ABSTRACT

Feature-based design has evolved as a fundamental paradigm for present-day CAD systems. Voxel-based modeling has many advantages over traditional representation schemes but lacks structural information which is lost in voxelization of the objects. In this paper, we propose to augment voxel models with features by a) storing the sequence of modeling operations along with voxel models and b) by defining feature operators. We have implemented these ideas in our prototype system Sirpi as regularised Minkowski operators using simple data structures and algorithms.

1 INTRODUCTION

Two modeling technologies are at the vanguard of the next generation of computer-aided mechanical design tools. These are feature-based modeling and parametric or constraint-based modeling:

Features: A representation that flags geometric features of a part can help identify manufacturing processes that match with particular features. Such hierarchical representation schemes have come to be called feature-based. Their need grew out of the fact B-Rep and CSG representations, currently the dominant schemes, are too low-level for a designer to work with. Feature-based schemes are thus an expedient interface between CAD and analysis or manufacturing systems today (Rossignac, 1990). Representing products in terms of features helps organize the design as well as the manufacturing data repositories. In design, features are helpful in capturing designer's intention, explicit engineering attributes, and relationships among entities which are essential for various design related tasks and analysis (Mantyla, 1996). Feature-based models in manufacturing help to define process models and resource models which lead to the creation of knowledge repositories for manufacturing that aid in data reuse in new product design, product development and quality management.

Parametric modeling: Parametric representation of solid models have made tremendous impact because of the ease with which a user can edit the model interactively. Editability of the modeling scheme is one important criterion for evaluating a modeling system. Editability encompasses both local changes as well as global changes in the model. The major underlying modeling paradigms of present day CAD systems, namely CSG, B-Rep and non-manifold are surface based and represent models using surface data alone. Of these three, B-Rep is currently the most widely used modeling paradigm mainly because it is possible to incorporate parametric and constraints in B-Rep models. This popularity of B-Rep is in spite of the fact that surface based representation schemes have serious deficiency from the perspective of support for feature modeling (Pratt, 1988).

In addition, the next generation manufacturing technologies will need new paradigms in geometric modeling. Keeping these drawbacks of present day CAD systems, Chandru et. al. (Chandru, 1997) present arguments to support the view that voxel-based modeling, one of volume representation schemes, has the attributes that make it the representation scheme of choice in meeting the requirements of emerging manufacturing systems. Further, Chandru et.al. (Chandru, 1995) describe the reasons why
Voxel-based Modeling

Voxel models, Spatial Occupancy Enumeration or Cellular Modelers use a very simple representation scheme. A volume of space (containing the object) is divided into a large number of cuboidal cells. The system then labels each cell as to whether or not it is occupied by material. The obvious data structure for this representation is a 3D array of 0’s and 1’s. More sophisticated data structures based on quadtree and octree representations are possible.

Voxel-based modeling enjoys the following advantages over other representation schemes (Chandru, 1997):

- Excellent support for heterogeneity: Traditional CAD tools available today implicitly assume that the part being designed is to be fabricated in a single homogeneous material. Voxel representation can model inhomogeneities of the following types: use of different materials, use of different densities of the same material across the cross section of the solid, use of hollow structures (e.g., honeycomb) and integrated electromechanical components.
- Insensitivity to object complexity and topology.
- Ease of implementation of boolean operations.
- Excellent morphological dexterity: Voxel models can represent shapes of arbitrary complexity and support complex shape modifications.
- Excellent local editability: The central purpose of any modeling paradigm is to empower the user to effect changes to the model. We intend the broad term editability to encompass both local changes as well as global changes in the model. Examples of global changes are scaling or shearing of the entire part in a single operation. Voxel models have very good local editability and permit the easy implementation of certain global operations (analogous to filtering) that alter the entire model.

Voxel models while enjoying these many virtues suffer from three serious deficiencies:

1. There is no notion of features in a voxel model.
2. There is no current method by which a voxel model could be parameterized. Without the ability of support features or parameterization, constraint based design is currently not possible in voxel modelers.
3. Rendering complexity, high memory requirement and the problems related to discretization are the other major problems associated with the voxel models. However, the volume graphics community (VG, 1999) is actively pursuing these problems and therefore will not be considered further in this paper.

Voxel models have not been able to play a significant role in solid modeling because of these limitations. However, feature-based modeling exists at a higher level and can, in principle, be equally well integrated with any of the B-Rep, CSG, Voxel or Non-Manifold volume representation schemes. Thus incorporating an expanded notion of features without disturbing the advantages of voxel models will be the focus of this paper.

In this manuscript we propose data-structures and algorithms for incorporating features in voxel models using regularised Minkowski operators. We have implemented these ideas in our prototype system Sirpi<sup>2</sup> (Sculpting Interface for Rapid Prototyping).

2 Sirpi means sculptor in the Dravidian language Tamil

2 RELATED WORK

To our knowledge, there is no literature on features in voxel models. However, there are some related ideas that we review here.

A naive approach to voxel features would be to tag voxels belonging to each structure/feature. This is memory intensive and hence usually discarded. Brian et. al., (Brian, 1990) discuss a method of constructing hashing functions for voxel models. A table of pointers is maintained to various features in a model. The assignment of voxels to table element is decided by the hashing function. Defining a hashing function is not always possible. It assumes that many of the voxels in grid are empty and the level of details of the grid (in case if it is octree) to be known a priori.

In solid modeling, feature recognition has been approached in two ways namely, graph-based methods and volume-based methods ( Mantyla, 1996). In volume based methods there are two popular approaches, cellular decomposition and trace-based methods. Cellular decomposition partitions the delta volume into convex cells, then combines the cells into manufacturing features. Shah et. al., (Shah, 1990) deal with decomposing a machining volume into minimum convex cells using half plane partitioning method and a cell adjacency graph is used to represent the relation or constraints between the convex cells. In voxel modeling, a voxel itself is a minimum convex cell, but, the approach is not appropriate for voxel models since if we were to use a cell-adjacency graph to represent the features, the space complexity would be more than that of the model itself. Trace-based approaches look for patterns of faces and edges in the delta volume that might have been produced by various types of features. Thus these volume based approaches...
for feature recognition are really based on surface representations and hence not suitable for voxel-based modeling.

In image synthesis, block operations perform transformations of group of voxels (Kaufman, 1993). The volume graphics community considers volume deformation and voxel mapping applied to blocks of voxels as the means to expand the modeling capabilities of voxel models (Prakash, 1999; Kaufman, 1998). Marching cube algorithms (Lorensen, 1987) are used to extract surfaces of isosurface grid points, which are considered as features in volume graphics. For morphing, Lerios et al (Lerios, 1995) consider features in a volume model extracted by marching cubes. The image processing community considers feature extraction in 3D pictures using connected component labeling algorithms. In their work, the notion of feature is used in a narrow sense that features refer to volume, moments, isosurface or extreme points (Thurjell, 1992). Since, 3D pictures are also 3D grid values it is related to our work.

Medical imaging community refers segmenting a voxel data based on position (spatial segmentation) and/or features (density segmentation). Spatial segmentation separates out regions which could be defined geometrically or interactively by the user. Density segmentation isolates parts of the volume data based upon its densities, tone-scales or colors.

Features, to a mechanical designer, usually represent partial geometries that provide the link between a geometric model and its process plan for manufacture. Such higher level features of voxel models have not been considered in the literature possibly because voxel models do not explicitly provide the requisite geometric information. Our approach is to use high level primitives for sculpting through set operations called regularised Minkowski operators. These primitives include any volume that can be generated by a single sweep operation (for example, a primitive machined volume). These Minkowski operations are recorded and used as features of the sculpted voxel model. The next section describes the details.

3 VOXEL FEATURES

Features expressed in voxel models can exhibit far more complexity than features in traditional solid modeling paradigms. For example,

- 3D tolerancing (interference volume) can be exactly quantified, and hence used to solve problems of assemblability and disassembleability in a shorter time.

Conversely, voxel models are enriched when annotated with features, for example,

- editability of the voxel models improves. Parameterized features in voxel models will accelerate the modeling process.
- the hierarchical nature of features and other object oriented advantages can be used to facilitate volume modeling.

Selective editing in voxel models has not been addressed in the literature. Once, our goal of augmenting voxel models with features is achieved, selective editing will become feasible. Another advantage of the annotation of high level features in voxel models is that the animation of voxel volumes can be achieved with increased realism (Chandru, 1999). Voxel features also enable the specification and use of constraints in the context of voxel based modeling.

4 OUR APPROACH

We propose to incorporate features in voxel models by

- storing the sequence of operations along with the voxel model
- by defining feature operators which will operate on features.

4.1 Sequence of Operations

In solid modeling many problems of feature recognition still remain unsolved and are likely to remain so. These problems arise due to the overlapping of features, since intersecting volume information is not explicitly available from surface representations such as B-Rep. The following is a quote from (Rappoport, 1995).

"...unless all original models and the sequence of operations are stored then any modeling scheme with some sophistication in its modeling space would be lossy..."

This point of view is supported by many researchers (Hoffmann, 1993).

4.2 Feature Operators

Instead of forcing a voxel model to store all the feature information, we introduce notions like reference counts which mark the presence of features. Feature operators are used for actually extracting the features from the voxel model.
7 ALGORITHMS FOR REGULARISED MINKOWSKI OPERATORS

The algorithms described in this section assume the voxel representation of the objects.

7.1 Naive approach

A naive algorithm would first create a feature using Minkowski operator, then compute its intersection with the existing features, and finally compute the set difference of the corresponding voxel sets.

The pseudo code is given below.

```c
RegularisedMSum(tool, toolPath, listOfFeatures)
{
    rMSum = RegularisedMSum(tool, toolPath, NULL);
    foreach(feature in listOfFeature)
    {
        rMSum = rMSum - intersection(rMSum, feature);
    }
}
```

Using an $O(N^3)$ algorithm for computing the Minkowski sum, this algorithm would take $O(rN^3)$ where $r$ is the number of features.

7.2 The Reference Count approach

The naive approach can be improved to an $O(N^3)$ as described below.

A counter called reference count is maintained for each voxel as shown in Figure 5. When a voxel is referred to by a feature its reference count is incremented. When the Minkowski operation is that of subtraction, the reference count is decremented. Thus,

regularised $\bar{f}^b$ feature =
all the voxels belong to $\bar{f}^b$ feature -
voxels having reference count more than one. (4)

A regularised Minkowski operator can now be implemented using the following procedures.

```c
incrementRefCount(pt)
{
    refCount ++;
}

decrementRefCount(pt)
{
    refCount --;
    if(refCount <= 0)
    {
        refCount = 0;
        setOff(pt);
    }
}
```

Note that the improvement in complexity has been achieved by avoiding the explicit computation of all pairwise intersections.

7.3 The Stack-of-Bits approach

In the Sirpi environment, the user can either start with the full clay (we call this approach additive sculpting), or with null clay (we call this approach subtractive sculpting).

Reference count approach will not allow this freedom to the user, i.e., additive and subtractive sculpting can not
etc (Kaufman, 1993). Different algorithms for implementing Minkowski operators in voxel representation are detailed in (Saurabh, 1998).

Cutting and pasting operations are implemented using Minkowski sum of the tool along its trajectory. We approximate the trajectory by a polyline. Hence, the task reduces to finding the Minkowski sum of the tool with a straight line segment.

Let $S$ be the volume swept by the tool along the straight line segment, and $C$ be the clay. Then, in set theoretic terms,

$$\text{Result} = \begin{cases} C - S & \text{for cut operation} \\ C \cup S & \text{for paste operation} \end{cases}$$

(1)

The following lemma gives an algorithm to calculate the swept volume by considering boundary voxels that lie along the sweep line alone.

$$A \oplus L = (A_s \bigcup \left( \cup_{v \in \delta A} v \oplus L \right))$$

(2)

where $L$ sweep line is the line segment with end points $s$ and $t$, $A$ is the tool, $A_s$ is the tool placed at $s$ and $\delta A$ is the boundary voxels visible to the sweep line. Since number of voxels in $\delta A$ is $O(N^2)$, the time complexity of the algorithm is $O(N^3)$.

### 6 FEATURE BASED SIRPI

We consider the result of each Minkowski operation to be a voxel feature. Thus our notion of a feature is much more general than the traditional feature. Minkowski operators can be used to describe traditional features. Some of the examples are shown in Figures 2, and 3.

A voxel model created by a sequence of Minkowski operations is called a Minkowski model. We give a comparison between a Minkowski model and a voxel model below.

<table>
<thead>
<tr>
<th>Voxel Model</th>
<th>Minkowski Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definite Model</td>
<td>Procedural Model</td>
</tr>
<tr>
<td>Explicit Model</td>
<td>Implicit Model</td>
</tr>
<tr>
<td>Memory intensive</td>
<td>Concise</td>
</tr>
</tbody>
</table>

The two models are complementary. So, we see advantage in storing the Minkowski model along with voxel model.

Regularised Minkowski Operators

We introduce a new notion of a regularised Minkowski operation as follows:

Suppose there are $n$ features in the model. A $n+1^{th}$ regularised feature ($rf_{n+1}$) is defined as follows:

$$f_{n+1} = A \oplus L$$
$$rf_{n+1} = f_{n+1} - \bigcup_{i \in n} (f_i \cap f_{n+1})$$

(3)

The result of regularised Minkowski operators will be regularised features. A regularised feature is nothing but, the feature itself except the intersecting volumes of other features. Regularised features thus remember the other interacting features. Note that a similar regularised feature can be used in Minkowski decomposition. An example of regularised Minkowski addition is shown in the Figure 4.

The notion of regularised Minkowski operators is not new in the literature. Menon et. al. (Menon, 1994) use regularised Minkowski operators in the same sense as used in regularised set operators for CSG modeling (i.e., to remove dangling edges and faces). Our regularisation of Minkowski operators is with respect to features.
be mixed. The algorithm described in this section provides this flexibility.

The approach is again simple. Each voxel has a stack, where bits (for binary voxels, for heterogeneous environment more bits per voxel has to be allocated) can be pushed or popped. Whenever there is request for change of the voxel state, that voxel tries to store the old state in a sequence of states by pushing the old state in the stack.

Unlike the reference count approach, this approach follows the sequence strictly. A change in sequence is simply not allowed, since if allowed, the result would not be the regularised Minkowski summation or decomposition.

Sirpi implementation of stack-of-bits approach supports to modes of sculpting, feature mode and edit mode. In the feature mode, the voxel state is pushed onto the stack. In the edit mode the voxel stack is popped to bring the voxel to its preceding state.

Fig 6 illustrates the stack-of-bits approach with two simple features of two voxels each along with the voxel stacks after the following sequence of sculpting operations.

1. Feature 1 is pasted in feature mode (Fig 6 (a)).
2. Feature 2 is pasted in feature mode (Fig 6 (b)).
3. Feature 2 is cut in edit mode (Fig 6 (c)).
4. Feature 2 is pasted at a different place in feature mode (Fig 6 (d)).

8 AN EXAMPLE

The modified outline of Sirpi is shown in Figure 7.

We have sculpted a chair in Sirpi using regularised Minkowski operators, as shown in Figure 8 through Figure 13. The handle support feature of the chair is chosen and edited to change its orientation. It should be noted that when the support was first removed, the voxels belong to the chair seat were not removed. This would not be possible in a pure voxel model.

9 CONCLUSION AND FUTURE WORK

We have incorporated features in voxel-based models using regularised Minkowski operations. We have also introduced the notion of a Minkowski model as a powerful complement to a voxel model. Sirpi, our prototype virtual machining system is now capable of creating and editing features in voxel models as demonstrated by the examples.

Sirpi provides an interface which facilitates selective editing of voxel features.

The next research challenge is to introduce constraint-based modeling on top of voxel features. We believe that this combination will propel voxel-based modeling as the leading volume modeling paradigm.
5 SIRPI

Sirpi is an implementation of our interactive sculpting system. It uses Interactive Virtual Machining (IVM) as an interface to interactive sculpting. Sirpi is based on voxel modeling approach (Mahesh, 1998). Our work in interactive sculpting is in the context of a geometric modeling framework for rapid prototyping (RP).

5.1 Interactive Sculpting

Interactive sculpting is the process by which a designer makes free-form changes on a volume model interactively. It is the extension of 2D paintbrush programs to 3D. IVM is a conceptual interface to sculpting and our unique approach to interactive sculpting. Simple milling, contour milling, turning, thread cutting and surface machining are the IVM tools Sirpi implements. The tool itself can be a voxel model, also the Sirpi sculpted models itself can be used as a tool and thus expanding Sirpi’s design space. Outline of Sirpi is shown in Figure. 1.

5.2 Minkowski Operators

Minkowski operators are shape operators using which we can design almost all real-world objects (Ghosh, 1988). IVM is implemented using Minkowski operators for cutting (or pasting) day along a path using a tool. A brief description of these operators and their role in Sirpi are given below.

Minkowski addition of two sets $A$ and $B$ in $\mathbb{R}^d$ is defined as the union of sets obtained by positioning one of them, say $B$, at every point of the other, say $A$ and vice versa, i.e., the set of points obtained by vectorially adding each point in $A$ with each point in $B$.

Mathematically, if $A_p$ denotes the translate of a set $A$ by the vector $p$, i.e., $A_p = A \oplus \{p\}$, then,

$$A \oplus B = B \oplus A = \bigcup_{a \in A} B_a = \bigcup_{b \in B} A_b$$

which is same as,

$$A \oplus B = \{a + b : a \in A, b \in B\}$$

where $\oplus$ stands for Minkowski addition.

Minkowski decomposition of two sets $A$ and $B$ in $\mathbb{R}^d$ is defined as

$$A \ominus B = \bigcap_{b \in B} A_{-b} = \bigcap_{b \in B} A_b$$

where $\ominus$ stands for Minkowski decomposition operation. The set $B' = \{-b : b \in B\}$ is generally known as the symmetrical set of $B$ with respect to the origin.

Computation of Minkowski operations when both the objects are general polyhedra is hard. Implementing Minkowski operators in B-Rep restricts the two object to be polyhedrons. The problem of finding algorithms for multiply connected B-Rep objects is still open. We avoid these problems by taking a voxel representation for both the tool and the day. Implementing Minkowski operators in voxel representation has special advantages such as topology insensitivity, trivial algorithms for boolean operations.
REFERENCES


Figure 6. Stack of Bits in action
Figure 7. Features in Sirpi using regularised Minkowski operators
Figure 8. The sculpted chair

Figure 9. Close-up view of the chair

Figure 10. Back view of the chair

Figure 11. Positioning the tool for editing

Figure 12. After cutting the old feature

Figure 13. After re-parameterization of the chair