A Fast Void Detection Algorithm for Three-Dimensional Deposition Simulation

Otmar Ertl and Siegfried Selberherr
Institute for Microelectronics, TU Wien
Gußhausstraße 27–29/E360, A-1040 Wien, Austria
Email: {ertl|selberherr}@iue.tuwien.ac.at

***Abstract*—We present an efficient algorithm for the detection of voids which potentially emerge during deposition process simulation. The application of modern level set techniques and data structures enables the fast determination of connected components directly from the implicit level set representation without the need of an additional surface extraction. The algorithm exhibits optimal linear scaling with surface size and is demonstrated on an example, where an isotropic etching process followed by conformal deposition is simulated.**

I. Introduction

Deposition processes can lead to the inclusion of voids. From the point of formation their shapes usually do not change, since they are disconnected from the process chamber. Deposition process simulations should reproduce this simple behavior. However, especially in three dimensions, the solution of more complex models is computationally very expensive. Therefore, often approximations or simplified models are used, which can result in non-physical movement of void boundaries. For example, a constant deposition rate is often used for isotropic deposition simulations.

If the surface is given as triangulation, it is possible to find voids by determining the connected components of the mesh. However, to overcome the arising difficulties with topographic changes during boundary movement, the surface is usually described implicitly using techniques like the level set method [1] or the equi-volume rate model [2]. Furthermore, it has been demonstrated that topography simulations are possible, where the surface rates are calculated using just the implicit surface representation [3], [4]. To avoid the need of a triangulated version of the surface at all, which has to be extracted every time step using costly techniques like the marching cubes algorithm [5], it would be convenient to have an efficient method to detect voids directly using the implicit surface representation.

II. Level set method

The level set method is a technique to describe geometric changes over time and is widely-used for three-dimensional topography simulation [6]. A moving surface $\mathcal{S}$ is described implicitly as zero level set of a continuous function Φ

$$\mathcal{S} = \{\vec{x} : \Phi(\vec{x}) = 0\}. \tag{1}$$

Using this level set function the time evolution of the surface can be simply described by the level set equation

$$\frac{\partial \Phi}{\partial t} + V(\vec{x})\|\nabla \Phi\| = 0. \tag{2}$$

Here $V(\vec{x})$ denotes a velocity field. Since this field has no physical meaning in topography simulations, it has to be extrapolated from the rates on the surface [7]. The level set equation can be easily solved on regular grids using simple finite difference upwind schemes [8].

The level set function is defined on the whole simulation domain, which implies that the memory requirements and also the computation time for time evolution scale with domain size. Different techniques have been developed to obtain optimal linear scaling with surface size (surface area measured in grid spacings).

A. Sparse field level set method

Only level set values of grid points close to the surface have an influence on the surface position. Therefore, it is sufficient to consider only grid points located around the surface for time integration. Only the level set values of these so-called active grid points have to be updated in time, leading to a linear complexity.

This idea was first put into practice by the narrow band method which uses several layers of active grid points around the surface [9]. We use another technique, the sparse field level set method [10], which further decreases the computation time by using just one single layer of active grid points, namely those with level set values in the range -0.5 to 0.5. Since all active grid points are very close to the surface (within a half grid spacing), the required surface velocity extension is very simple. It is even possible to avoid this extrapolation at all, if the surface velocities are calculated directly for all active grid points [4].

B. Hierarchical run-length encoding

The narrow band or the sparse field level set method requires only the level set values of all active grid points and additionally those of grid points in neighboring layers to enable the calculation of derivatives. Hence, the memory requirements can be minimized by storing just the needed level set values.

We use the hierarchical run-length encoded (HRLE) data structure [11] to store just the level set values of the so-called

978-1-4244-3947-8/09/$25.00 ©2009 IEEE

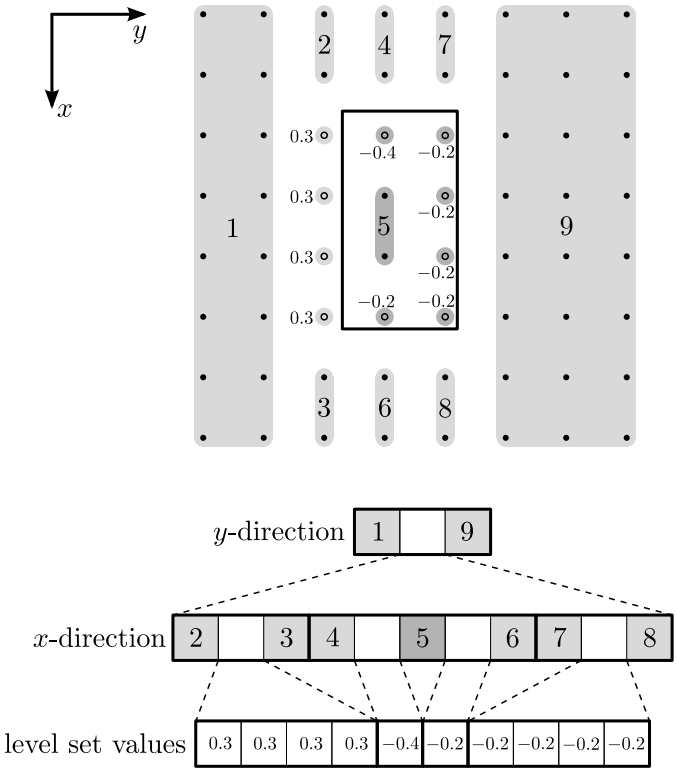

Fig. 1. A rectangle is described by the level set values of the closest grid points (circles). All other undefined grid points (dots) are combined in runs, as shown by the light and dark gray colored regions, which correspond to positive and negative level set signs, respectively. The defined grid points are also colored by the same color scheme. The run-length encoding is hierarchically applied to all grid directions as schematically shown. The integer numbers label corresponding run codes and sets of undefined grid points.

defined grid points. For all other grid points only the signs of their level set values are stored using run-length compression, which is applied recursively to all dimensions. Fig. 1 shows schematically the idea of hierarchical run-length encoding. However, for a detailed description of the data structure we refer to the original paper [11].

The memory requirements of the HRLE data structure scale linearly with the number of defined grid points with little overhead. The defined grid points are stored in lexicographical order. This enables the sequential lexicographical iteration over all grid points in linear time. Random access to grid points is provided with a worst case logarithmic complexity. Since for each grid point at least the sign of the level set value is stored, the information is available on which side of the surface a point is located. This is useful for geometric boolean operations or for multi-level-set methods to describe different material regions [12]. Furthermore, the adaptiveness of the HRLE data structure allows open boundaries, which enables the simulation on grids with infinite extensions.

The HRLE data structure can be combined with the sparse field level set method to obtain a fast level set framework, which exhibits linear scalings in terms of surface size for memory requirements and computation time [12].

III. Void detection algorithm

The level set function partitions the simulation domain into connected components. In the following we present a fast algorithm which uses some properties of the HRLE data structure to determine for each grid point the corresponding component it belongs to. The determination of connected components gives the information about existing voids. If there are no voids, there are just two components which correspond to the bulk material and the region above the surface as part of the process chamber. The obtained connectivity information can also be used to ensure that the geometry of voids does not change after they have been formed.

A. Connected components

We define two neighboring grid points to be connected, if and only if they have the same level set sign. If they do not have the same sign, the zero level set, and hence the surface separates them. The connectivity relations between neighboring grid points can be described by a graph, where each grid point corresponds to a vertex. The connectivity of two neighboring grid points is represented by an edge between the corresponding vertices. According to elementary graph theory the connected components of a graph can be determined with a complexity of $\mathcal{O}(V + E)$, where V and E denote the number of vertices and edges, respectively [13]. Obviously, setting up a full graph with vertices for each point of the regular grid is not reasonable, since the memory requirements and also the computation of the connected components would scale with the domain size and not linearly with the surface size.

B. Graph setup algorithm

If the HRLE data structure is used, the utilization of the following properties allows the setup of a reduced graph, which already combines several grid points within a vertex, and consequently, for which it is much easier to determine its connected components:

- The HRLE data structure leads to a segmentation of the grid. Such a segment is either a defined grid point or an undefined run, which combines one or more undefined grid points with the same level set sign (compare Fig. 1).
- All grid points within a segment are connected. The connectivity follows for undefined runs from the fact, that all contained grid points are neighbored and have the same sign. Hence, if any points of two different neighboring segments are connected, all of their points are connected among each other.
- Two segments are neighbored, if and only if at least one of their corresponding first points is a neighbor to the other segment. Here the first point of a segment means the first point according to the lexicographical order given by the HRLE data structure. As consequence, it is sufficient to obtain all required connectivity relations between segments, by testing the 6 neighbor points of all first points for connectivity.

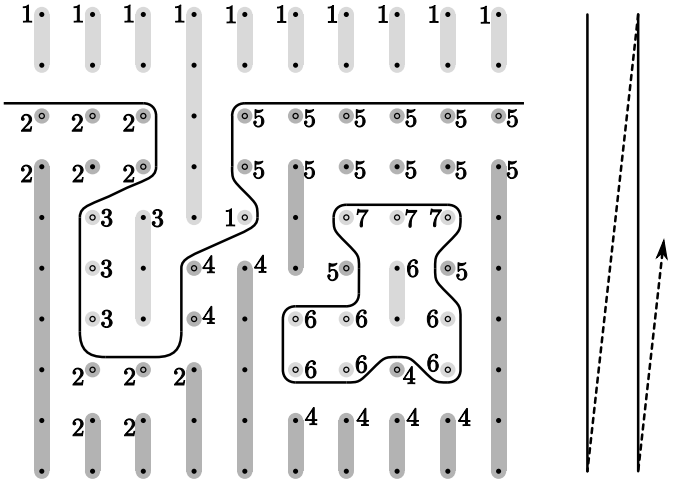

Fig. 2. The surface and a void are represented by a level set. Using sequential iterators the corresponding HRLE data structure can be efficiently processed in lexicographical order (arrow) to set up the reduced graph. Each segment is assigned to a vertex of the graph.

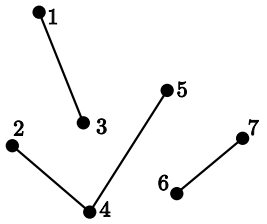

Fig. 3. The corresponding reduced graph as set up by our algorithm consisting of 3 connected components $\{1,3\}$, $\{2,4,5\}$, and $\{6,7\}$, which correspond to the regions above and below the surface, and the void. The number of vertices is very small compared to the number of defined grid points.

To set up the reduced graph an array is needed to store for each segment in the HRLE data structure a reference to the corresponding vertex. The HRLE data structure is sequentially traversed and for each segment the following two tasks are performed:

1) The 6 neighboring points of the first grid point in the current segment are tested for connectivity. If none of the corresponding connected neighbor segments is assigned to a vertex, a new vertex is inserted into the graph to which the current segment is assigned. Otherwise, the current segment is assigned to any vertex to which a connected neighbor belongs.
2) All connected neighbor segments which do not belong to any vertex yet are assigned to the same vertex as the current segment. If there is any connected neighbor belonging to a different vertex, a new edge between the corresponding vertices is inserted in the graph.

Fig. 2 shows an example with a level set representing the surface and a void. After the procedure each segment is assigned to a vertex of the reduced graph which is depicted in Fig. 3. Due to the incorporation of connectivity relations during the setup the number of vertices of the reduced graph is usually only a fraction of the number of defined grid points.

C. Algorithmic complexity

The setup of the graph requires the lexicographical traversal over the HRLE data structure. For the first point of each segment the 6 neighbor grid points have to be found and tested for connectivity. To avoid the logarithmic random access, it was proposed to use 6 additional offset iterators which are moved simultaneously over the data structure [11], [12]. As result, access to neighbor grid points can be performed in constant time. Hence, the setup of the reduced graph has a complexity of $\mathcal{O}(N)$, if N denotes the surface size. For the size of the reduced graph $E + V \leq \mathcal{O}(N)$ holds, since each segment in the HRLE data structure leads to the insertion of at most one vertex and 6 edges. As already mentioned, the connected components of a graph can be obtained with linear complexity, which leads to an overall algorithmic complexity of $\mathcal{O}(N)$.

The memory requirements are optimal. For each segment of the HRLE data structure a reference of the corresponding vertex has to be stored. The memory requirements for the reduced graph can be usually neglected, because in practice the number of vertices is much smaller than the number of segments.

D. Preservation of voids

The connectivity information can be used to ensure that voids do not change over time. If the level set values of all active grid points which do not belong to and which are not connected to any neighbor grid point belonging to the region above the surface, are not changed, the shapes of all voids are maintained. Depending on the orientation of the surface the region above the surface is represented by the connected component which contains the first or the last defined grid point in the HRLE data structure.

IV. Results

To test our algorithm we use a two-layer structure as given in Fig. 4. First, the structure is exposed to an isotropic etching process (Fig. 5). The different material regions with different etching rates are accurately described by our recently developed multi-level-set framework [12]. Afterwards, a conformal deposition process is applied (Fig. 6), where the void detection algorithm is utilized every time step to preserve the shape of the voids.

To proof the linear scaling laws of our algorithms, the initial geometry is scaled by various factors. Table I lists the corresponding lateral grid extensions, the average calculation times for a time integration step during etching and deposition, respectively, and the average computation time for the void detection algorithm. All calculations are performed on an Intel Core 2 Quad Q9550 processor (2.83GHz). The number of vertices of the largest graph, which was set up during the whole simulation, is also given and shows that it is several orders of magnitude smaller than the number of defined grid points (which is at least larger than the product of the lateral grid extensions). Therefore, the memory requirements for storing the reduced graph are marginal.

TABLE I

Scale factor	0.5	1.0	1.5	2.0	2.5
Lateral grid extensions	600×100	1200×200	1800×300	2400×400	3000×500
Average calculation time for a time step (etching)	0.27s	1.09s	2.53s	4.51s	7.15s
Average calculation time for a time step (deposition)	0.34s	1.40s	3.32s	5.25s	9.38s
Average calculation time for void detection (deposition)	0.09s	0.31s	0.73s	1.29s	2.06s
Maximum number of vertices in the reduced graph	76	160	271	390	538

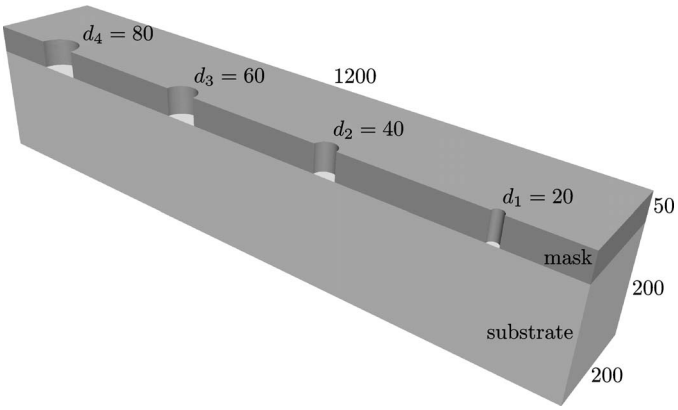

Fig. 4. The initial geometry consisting of a substrate and a mask with cylindrical holes of varying diameters d_i. All lengths are given in multiples of the grid spacing. Reflective boundary conditions are used for both lateral directions.

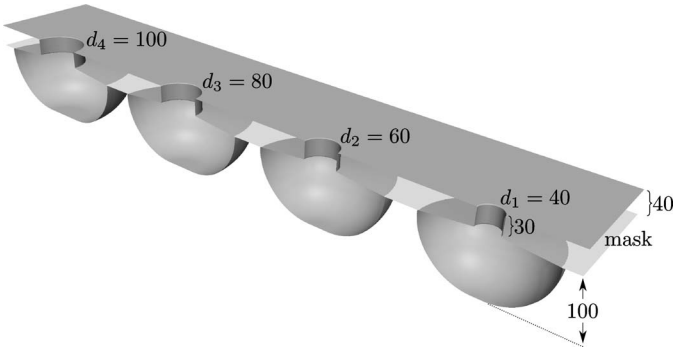

Fig. 5. First an isotropic etching process is applied. Mask etching is also incorporated. An etch rate ratio of 1:10 is assumed. The two material regions are represented by two level sets.

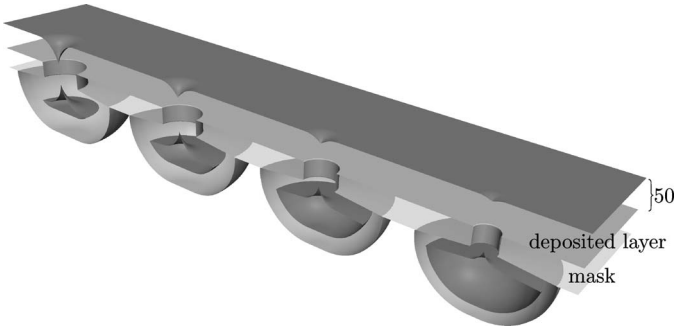

Fig. 6. The final profile after applying a conformal deposition process. The deposited layer has a thickness of 50. Due to the varying hole diameters the voids form at different points of time leading to different thicknesses of the deposited layer within the cavities. The geometry is described by three level sets.

V. Conclusion

We presented a fast void detection algorithm based on the HRLE data structure. Its properties allow the efficient determination of connected components by setting up a reduced graph. The linear complexity of the algorithm was analyzed and demonstrated on an example. The algorithm is not restricted to deposition processes. It can also be used, for instance, to simulate isotropic etching of materials with inclusions. As soon as the etch front reaches an inclusion, its surface is also attacked.

Although used in combination with the level set method, the void detection algorithm can also be applied to the equivolume rate model [2]. There, the HRLE data structure could be used to store the volume rates of surface cells, while run-length encoding bulk and air cells. As consequence, the connected components can be found in an analogous manner.

References

[1] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988.

[2] M. Fujinaga and N. Kotani, "3-D topography simulator (3-D MULSS) based on a physical description of material topography," *IEEE T. Electron. Dev.*, vol. 44, no. 2, pp. 226–238, 1997.

[3] O. Kwon, H. Jung, Y. t. Kim, I. Yoon, and T. Won, "Level-set modeling of sputter deposition," *J. Korean Phys. Soc.*, vol. 40, no. 1, pp. 72–76, 2002.

[4] ——, "Three-dimensional level set based Bosch process simulations using ray tracing for flux calculation," *Microelectronic Engineering*, 2009, doi: 10.1016/j.mee.2009.05.011.

[5] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[6] U.-H. Kwon and W.-J. Lee, "Three-dimensional deposition topography simulation based on new combinations of flux distribution and surface representation algorithms," *Thin Solid Films*, vol. 445, no. 1, pp. 80–89, 2003.

[7] D. Adalsteinsson and J. A. Sethian, "The fast construction of extension velocities in level set methods," *J. Comput. Phys.*, vol. 148, no. 1, pp. 2–22, 1999.

[8] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge Univ. Press, 1999.

[9] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *J. Comput. Phys.*, vol. 118, no. 2, pp. 269–277, 1995.

[10] R. T. Whitaker, "A level-set approach to 3d reconstruction from range data," *Int. J. Comput. Vision*, vol. 29, no. 3, pp. 203–231, 1998.

[11] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth, "Hierarchical RLE level set: A compact and versatile deformable surface representation," *ACM Trans. Graph.*, vol. 25, no. 1, pp. 151–175, 2006.

[12] O. Ertl and S. Selberherr, "A fast level set framework for large three-dimensional topography simulations," *Comput. Phys. Commun.*, 2009, doi: 10.1016/j.cpc.2009.02.002.

[13] J. Siek, L.-Q. Lee, and A. Lumsdaine, *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional, 2002.