Proceedings of ASME 2000 IDETC/CIE
2000 ASME Design Automation Conference
September 10-13, 2000, Baltimore, Maryland, USA

DETC2000/DAC-14278

ALGORITHMS FOR DESIGN AND INTERROGATION OF FUNCTIONALLY GRADIENT MATERIAL OBJECTS

Hongye Liu[1]
Wonjoon Cho[1*]
Todd R. Jackson[1]
Nicholas M. Patrikalakis[1]
[1]Design Laboratory
Department of Ocean Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307
Emails:hyl,wjcho,trjackso,nmp@deslab.mit.edu

Emanuel M. Sachs[2]
[2]3D-Printing Laboratory
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307
Email: sachs@mit.edu

ABSTRACT

Solid Freeform Fabrication (SFF) processes have demonstrated the ability to produce parts with locally controlled composition. In the limit, processes such as Three-Dimensional Printing(3DP) can create parts with composition to the length scale of 100 μm. To exploit this potential, new methods for automatic design of Functionally Gradient Material (FGM) parts need to be developed. This paper presents an efficient method for design and composition interrogation of FGM solids. The design tool based on the distance function from the surface of the part requires enhanced efficiency, and so does the interrogation of the part. The approach for improving efficiency includes preprocessing the model with bucket sorting and 3D digital distance transform, and an efficient point classification algorithm. Based on these tools, an efficient algorithm for distance function computation is developed for the design of FGM through distance to the surface of the part or distance to an .STL surface boundary. An efficient algorithm to evaluate composition at a point, along a ray or on a plane is also presented.

Keywords: 3D Printing, CAD/CAM, distance function, FGM design/interrogation, SFF

INTRODUCTION

Solid Freeform Fabrication (SFF) technology is a modern Computer Aided Manufacturing technology through which prototypes, parts, and tools are built in an additive fashion directly from CAD models. Among various SFF processes, Three-Dimensional Printing (SHCW93) is one of the SFF manufacturing processes in which a 3D structure is built layer by layer and completed with near-pointwise local composition control (LCC) via selective placement of different materials, as illustrated in Figure 1. The LCC characteristics of 3DP manufacturing opens the door to the production of parts with Functionally Graded Material (FGM). FGM has many possible applications such as structural property control, thermal property control, medicine delivery control, multicolor visualization, etc.

The traditional CAD systems by representing models only in terms of geometry and topology do not facilitate the fabrica-

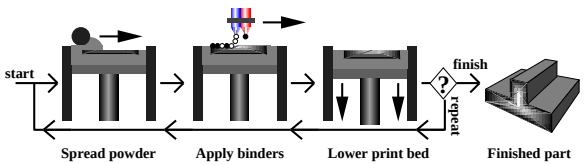

Figure 1. FUNCTIONING OF LCC OF 3D PRINTING, ADAPTED FROM (JLP+99)

*Corresponding author. Tel.: (617)253-7950; fax: (617)253-8125; email: wjcho@deslab.mit.edu; MIT Room 5-424, Cambridge, MA 02139-4307.

tion of an FGM part through LCC. In order to overcome this obstacle, new methods to represent, design, and process models with graded material compositions need to be developed. Therefore, researchers in SFF community have been investigating the method of representing FGM objects by extending the existing CAD representation methods. Based on the r-sets model, Kumar and Dutta (KD98) proposed a heterogeneous solid model (r_m-object) as a finite number of subdivisions (r_m-set) with each subdivision being a material domain with an analytic material function. The models are constructed through Boolean operations. Pegna and Safi (PS98) proposed a multi-material model using volumetric data sets. This decomposition method allows easy representation and ready visualization using existing FEA software. Jackson *et al.* has developed a prototype FGM representation using an extension of cell-tuple data structure (Bri93) with volumetric FGM cells (JLP$^+$99). Over each cell's domain, the shape and composition is formulated in terms of a set of control points and control compositions which are blended with the barycentric Bernstein polynomials (HL93). The prototype FGM system at MIT has been tested with models subdivided into tetrahedral meshes, and the conversion from traditional solid models to FGM system was done by employing standard meshing algorithms (WSP$^+$00).

Although the developed system provides a useful tool for designing compositions in terms of distance from a fixed feature in a straightforward manner, the efficiency in the evaluation of distance function becomes an issue especially when designing from the boundary of a model. For example, the algorithm for assigning the control compositions must compute the minimum distance from each query point (corresponding to a control composition) to the boundary of the model. With the potential of a model having a large number of query points, the exhaustive searching through all the boundary facets may be prohibitively time consuming. In this paper, an efficient distance function algorithm is developed based on the bucketing algorithm and the digital distance transform of the buckets.

The CAD modeling system needs to provide not only the design tool, but also functionality for the user to evaluate the properties of the FGM model. Efficient evaluation of composition of FGM will be most important for either the visualization or the post-processing of the FGM model. Query of the composition may be in the form of the query for a point, a ray, or a plane. Composition evaluation can be evaluated either directly using the efficient distance functions or using interpolation of the control compositions in case the model is a subdivision method. In the second case, an efficient point location algorithm is developed using also the bucketing method in order to identify the sub-region corresponding to the query point, and the control compositions of that sub-region are used to interpolate for the query point. Composition evaluation along a given ray or on a given plane can be easily achieved by extending the composition evaluation algorithm at a point.

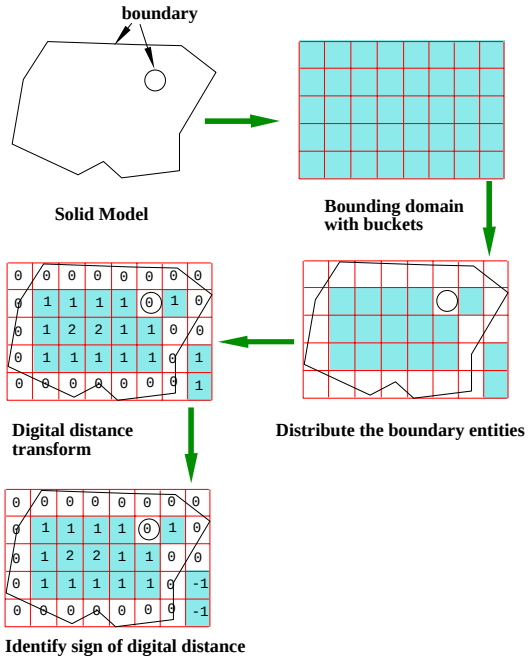

Figure 2. PREPROCESSING

3D BUCKETING AND EFFICIENT DISTANCE FUNCTION EVALUATION

In order to improve the efficiency in the computation of minimum distance from a query point to the boundary of the model or a given .STL boundary, a preprocessing method including bucket sorting (CLR90), digital distance transform, and identification of solid buckets is employed. After such preprocessing, computation of the Euclidean distance for a specific query point has to be done with respect to boundary entities in the buckets which have the nearest digital distance to the query point (Liu00). Figure 2 graphically illustrates this preprocessing.

Construction of 3D Bucketing System

The 3D bucketing process begins by dividing the space occupied by model's bounding box into equal sized cubic subregions (buckets). (CMP96) has detailed formula for the bucketing frame construction. After we build the bucketing frame system we need to transfer the physical coordinate position of each geometric entity to its new bucketing coordinates by translation and scaling.

Placement of the Geometric Entities into Buckets

Once the system of buckets is established, the geometric entities defining the model are sorted in to buckets. *These geometric entities may be the trimmed NURB patches of a B-rep model, facets of an STL file, or tetrahedra in a FE mesh*. This process consists of intersecting each geometric entity with the system of

```
* * * * * * *        3 3 3 3 3 3 3
* * * * * * *        3 2 2 2 2 2 3
* * * * * * *        3 2 1 1 1 2 3
* * * 0 1 2 3        3 2 1 0 1 2 3
* * 1 1 1 2 3        3 2 1 1 1 2 3
* 2 2 2 2 2 3        3 2 2 2 2 2 3
3 3 3 3 3 3 3        3 3 3 3 3 3 3
After forward pass   After backward pass
```

Figure 3. EXAMPLE OF CHESSBOARD DISTANCE TRANSFORM: 0 IS THE FEATURE PIXEL

buckets. For each bucket an entity intersects, a reference to that entity is added to the corresponding bucket(s).

3D Digital Distance Transform

With the relationship established between the geometric entities defining the model and the bucketing system, the chessboard distance transform of the buckets is computed. A distance transform (DT) in 2D is an operation that converts a binary picture, consisting of feature and nonfeature elements, to a picture where each element has a value that approximates the distance to the nearest feature element. Please refer to (Bor84) for detailed digital distance transforms in arbitrary dimension. The chessboard distance transform is illustrated in Figure 3. In 3D chessboard transform, the masks would be like in Figure 4 that is composed of two planar masks in two planes. Considering that one 3D voxel has 26 neighbors, in each path of the transform, the masks will transform 13 neighbors which is consistent with the 13 masks apart from the feature voxel. The algorithm is also adapted for the buckets that are on the border of the bounding box. For those buckets, the number of operators in mask should be reduced accordingly.

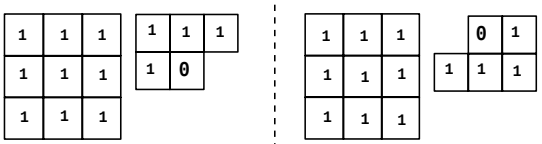

Figure 4. CHESSBOARD DT MASK IN 3D

Identification of Solid Buckets

From the previous step computing the distance transform, the boundary buckets and non-boundary buckets can be identified. Here to efficiently search the sub-region location of a query point with Cartesian coordinates, the method involves processing of all the non-boundary buckets in order to identify all the buckets that are inside the solid. For convenience we call such buckets 'solid buckets'. Solid buckets are non-boundary buckets, therefore, as long as one point inside a bucket is inside the solid, then the bucket is a solid bucket. So the algorithm is to use the center point of a non-boundary bucket as a seed, check if it is inside the solid. If the seed is inside the solid then the corresponding bucket is a solid bucket, otherwise, the bucket is outside the solid.

Efficient Distance Function Evaluation Examples

Based on the preprocessing described so far, we can compute the exact minimum Euclidian distance for each query point (interior) by computing minimum distances to those boundary entities located in the nearest buckets. Suppose a bucket the

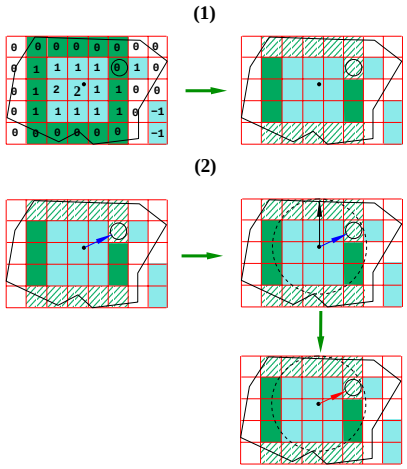

Figure 5. COMPUTE EXACT EUCLIDEAN MINIMUM DISTANCE FOR POINT INSIDE BOUNDING BOX

query point lies in is B_q, the nearest non-empty buckets relative to B_q make up a list of buckets called $ListB_{nr}$. All $B_i \in ListB_{nr}$ have the same chessboard distance from B_q of the value that is stored in B_q. $ListB_{nr}$ can be easily constructed by checking each bucket in the cubic shell that is offset at that value. After the nearest buckets are found, the exact minimum distance to the boundary of the object can be calculated by computing the minimum distance from the query point to every boundary entity inside those buckets. The next step is to check and repeat the procedure to guarantee the exact minimum distance is obtained because we were using the biggest empty cube in probing instead of the sphere to find the minimum distance. Figure 5 illustrates the procedure in steps. For sequence of query points, we can check if a new query point lies in the same bucket as the one before, so time is saved in finding $ListB_{nr}$.

When we extend the design method via distance to the model boundary to the design method via distance to an arbitrary .STL boundary, an efficient evaluation of the distance from a query

point outside the .STL boundary bounding box needs to be developed. For a query point outside the bounding box of the given .STL boundary surface, there is no query bucket existing, therefore, we don't have the information available about the nearest boundary buckets in terms of minimum digital distance. The method for this situation is to calculate the worst estimated buckets that might have the minimum digital distance from the virtual query bucket, then to search the minimum distance progressively further from that shell of buckets.

Complexity Analysis of Distance Function Computation

It should be noted that the performance of the bucketing algorithm depends on both the number of boundary entities (n_{bf}) and the number of query points (m). Because different points require different computing time; i.e. the nearer a point is to the boundary, the fewer buckets need to be searched. In addition, the geometry of the boundary also influences the performance of the bucketing algorithm; the bigger amplitude the boundary oscillates to (relative to the dimension of boundary), the fewer searches are needed, because points are more locally confined. From these observations and for the simplicity of the analysis, we use a cube as a representative model. Here we only consider the average of (m) operations of minimum distance computation. Recall from previous parts of this paper that query points correspond to control composition points within the model and maybe distributed throughout the model. Here below, we will use parameter n instead of n_{bf} because the number of buckets n is equal to n_{bf}. For convenience of this complexity analysis, it

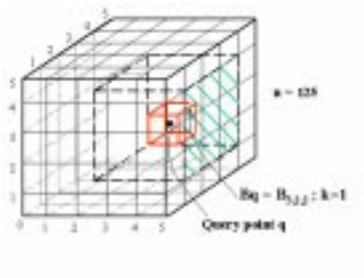

Figure 6. k IS THE CHESSBOARD DISTANCE OF THE QUERY BUCKET TO THE BOUNDARY BUCKETS; NUMBER OF NON-EMPTY BUCKETS THAT THE BOUNDARY OCCUPIED IS $6n^{2/3}$, WHERE n IS THE NUMBER OF TOTAL BUCKETS; BUCKETS IN THE SHADED AREA ARE NEAREST NON-EMPTY BUCKETS NEEDED TO BE SEARCHED FOR THE QUERY POINT

is assumed that the query points result from uniform rectangular meshing. Consider a cube object as in (Figure 6), with the boundary meshed into n_{bf} triangular facets. Assuming there are m number of interior points uniformly distributed throughout the body, the total time cost T shall be

$$T = T_{pre} + \sum_{i=1}^{m} T_i \tag{1}$$

where T_{pre} is the time cost for the preprocessing and T_i is the time cost for the i^{th} query point.

From the algorithm of the preprocessing, one can see $T_{pre} = T_{bk} + T_{dt}$, where T_{bk} denotes the time cost for bucketing and T_{dt} denotes the time cost for digital distance transform. It is obvious that T_{bk} is linear to the number of boundary facets ($T_{bk} = O(n)$), and T_{dt} is linear to the number of total buckets which is also n ($T_{dt} = O(n)$). Therefore, $T_{pre} = O(n)$. From the algorithm of the distance function, we can see query points that fall in the same bucket basically cost the same amount of time except for those points lying along the diagonal lines and diagonal planes. Suppose the time cost for distance computation of query points lying in the buckets that have chessboard distance k to the boundary buckets is $T_q(k)$, we can rewrite $\sum_{i=1}^{m} T_i$ as follows:

$$\sum_{i=1}^{m} T_i = \sum_{k=0}^{\lfloor n_s/2 \rfloor} T_q(k) \cdot N(k) \tag{2}$$

where $n_s = \sqrt[3]{n}$ and $N(k)$ is the number of query points whose buckets have chessboard distance value k to the boundary buckets. Since we sample query points uniformly, we can define the density of number of query points per bucket as ρ, and because we have in total n buckets and m query points, therefore $\rho = \frac{m}{n}$. Further, we define the number of buckets which have chessboard distance value k to the boundary buckets as $N_b(k)$. Graphically, we can see those buckets make up the cubic shell that is between the cube with side length $n_s - 2k$ and the cube with side length $n_s - 2k - 2$. Therefore, one can deduce that $N_b(k) = [(n_s - 2k)^3 - (n_s - 2k - 2)^3]$, and then

$$N(k) = N_b(k) \cdot \rho = [(n_s - 2k)^3 - (n_s - 2k - 2)^3] \cdot \frac{m}{n} \tag{3}$$

From the algorithm for computation of distance for a single query point, one can deduce that the time cost $T_q(k)$ includes the time for searching for nearest boundary buckets in the layer of buckets that have chessboard distance to the query bucket B_q and the time cost in computing the exact Euclidean distances from the query point to all the boundary facets that are in the nearest boundary buckets. Here we denote the time for searching as T_{sr} and the time for computing exact distances as T_{cmp}. Therefore,

$$T_q(k) = T_{sr}(k) + T_{cmp}(k) \tag{4}$$

For convenience of analysis, here we define several characteristic numbers that are related. We define $N_{tb}(k)$ as the number of buckets that need to be searched for a query point that has chessboard distance k to the boundary buckets. Buckets to be searched are the cubic shell of buckets that has offset of k to B_q, therefore, $N_{tb}(k) = (2k+1)^3 - (2k-1)^3$ if $k > 0$, $N_{tb}(0) = 1$. We also define N_{bs} as the number of boundary squares on the six outer sides of the cubic shell and $N_{bs} \leq 6(2k+1)^2$. We use $N_{ft}(k)$ as the number of facets in the nearest buckets that have to be calculated for exact Euclidean distances and $N_{ft} = P_f \cdot N_{bs}(k)$, where $P_f = \frac{\sqrt[3]{n}}{6}$ which is the number of facets per square on the boundary of the cubic object. After the above definitions, we can deduce that: $T_{sr}(k) = c_1 \cdot N_{tb}(k); T_{cmp}(k) = c_2 \cdot N_{ft}(k)$ where c_1 and c_2 are constants, therefore

$$T = O(n) + \sum_{k=0}^{\lfloor n_s/2 \rfloor} T_{sr}(k) \cdot N(k) + \sum_{k=0}^{\lfloor n_s/2 \rfloor} T_{cmp}(k) \cdot N(k) \qquad (5)$$

After expanding the above formula, we have the final expression of time cost:

$$T = O(n) + O(\frac{3c_1}{5} m n^{\frac{2}{3}} + \frac{c_2}{10} m n) \qquad (6)$$

APPLICATION TO FGM OBJECT MODELING

Design of Composition in FE-mesh based Model through Distance Functions

The data structure of an efficient finite-element based FGM model (Liu00) is presented here. Specifically, the data structure maintains an array of object "Vertex" pointers, an array of objects "Tetrahedron" pointers, a list of indices of boundary tetrahedra and a bucketing system. Each bucket is associated with a list of triangular boundary facets and a list of vertices. An object of "Vertex" is associated with the position of the vertex, a queue of incident tetrahedra indices and the material composition vector at that vertex. A tetrahedron has an array of four vertices, and the status of each tetrahedron, i.e. boundary tetrahedron or not; in addition, if a tetrahedron is a boundary tetrahedron, the status of each face is stored according to the face being a boundary face. Figure 7 demonstrates the data structure of the FE-based FGM model. After the initialization of the data structure, the list of boundary tetrahedra is obtained, each of which has its boundary facets identified explicitly. Based on this list, an array of boundary facets is constructed with their normal vectors calculated. The normal vectors of the boundary facets is pointing outward of the solid.

Based on the above FGM modeling system, the design of graded material composition can be done by assigning composition value on each control composition vertex inside the model.

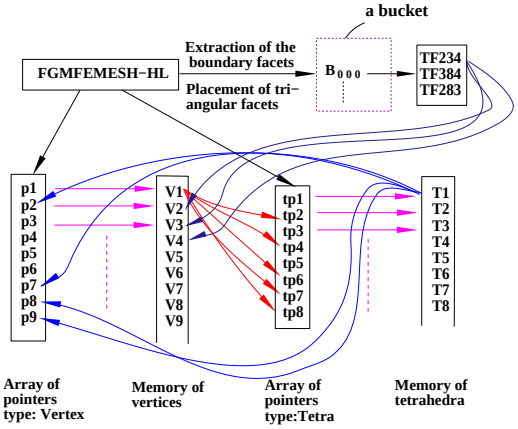

Figure 7. FE-MESH BASED DATA STRUCTURE

Nevertheless, this approach is ineffective because a model may have millions of cells and millions of control compositions to assign. Furthermore, this approach is not intuitive for users to implement their design ideas. Therefore, an efficient design tool needs to be developed. One of the methods of designing FGM is to assign composition to the control compositions according to its minimum distance to the boundary of the object. For a single point inside the FGM body, the minimum distance to the boundary surface is the minimum of the minimum distances from the point to every boundary triangular facet. As we can see immediately, such a minimum distance can be computed repeatedly over each boundary facets, but since the represented real part model usually has a large number of boundary facets, the direct computational method will be too costly in time. Therefore an efficient algorithm is necessary for the design of FGM through distance function. With the tool we have presented in the previous section, the user only needs to choose the materials variation in terms of distance from the boundary. Here the variation of materials has to satisfy certain design rules for matetial system (JLP+99). The method of designing FGM as to the distance from the boundary of model can be extended to designing FGM as to the distance from arbitrary .STL shell, which is in the form of a set of triangular facets.

Another design scheme is to allow users to design only the control compositions contained in a given .STL shell, and the composition functions can be one of the distance functions to different fixed features. It is necessary to efficiently identify those control compositions inside the .STL among the whole set of control compositions. Because of the large number of control compositions, efficiency is also important here.

Efficient Evaluation of Composition

Composition Evaluation at a Point In order to evaluate the property of a FGM model or processing the FGM for manufacturing, it maybe necessary to evaluate the composition at a

point. The method may vary for different FGM models, i.e. STL model, r_m-set, cell-tuple, or finite element mesh. Given an .STL or r_m-set like B-rep model, with the boundary bounded by many faces, if the composition function is assigned as to the distance from the boundary, the efficient computation of distance function can be applied to evaluate the composition directly. Given a decomposition representation like FEM structure, i.e. a tetrahedra mesh, the composition evaluation at a query point Q can be done by interpolating the composition values at the nodes of the object tetrahedron. If given a cell-tuple model, the interpolation can be arbitrary degree of Barycentric Bernstein Polynomial.

To evaluate a composition at a point, the identification of a sub-region (or an element) containing the point should be performed efficiently. Our preprocessing algorithms could provide a possible solution for the efficiency improvement. As an appliation example, we consider a finite-element based model described in the previous section. Point location (O'R93) is one of the classical problems in computational geometry and has been extensively studied. Bucketing(AEI$^+$85) as one optimal method can even achieve constant time for a uniformly distributed mesh. Here in our application: for a given query point, the algorithm checks if the model contains this point (Point membership classification); and for an interior query point, the algorithm return the tetrahedron that contains it (Identification of the object tetrahedron).

Point Membership Classification (PMC) involves testing if a given query point Q is contained inside the body (or on its boundary) or not. The most popular algorithm of PMC with respect to B-rep solid is ray-casting. In our method, the steps are as follows: Shooting a ray from Q to one of the six coordinate directions, using the stored digital distance value to find the *first* boundary bucket in that direction, start from this bucket to find the first boundary triangular facet the ray intersects. In this work, the shooting direction is chosen such that the bounding box is nearest in that direction. By checking if the inner product of ray vector and the oriented normal vector of the triangular facet is positive or negative, one can determine if a point is inside or outside the solid, respectively. Here the method for detecting the first boundary triangular facet the ray intersects is to compare the parameter values of the intersection points. If the intersection points happen to coincide, then the triangular facet that has the smaller minimum distance to the query point will be chosen. If the facets involved have the same minimum distance to the query point, then the boundary triangular facet that has the bigger inner product magnitude with the ray is chosen. One other special case is that the intersection point has parameter 0, which means the query point is on the boundary, hence we can use the corresponding boundary facet to identify the tetrahedron it is located in. In case of a large FEM model and large number of query points, an efficient method is necessary instead of the exhaustive searching method (checking every mesh element in the mesh) to identify the object tetrahedron. Our algorithm for locating the object tetrahedron is to jump onto a vertex that is near to the query point with the help of bucket sorting of vertices and then trace the straightline segment between the starting vertex and the query point to search and check each tetrahedron that intersects the line segment. The idea is illustrated in Figure 8. Detailed algorithms with respect to the input model and the query point are described in (Liu00).

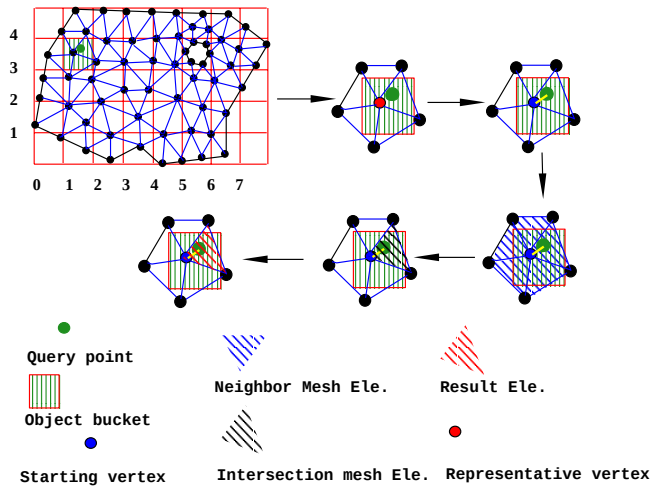

Figure 8. LOCATING THE TETRAHEDRON CONTAINING A GIVEN POINT

Composition Evaluation on a Plane or along a Ray

For the purpose of processing an FGM model to machine instruction, composition evaluation on a plane or along a casting ray will be necessary. Given a plane in general form $A \cdot x + B \cdot y + C \cdot z - D = 0$, The method is calculating the intersection points of the plane with the bounding box of the model, using the intersection points coordinates one can express the plane in parametric form $X(u,v) = X_{b_1} + (X_{b_2} - X_{b_1})u + (X_{b_3} - X_{b_2})v$. Then the composition evaluation reduces to the evaluation of compositions with respect to the set of parametric points. This is achieved by using the algorithm described in the previous section. Similarly, given a ray, we can first calculate the intersection points of the ray with the domain of the body, then using the intersection points to express the ray $X(t) = (1-t) \cdot X_b + t \cdot X_e$ and evaluate the parametric point along the ray at a given resolution. Figure 9 illustrates the algorithm of evaluation of composition along a ray.

Time Complexity Analysis of the Algorithms

Time Cost of Point Location Algorithm From the algorithm described in the previous text, we can see the time cost of point location for a single query point is $T = T_{in} + T_{mch}$,

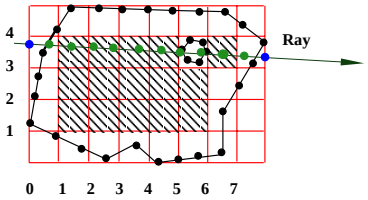

Figure 9. EVALUATE COMPOSITION ALONG A GIVEN RAY

where T_{in} is the time spent on checking if the query point is inside the body, here this time will be linear to the number of boundary facets in one bucket, which is about $\sqrt[3]{n_{bf}}/6$ for a cube, where n_{bf} is the total number of boundary facets. Hence, $T_{in} = O(\sqrt[3]{n_{bf}})$. T_{mch} here is the time spent on marching all the tetrahedra from a starting vertex to the query point. If we define the number of tetrahedra we checked as N_{tc}, the number of faces we checked to find the neighbor tetrahedron along the line as N_{fc} and the time for each neighboring check as T_{cnb}, we have $T_{mch} = O(N_{tc}) + O(N_{fc} \cdot T_{cnb})$. We use a modified version of the algorithm used for checking interior faces to check the neighboring tetrahedron. From the *Matching Theorem* (Vos93) and by assuming the FEM data homogeneous, we know on average one vertex has $4n_t/n_v$ number of incident tetrahedra, hence, $T_{cnb} = 12 \cdot O(n_t/n_v)$. Since for each tetrahedron visited, we need to check 0 to 1 face for neighboring, therefore,

$$T_{mch} = O(N_{tc}) + O(N_{tc} \cdot \frac{n_t}{n_v}) \tag{7}$$

From all the above, we deduce that:

$$T = O(c_1 \cdot \sqrt[3]{n_{bf}} + c_2 \cdot N_{tc} + c_3 \cdot N_{tc} \cdot \frac{n_t}{n_v}), \tag{8}$$

where c_1, c_2, c_3 are constants. It will be interesting if we can give an expectation of the value of n_t/n_v. We know for a Delaunay mesh, the worst case would be quadratic, and in problems of practical relevance, this degree is expected to be constant (MSZ99).

Time Cost of Ray Casting Algorithm In order to analyze the time cost of ray casting, we define several parameters as follows: δt as the resolution which is a number between $0-1$; n_r as the number of query points inside the bounding box along the ray; N_{br} as the number of boundary buckets that intersect the ray; n_{bq} as the number of query points in boundary buckets along the ray; ρ as the number of query points per bucket that intersects the ray. From our ray casting algorithm described before, we know that the total time of the algorithm T should be equal to the time spent on checking each point for interior (T_{ckin}) status plus the time spent on locating the sub-regions for points identified as interior (T_{lcpo}). Therefore, we have:

$$T = T_{ckin} + T_{lcpo} \tag{9}$$

$$T_{ckin} = n_{bq} \cdot O(\sqrt[3]{n_{bf}}) + (n_r - n_{bq}) \cdot O(1) \tag{10}$$

$$n_{bq} = N_{br} \cdot \rho \tag{11}$$

$$n_r \leq 1 + \frac{1}{\delta t} \tag{12}$$

For simplicity, suppose the bounding box is a cube, that is $n_x = n_y = n_z$, then the shooting ray can at least intersects $\sqrt[3]{n_{bf}}$ buckets. Hence

$$\rho \leq \frac{n_r}{\sqrt[3]{n_{bf}}} \leq \frac{\delta t + 1}{\delta t \cdot \sqrt[3]{n_{bf}}} \tag{13}$$

Therefore,

$$n_{bq} \leq \frac{(1+\delta t) \cdot N_{br}}{\sqrt{3} \cdot \delta t \cdot \sqrt[3]{n_{bf}}}; \qquad 0 \leq \delta t \leq 1 \tag{14}$$

Hence $T_{ckin} = O(\frac{N_{br}}{\delta t})$. As to the time of locating positions, suppose that all the query points are inside the body, we know that: $T_{lcpo} = n_r \cdot T_{mch}$, where T_{mch} is defined as in the last section. Therefore,

$$T_{lcpo} = O(\frac{N_{tc}}{\delta t} \cdot [1 + \frac{n_t}{n_v}]) \tag{15}$$

$$T = O(c_1 \cdot \frac{N_{br}}{\delta t} + c_2 \cdot \frac{N_{tc}}{\delta t} \cdot [1 + \frac{n_t}{n_v}]) \tag{16}$$

IMPLEMENTATION

The algorithms developed in this paper are implemented in Microsoft Visual C++ on a Intel Pentium II CPU 450M with SRAM 128MHz. This work is integrated with an existing environment of modeling, designing, and postprocessing of FGM. The approach is using object oriented programming techniques such as inheritance of classes and virtual functions. For the newly developed functionalities, the relevant user interfaces are extended using also OOP and Microsoft Foundation Classes programming.

Experimental Results of Distance Function Comparison with Exhaustive Searching Method

The test runs are done on several models with uniform query points chosen in the bounding domain of each model. Models are

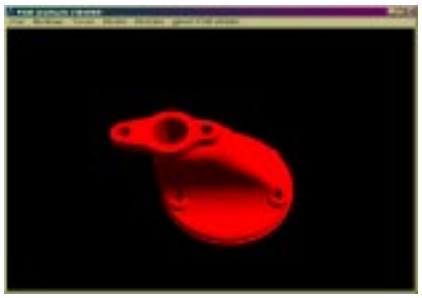

Figure 10. MODEL 'Sump'

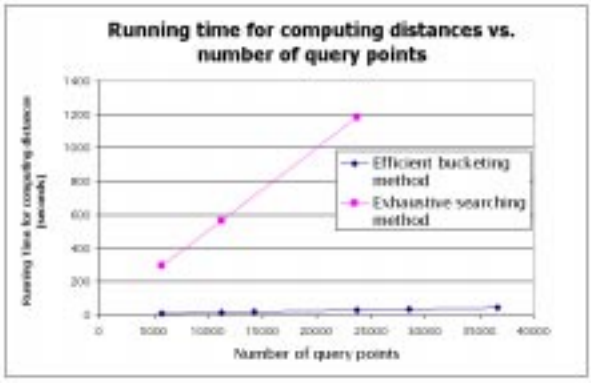

Figure 11. EXPERIMENTAL RESULTS ON 'Sump'

listed in Table 1 with the corresponding factor of efficiency enhancement (E_f) and the preprocessing time (PT) for each model. From this table, we can see for all the models, the efficiency enhancement is better than the theoretical value from the previous worst case time complexity analysis. One visualized model and the corresponding curve of experimental results comparison with exhaustive searching method are also given in Figures 17 to 18. The factor of efficiency enhancement is obtained through comparing the slopes of the linear regression of running curves with both methods. From these experimental results, one can conclude that our bucketing-based algorithms significantly improve the efficiency compared to the exhaustive searching method by a large factor.

Table 1. EFFICIENCY ENHANCEMENT ON THE EXAMPLE MODELS

	MODEL 'Pill'	*MODEL 'Propeller'*	*MODEL 'Bracket'*	*MODEL 'Sump'*
n_{bf}	9572	6210	2924	10296
E_f	15.79	24.42	13.9	44.82
$PT(sec)$	2.47	2.47	1.02	3.81

Table 2. PARAMETERS OF THE FGM EXAMPLE MODELS

	MODEL 'Pill'	*MODEL 'Brack'*	*MODEL 'Widget'*
n_{bf}	788	2884	11038
n_v	1329	2356	18683
n_t	6091	8503	83827

Table 3. PERFORMANCE OF PROGRAM ON THE EXAMPLES

Time (sec)		*MODEL 'Pill'*	*MODEL 'Brack'*	*MODEL 'Widget'*
IT		*10.88*	*16.26*	*382.01*
PT	*BT*	*0.11*	*0.22*	*0.88*
	DT	*0*	*0.11*	*0.39*
	ST	*0.38*	*1.21*	*4.61*
DBT		*6.43*	*6.26*	*31.75*
PCT $\delta t = 0.01$		*7*	*7*	*6.43*

Results

Results on three models are given here, whose parameters are given in Table 2. Table 3 gives the experimental running results of the above three models, where IT is the time spent on the initialization of data structure and extraction of the boundary tetrahedra, BP is the time spent on creating the bucketing system, DT is the time spent on digital distance transformation, ST is the time spent on identifying the signs of buckets, DBT is the time spent on the designing composition as to the distance from boundary and PCT represents the time spent on the evaluation of composition on one slice.

Using an exploratory environment of modeling, designing, and post-processing of FGM object developed at M.I.T., FGM objects can be automatically designed through various distance functions with respect to different features. The visualization of an FGM object can be either the composition on the boundary, the color-coded control compositions, cuberille or in the forms of a bunch of color-coded slices. Figure 12 together with Figures 13, 14 show an example FGM model created by designing through distance function to the boundary of the object.

CONCLUSION

As part of the larger project of "Modeling and Designing Functionally Graded Material Components for Fabrication with Local Composition Control", this paper introduces efficient algorithms for design and composition interrogation of FGM solids. Enhancement of the efficiency in designing FGM becomes necessary when the representation of the FGM model has a large number of boundary faces and/or sub-regions. Similarly the algorithm for composition evaluation of an FGM has to be efficient

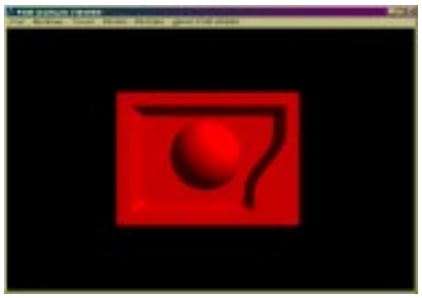

Figure 12. WIDGET

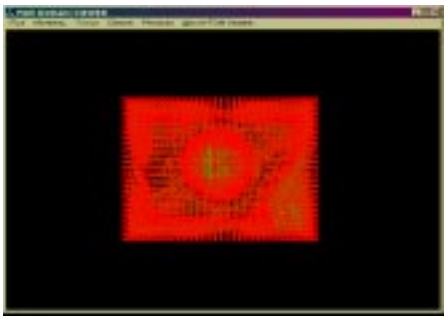

Figure 13. WIDGET COMPOSITION DATA

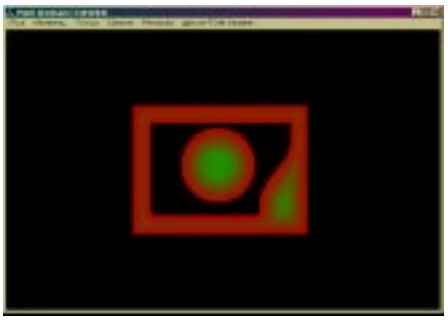

Figure 14. WIDGET SLICE

as well. Efficient algorithms for design and composition evaluation have been developed through the methods of bucket sorting, distance transform, and point classification, etc. The analysis on the developed algorithms and experimental results demonstrated these algorithms are effective. The developed algorithms have also been integrated into an existing FGM modeling, designing and post-processing program through software development techniques, which provides the pathway for the designer to design and model FGM and then see an FGM part fabricated through LCC, especially through 3D Printing process.

ACKNOWLEDGMENTS

Funding was provided by NSF (grant #DMI-9617750) and ONR (grant #N00 014-96-1-0857).

REFERENCES

T. Asano, M. Edahiro, H. Imai, M. Iri, and K. Murota. Practical use of bucketing techniques in computational geometry. In G. T. Toussaint, editor, *Computational Geometry*, pages 153–195. 1985.

G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27:321–345, 1984.

E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete and Computational Geometry*, 9:387–426, 1993.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

W. Cho, T. Maekawa, and N. M. Patrikalakis. Topologically reliable approximation of composite Bézier curves. *Computer Aided Geometric Design*, 13(6):497–520, August 1996.

J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

T. R. Jackson, H. Liu, N. M. Patrikalakis, E. M. Sachs, and M. J. Cima. Modeling and designing functionally graded material components for fabrication with local composition control. *Materials and Design*, 20(2/3):63–75, June 1999.

V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. *Journal of Mechanical Design, ASME*, 120:659–667, December 1998.

H. Liu. Algorithms for design and interrogation of functionally graded material solids. Master's thesis, M.I.T., Cambridge, MA, February 2000.

E. P. Mücke, I. Saias, and B. Zhu. Fast randomized point location without preprocessing in two- and three-dimensional delaunay triangulations. *Computational Geometry*, 12:63–83, 1999.

J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993.

J. Pegna and A. Safi. CAD modeling of multi-modal structures for free-form fabrication. In *Presentation at Solid Freeform Fabrication Symposium*, Austin, Texas, August 1998.

E. Sachs, J. Haggerty, M. Cima, and P. Williams. Three-dimensional printing techniques. U.S. Patent No. 5,204,055, April 20 1993.

K. Voss. *Discrete Images, Objects, and Functions in Z_n*. Springer-Verlag, 1993.

H. Wu, E. M. Sachs, N. M. Patrikalakis, D. Brancazio, J. Serdy, T. R. Jackson, W. Cho, H. Liu, M. Cima, and R. Resnick. Distributed design and fabrication of parts with local composition control. In *2000 NSF Design and*

Manufacturing Grantees Conference, Vancouver, BC, Canada, January 2000. http://deslab.mit.edu/3dp/3dppapers.html, http://www.engr.washington.edu/ uw-epp/nsf/.