Effective Generation of Nonuniform 3D Meshes for Finite-Difference Time-Domain Simulations

Insung Ihm*, Hyeong-Seok Kim**, and Byung-Joon Chang*
* Department of Computer Science, Sogang University, Seoul, Korea
** Department of Electrical and Electronics Engineering, Choong-ang University, Seoul, Korea
ihm@sogang.ac.kr, kimcaf2@cau.ac.kr, hergeg@hanmail.net

Abstract — **Nonuniform meshes are often inevitable for discretizing structures with fine details because it is critical to keep the number of produced mesh cells at reasonable size. This paper presents an automatic mesh generation algorithm which produces effective nonuniform rectangular meshes for FDTD simulations. In generating meshes, our algorithm attempts to minimize the total number of mesh cells, while satisfying the conditions on the maximum mesh spacing and the maximum grading ratio. We briefly describe the algorithm, and report its performance.**

I. Introduction

For an effective analysis of electromagnetic waves through Finite-Difference Time-Domain (FDTD) simulations[1], it is important to produce a quality mesh for a given structure. Simple uniform rectangular meshes are frequently used in tessellation, however, they often entail an enormous number of cells when structures with fine details have to be resolved precisely. A more efficient way is to generate nonuniform meshes that adapt the grid spacing according to the various properties of structures. This paper presents an automatic 3D mesh generation technique that produces effective nonuniform rectangular meshes. Our technique attempts to minimize the number of mesh cells, while satisfying the requirements, imposed by users, on the maximum mesh spacing and the maximum grading ratio. In this abstract, we outline our mesh generation algorithm and then discuss its performance.

II. Problem Definition

Consider a 3D structure, composed of elementary objects which may be either an axis-aligned rectangular parallelepiped (cuboid), a sphere, or an axis-aligned cylinder. Each elementary object is associated with the respective upper limit d_{max} on the maximum grid spacing, which requires that any generated cells inside the object may not have edges longer than the imposed limit.

When the elementary objects are projected orthogonally onto three axis-aligned planes and then, repeatedly, onto three principal axes, we obtain three sets of *initial subdivision intervals*. Fig.1. illustrates one such set on x-axis: it is made of m (m = 10 in this figure) intervals where each interval is associated with its length d_i and the upper limit d_i^{max}, inherited from the projected object(s). While we are concerned with the tessellation of 3D models, the mesh generation problem may be considered as one of adaptive subdivisions of three sets of 1D intervals, one for each axis. In order to generate an effective nonuniform rectangular mesh, we impose the following three requirements on the 1D interval tessellation problem:

1. **[Minimization requirement]** Minimize the number of subintervals generated from the tessellation of the initial subdivision intervals. This condition is to optimize the time and space complexity of the analysis process.
2. **[d_i^{max} requirement]** The lengths of all tessellated subintervals in the ith interval must be shorter than d_i^{max}. This condition is to satisfy the maximum mesh spacing constraint.
3. [α^{max} **requirement**] The ratios of the lengths of any two adjacent subintervals must be smaller than a user-specified α^{max} value. This condition is to satisfy the maximum grading ratio constraint.

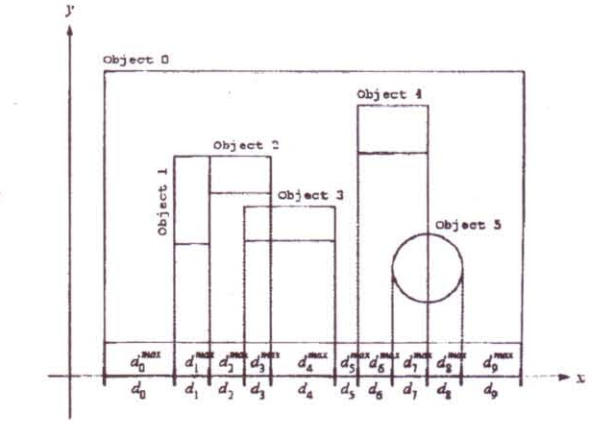


Fig. 1. Projection of a 3D model onto x-axis

III. The Proposed Algorithm

Given a set of m initial subdivision intervals, we first focus on tessellating each interval. Consider the ith interval. Due to the d_i^{max} requirement, it can not be subdivided using fewer than $n_i = ceil(d_i / d_i^{max})$ intervals. Since one of the most

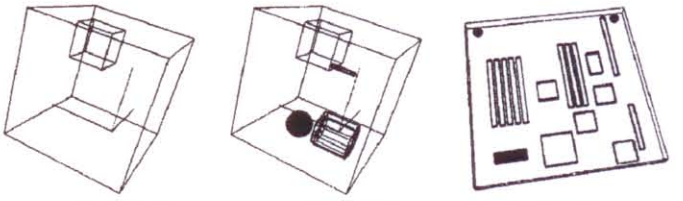
important factors in tessellation is to minimize the number of cells, we attempt to tessellate each initial interval using n_i subintervals. In the process of fulfilling the α^{max} requirement, however, it is frequently forced to increase the number of subintervals (imagine two adjacent initial intervals with steeply varying d_i^{max} values). In the presented technique, the interval is tessellated symmetrically, in other words, $e_{i,j} = e_{i,n_i-1-j}$, $\frac{1}{\alpha^{max}} \le \frac{e_{i,j}}{e_{i,j+1}} \le \alpha^{max}$, where $e_{i,j}$ (j = 0, 1, …, n_i-1) is the length of the jth subinterval. There are two cases of the tessellation: the first one is that the subintervals' lengths increase gradually with ratio α_i and then in turn decrease symmetrically. In the second one, the lengths vary the other way around. The key in this stage is how to appropriately select the length $e_{i,0}$ ($=e_{i,n_i-1}$) of the boundary subintervals. Note that, if the ith interval is to be subdivided using n_i subintervals, $e_{i,0}$ must be chosen from a feasible range $[e_{i,0}^{min}, e_{i,0}^{max}]$ that is determined by the d_i^{max} and the α^{max} requirements (we will show how to derive this range in the full paper).

Now, we have m such feasible ranges from which each $e_{i,0}$ needs to be selected. One strategy is to assign a properly chosen value $e^*_{i,0}$ identically to all $e_{i,0}$ values. Suppose that the intersection of the m ranges is non-empty. Because it is desirable to set $e^*_{i,0}$ as large as possible, we compute it as $e^*_{i,0} = \min_{i=0}^{m-1} e_{i,0}^{max}$. When the intersection is null, this is the case where the feasible range of some initial subdivision interval, say, interval j, is simply to the right of $e^*_{i,0}$. We need to move such a range to the left by inserting more subintervals, that is, by increasing n_j gradually until a non-empty intersection exists. By adapting such *bad* intervals one by one, $e^*_{i,0}$ can be chosen as we wish at the expense of increasing the number of subintervals. Once $e^*_{i,0}$ is set, the proper α_i value is determined per interval using it and then the tessellation process proceeds.

While the described technique produces a tessellation which fulfills both the d_i^{max} and the α^{max} requirements, it turns out that the restriction that all initial subdivision intervals share the same $e_{i,0}$ value is too severe, often resulting in rapid increases in the number of subintervals. A more efficient way is to allow each interval to have a distinct $e_{i,0}$ value. We have developed a selection scheme which reduces the number of generated cells remarkably. The main idea is to decide the $e_{i,0}$ values one by one from the interval whose $e_{i,0}^{max}$ is the minimum first. According to the relation to its adjacent intervals, $e_{i,0}$ is chosen from the four cases: $e_{i,0}^{max}$, $\min\{e_{i-1,0}\cdot\alpha, e_{i,0}^{max}\}$, $\min\{e_{i,0}^{max}, e_{i+1,0}\cdot\alpha\}$, and $\min\{e_{i-1,0}\cdot\alpha, e_{i,0}^{max}, e_{i+1,0}\cdot\alpha\}$. As expected, we find that this adaptive assignment produces a much more efficient tessellation of the initial intervals (see the full paper).

IV. The Performance

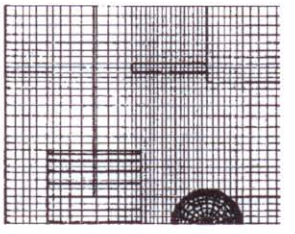
Fig.2. illustrates three tested 3D models, that are made of cuboids, spheres, and cylinders, and TABLE I shows how well the presented mesh generation algorithm works.



(a) Test data 1 (a) Test data 2 (a) Test data 3

Fig. 2. Three example data

In the table, 'Cells' represents the number of produced mesh cells, and 'x-', 'y-', and 'z-axis' indicate the numbers of tessellated subintervals along the respective axes (α^{max} = 1.3). The figure given in the column named `Lower` means the optimal lower bound to each quantity. For instance, one in the 'x-axis' row indicates the sum of all initial n_is. Note that this lower bound is often impossible to achieve. Usually, the actual lower bound is higher since the d_i and the d_i^{max} values along with the α^{max} condition enforce an increase in the real lower bound. In the column named `Ours`, two numbers per row are given. The number in the parenthesis is one produced when an identical $e_{i,0}$ value is applied to all intervals. The other one is the figure that is obtained when $e_{i,0}$ is selected adaptively as proposed in this paper. Considering the fact that the lower bound in the `Lower` column can not usually be realized practically, we believe the presented mesh generation algorithm is quite effective.



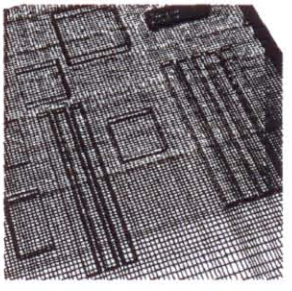


TABLE I Performance statistics

	Test data 1		Test data 2		Test data 3	
	`Lower`	`Ours`	`Lower`	`Ours`	`Lower`	`Ours`
Cells	23,520	32,736 (40,392)	77,326	115,056 (370,668)	112,320	131,625 (467,840)
x-axis	28	31(33)	46	51(68)	104	117(272)
y-axis	28	32(34)	41	48(69)	120	125(172)
z-axis	30	33(36)	41	47(79)	9	9(10)

Acknowledgement: This research was supported by the IITA of MIC, Korea under the ITRC support program.

References

[1] A. Taflove and S. C. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, end ed., Artech House, 2000.