyhedron $P \subset \mathbb{R}^3$, we use the sets V, E and F to represent its vertices, edges and faces respectively. Let

- $\Phi_0(P) := V$
- $\Phi_1(P) := E$
- $\Phi_2(P) := F$

and

- $\Phi(P) := \bigcup_{i=1}^{2} \Phi_i(P)$

Elements in $\Phi(P)$ are called **features**. Also let $\Phi_i(P,B)$ denote the set $\{f \in \Phi_i(p) : f \cup B \neq \emptyset \text{ and } \Phi_i(P,B) = \bigcup_{i=1}^{2} \Phi_i(P,B)\}$

4.4 Elementary Box

We call a box B to be **elementary** for a polyhedron P if one of the following conditions hold:

1. $B \cap P = \emptyset$.

   Call B an **empty box** for P.

2. $B \subseteq P$.

   Call B a **full box** for P.

3. Exists only one v st. $\Phi_0(B,P) = \{v\}$ and $\forall e \in \Phi_1(B,P)$ e is bounded by v and $\forall f \in \Phi_2(B,P)$ f is bounded by v.

   Call B a **vertex box** for P.

4. $V \cap B = \emptyset$, exists only one e st. $\Phi_1(B,P) = \{e\}$ and $\forall f \in \Phi_2(B,P)$ f is bounded by v.

   Call B a **edge box** for P.

5. $V \cap B = \emptyset$, $E \cap B = \emptyset$, exists only one f st. $\Phi_2(B,P) = \{f\}$.

   Call B a **face box** for P.

If none of the above conditions holds, we call the box **mixed box**.

4.5 Adequate subdivision tree

1. A **subdivision tree** is a tree where each interior node has degree 8 (octree) and each node u is associated with a box $u.box$ such that if $u_0,\ldots,u_7$ are the children of u. Then $\{u.box : i = 0,\ldots,7\}$ is the split of $u.box$.

2. An **adequate** subdivision tree for a box B_0 and a polyhedron P is a subdivision tree rooted at B_0, in which all leaves are associated with elementary boxes for P.

V. Boolean Operations on subdivision trees

Suppose we already have $\Phi(P, B)$ and $\Phi(P', B)$ for two polyhedra P, P'. We now address the question of computing $\Phi(Q, B)$ where Q is the union or intersection of P and P'.

5.1 Make two trees compatible

Give box B_0 and polyhedron P_1, P_2. We have two adequate subdivision trees T_1 for P_1, T_2 for P_2.

We say T_1, T_2 are congruent if they are isomorphic. Our approach is to organize $\Phi(B, P)$ using a subdivision tree that is adequate for B and P.

The process of **Make Compatible**(T_1, T_2) using BFS:

Input: T_1, T_2 are two adequate subdivision trees for P_1, P_2 respectively and both rooted at B_0.
Output: The modified T_1 and T_2 which are compatible.

```
n1 ← T1.root.
n2 ← T2.root.
insert n1 into queue Q1, n2 into queue Q2.
while Q1 ≠ ∅ and Q2 ≠ ∅ do:
    n1 ← Q1.dequeue().
    n2 ← Q2.dequeue().
    if (n1.box is elementary for T1
          and n2.box is elementary for T2)
          then
        continue.
    if (n1 is a leaf) then
        T1.expand(n1).
    elif (n2 is a leaf) then
        T2.expand(n1).
    Q1.enqueue(n1.children).
    Q2.enqueue(n2.children).
```

Note that the termination of this algorithm is obvious.
Observe that suppose B' is a subbox of B, P is a polyhedron, $\Phi\ (B', P) \subseteq \Phi\ (B, P)$.

Lemma 1 *Given a box B and a polyhedron P, we get its adequate subdivision tree in limited subdivisions.*

proof. We get its adequate subdivision tree, that is to say, we would get a state that all **leaf** subboxes of B are elementary for P.

We define $d(x, y)$ to be Euclidean distance between x and y (x, y can be vertex, edge or face). Given a subbox B'.

Therefore we have proved that during subdivision we would split all the unrelated features into different subboxes. *Lemma* 2 holds.

Furthermore, we consider using the process above to compute $\Phi\ (B', P_1) \cup \Phi\ (B', P_2)$. We

would get an adequate subdivision tree T_3 for both P_1 and P_2. When we extend both T_1 and T_2 to T_3, we make them compatible. We proved

Lemma 2 *Given a Box B, its adequate subdivision tree T_1 for polyhedron P_1 and adequate subdivision tree T_2 for polyhedron P_2, we even T_1 and T_2 up in limited steps.*

Now we have two congruent adequate subdivision trees and then can step forward to the boolean operations.

5.2 Boolean operations

When we do boolean operations on two trees, we care about the leaves since they show the boundary condition. For simplification, we turn the boolean operations on trees to the same operations on two corresponding nodes (corresponding means two nodes are at the same position of their congruent trees).

Suppose n_1, n_2 are two corresponding elementary leaf nodes of T_1, T_2 for P_1, P_2 respectively. Each of them belongs to one of the five types: empty box (EMB), full box (FUB), vertex box (VB), edge box (EDB), face box (FAB). We have the following rules (left operand is of n_1, right operand is of n_2):

5.2.1 Empty boxes

- $EMB \cup B^* = B^*$
- $EMB \cap B^* = EMB$
- $EMB - B^* = EMB$
- $B^* - EMB = B^*$

5.2.2 Full boxes

- $FUB \cup B^* = FUB$
- $FUB \cap B^* = B^*$
- $\begin{cases} FUB - FUB = EMB \\ FUB - VB = VB \\ FUB - EDB = EDB \\ FUB - FAB = FAB \\ FUB - EMB = FUB \end{cases}$
- $B^* - FUB = EMB$

5.2.3 Vertex boxes

- $VB_1 \cup VB_2$
  If VB_1 and VB_2 share the same vertex, the result is a vertex box, otherwise the result is a mixed box (MB) which needs further subdivision.
- $VB \cup EDB$
  If $VB \cup P_1 \subset EDB \cup P_2$, the result is a edge box, else if $EDB \cup P_2 \subset VB \cup P_1$, the result is a vertex box, else if the vertex of VB bounds the edge of EDB, the result is a vertex box, otherwise the result is a mixed box.
- $VB \cup FAB$
  If $VB \cup P_1 \subset FAB \cup P_2$, the result is a face box, else if $FAB \cup P_2 \subset VB \cup P_1$, the result is a vertex box, else if the vertex of VB bounds the face of FAB, the result is a vertex box, otherwise the result is a mixed box.
- $VB_1 \cap VB_2$
  Todo
- $VB \cap EDB$
  Todo
- $VB \cap FAB$
  Todo
- $VB_1 - VB_2$
  Todo
- $VB - EDB$
  Todo
- $VB - FAB$
  Todo

- $EDB - VB$
  Todo
- $FAB - VB$
  Todo

5.2.4 Edge boxes

- $EDB_1 \cup EDB_2$
  Todo
- $EDB \cup FAB$
  Todo
- $EDB_1 \cap EDB_2$
  Todo
- $EDB \cap FAB$
  Todo
- $EDB_1 - EDB_2$
  Todo
- $EDB - FAB$
  Todo
- $FAB - EDB$
  Todo

5.2.5 Edge boxes

- $FAB \cup FAB$
  Todo
- $FAB \cap FAB$
  Todo
- $FAB - FAB$
  Todo