Computer Physics Communications 180 (2009) 1242–1250

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

A fast level set framework for large three-dimensional topography simulations

Otmar Ertl [*], Siegfried Selberherr

Institute for Microelectronics, TU Wien, Gusshausstrasse 27-29/E360, A-1040 Wien, Austria

ARTICLE INFO

Article history:
Received 20 August 2008
Received in revised form 2 February 2009
Accepted 7 February 2009
Available online 11 February 2009

PACS:
02.70.-c
81.65.-b
85.40.-e

Keywords:
Topography simulation
Deposition
Etching
Sparse field
Level set method
Run-length encoding

ABSTRACT

We present fast methods to describe the surface evolution of large three-dimensional structures. Based on the sparse field level set method and the hierarchical run-length encoding level set data structure optimal figures for the computation time and for the memory consumption are achieved. Furthermore, we introduce a new multi-level-set technique, which is able to incorporate multiple material regions, and which can also handle material specific surface speeds accurately. We also describe an optimal algorithm for the visibility check for unidirectional etching. The presented techniques are demonstrated on various examples.

1. Introduction

For the simulation of topographic manufacturing steps in semiconductor processing, such as deposition and etching, numerous methods were applied in the past to describe the surface evolution over time [1]. Segment-based models track individual points which are connected among each other by segments [2]. Due to topographic changes it is necessary to thin out or to introduce new surface points. This is a difficult task, especially in three dimensions. Other methods for describing the surface evolution, which base on cells, are the cellular model [3], the building block model [4], the equi-volume rate model [5], or the cellular automata model [6]. A well established method for surface evolution is the level set method [7]. It describes the surface $\mathcal{S} = \partial\mathcal{M}$ of a region $\mathcal{M}$ implicitly by a continuous function $\Phi(\vec{x})$. The surface can then be obtained as the zero level set

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

As convention, we define for the sign of the level set function

$$\Phi(\vec{x}) \leqslant 0 \quad \Leftrightarrow \quad \vec{x} \in \mathcal{M}. \tag{2}$$

The time evolution of the surface can be easily described by the level set equation

$$\frac{\partial\Phi}{\partial t} + V(\vec{x})\|\nabla\Phi\| = 0, \tag{3}$$

where $V(\vec{x})$ denotes the normal component of the extended surface velocity field [8]. If the velocity field is known, the level set method allows a simple calculation of the surface evolution by means of solving (3) on regular grids. Thereby topographic changes can be handled without special consideration. Originally, the level set values were stored and updated in each time integration step for all grid points. Therefore, a complexity of order $\mathcal{O}(N^{3/2})$ can be expected in three dimensions, where N denotes the surface size. Several techniques were introduced to achieve a linear scaling $\mathcal{O}(N)$.

2. Sparse field level set method

To reduce the calculation time down to $\mathcal{O}(N)$ the narrow band method was developed [9]. With this technique only grid points which are close to the surface are updated. These points are called active points. A further development of this method is the sparse field method [10], which was first introduced to topography simulation in [11]. The sparse field level set method updates just one

* Corresponding author.
E-mail addresses: ertl@iue.tuwien.ac.at (O. Ertl), selberherr@iue.tuwien.ac.at (S. Selberherr).

0010-4655/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2009.02.002

layer of active grid points in time. The set of active grid points L_0 is defined as

$$L_0 := \left\{\vec{p} \in P\colon\ -\frac{1}{2} \leqslant \Phi(\vec{p}) \leqslant \frac{1}{2}\right\}, \tag{4}$$

where $P \subseteq \mathbb{Z}^3$ denotes the set of all grid points. Additional layers of grid points are necessary to calculate first or higher order derivatives at active grid points, which are required for conventional time integration schemes. These layers can be expressed as

$$L_n := \left\{\vec{p} \in P\colon\ \operatorname{sgn}\bigl(\Phi(\vec{p})\bigr) \cdot \min_{\vec{p}' \in L_0} \|\vec{p} - \vec{p}'\|_1 = n\right\} \tag{5}$$

using the Manhattan norm $\|\cdot\|_1$. First order derivatives require all grid points of layers $L_{\pm 1}$, while also those of layers $L_{\pm 2}$ are necessary for second order derivatives. After updating the level set values of all active grid points $\vec{p} \in L_0$ using an appropriate time integration scheme, grid points of neighboring layers are processed in the order $L_{\pm 1}, L_{\pm 2}, \ldots$. The level set values of grid points in layer $L_{\pm n}$ only depend on the values of the next inner layer $L_{\pm(n-1)}$. The level set values are updated using a simple scheme

$$\Phi(\vec{p} \in L_n) = \begin{cases} \min_{\vec{p}' \in \eta(\vec{p}) \cap L_{n-1}} \Phi(\vec{p}') + 1 & \text{if } n > 0, \\ \max_{\vec{p}' \in \eta(\vec{p}) \cap L_{n+1}} \Phi(\vec{p}') - 1 & \text{if } n < 0. \end{cases} \tag{6}$$

Here $\eta(\vec{p})$ denotes the set of grid points which are neighbors of $\vec{p}$.

Finally, the set of active grid points L_0 is refreshed using (4). The sets of grid points for the neighboring layers are also updated using (5). Due to the update scheme (6) and due to (4)

$$L_n = \begin{cases} \{\vec{p} \in P\colon\ n - \frac{1}{2} \leqslant \Phi(\vec{p}) < n + \frac{1}{2}\} & \text{if } n < 0, \\ \{\vec{p} \in P\colon\ -\frac{1}{2} \leqslant \Phi(\vec{p}) \leqslant \frac{1}{2}\} & \text{if } n = 0, \\ \{\vec{p} \in P : n - \frac{1}{2} < \Phi(\vec{p}) \leqslant n + \frac{1}{2}\} & \text{if } n > 0 \end{cases} \tag{7}$$

holds. Therefore, for time integration schemes using first and second order derivatives it is sufficient to consider only level set values up to an absolute value of $\frac{3}{2}$ and $\frac{5}{2}$, respectively.

The solution of the level set equation (3) with finite difference schemes requires the observance of a Courant-Friedrichs–Lewy (CFL) condition to guarantee stability [7]. It limits the maximum advancement of the surface measured in grid spacings. In case of the sparse field level set method, the CFL condition limits the maximum change of all level set values

$$\max_{\vec{p} \in L_0} \left|\Phi^{(t+\Delta t)}(\vec{p}) - \Phi^{(t)}(\vec{p})\right| \leqslant \Delta\Phi_{\text{CFL}}. \tag{8}$$

For the following considerations we demand

$$\Delta\Phi_{\text{CFL}} < \frac{1}{2}, \tag{9}$$

which implies that only the level set values of active grid points can change their signs during one time integration step. This statement follows from (4) and (6).

The sparse field level set method saves a lot of computation time in comparison to the narrow band method. First of all, only a minimal set of active grid points is involved in the time integration procedure. Furthermore, the time consuming surface velocity extension can be avoided. This extension is necessary for topography simulations, since the velocities are only defined on the surface and the level set method requires a velocity field [8]. Finally, the sparse field level set method does not require periodic re-initializations like the narrow band method [9].

2.1. Sets of active grid points

In [10] a set of active grid points is defined as efficient, if all active grid points have at least one neighbor with opposite signed level set value. The sparse field level set method does not maintain efficient sets of active grid points over time. Two examples, where active grid points without opposite signed neighbor are produced by the update scheme, are shown in Fig. 1. These points often appear in regions, where the surface converges, and do not necessarily have to be close to the surface afterwards, as illustrated by the second example. The consideration of these unnecessary grid points makes the expansion of the surface velocity field more complicated and computationally more intensive. Even worse, dense sets of such points may be produced, essentially increasing the memory consumption and the calculation time during time integration. To circumvent all these problems a pruning procedure was proposed [10], which eliminates all active points which do not have an opposite signed neighbor, after each time step.

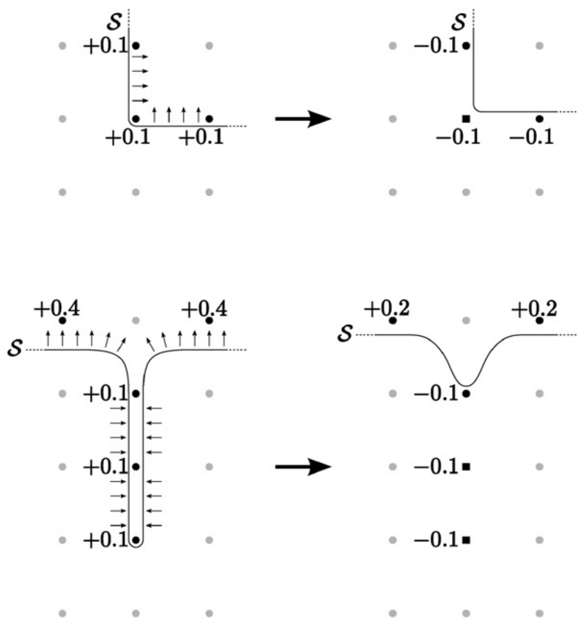

Fig. 1. Two examples, where the sparse field level set method produces inefficient sets of active (black) grid points. The surface $\mathcal{S}$ moves with a uniform positive surface velocity (arrows). After a time step some active grid points (squares) do not have any neighbor with opposite signed level set value.

A requirement of the sparse field level set method is that in each pair of neighboring grid points with opposite level set signs, there is at least one active grid point

$$\begin{aligned} &\vec{p}' \in \eta(\vec{p}) \wedge \Phi(\vec{p}')\Phi(\vec{p}) \leqslant 0 \\ &\quad \Rightarrow \quad \left|\Phi(\vec{p})\right| \leqslant \frac{1}{2} \vee \left|\Phi(\vec{p}')\right| \leqslant \frac{1}{2}. \end{aligned} \tag{10}$$

In other words, the sets of positive and negative grid points have to be always separated by active grid points. However, in very seldom cases (10) can be violated as exemplified in Fig. 2. To get a robust algorithm it is necessary to resolve such situations by reducing the level set values of both involved grid points to $\frac{1}{2}$ or $-\frac{1}{2}$, respectively.

2.2. Initialization

The initial data, which is necessary for the sparse field level set method, must include at least the level set values of all active grid points as well as the signs of all other grid points. If this information is available, it is possible to calculate all other level set values following (5) and (6).

It is sufficient to provide the level set values of all grid points, which are vertices of grid line segments intersected by the initial surface. These points with level set values in the range of $[-1, 1]$

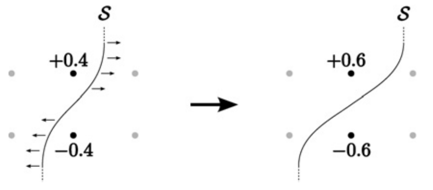

Fig. 2. A special case which can occur within the sparse field level set method, and which needs special consideration: Two neighboring active grid points (black) with opposite signed level set value are updated in time. Due to the given surface velocities (arrows), both grid points loose their activity status, leading to a pair of non-active neighboring grid points with opposite signs.

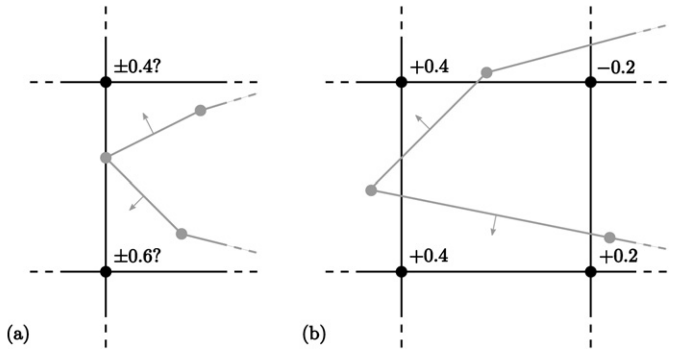

Fig. 3. (a) Two elements of the surface mesh (gray) meet on a grid line (black). Hence, for both grid points the distance is equal to both elements. As consequence the determination of the signed distance according to (11) is ambiguous. (b) The distance transformation can produce inefficient sets of active grid points. The bottom left grid point does not have an opposite signed neighbor.

clearly separate all grid points into regions with positive and negative level set values, respectively. Starting from these initial grid points the level set values of all other grid points can be straightforwardly determined using the update scheme (6). Hereby, the sign can be obtained by taking the sign of a neighbor grid point belonging to the next inner layer.

Hence, what is needed at the very beginning is a list of the coordinates of all required grid points together with their initial level set values. Usually the initial geometry is given as triangulation. A distance transformation is necessary to calculate the level set values. Since we only need the level set values of nearby grid points, the initialization can be performed in an efficient manner.

In case of the sparse field level set method it is more beneficial to use the smallest Manhattan distance rather than the smallest Euclidean distance for the initialization. With the latter, the first time step of the sparse field level set method gives wrong results for the position of a (non-axis-parallel) plane moving with constant speed. However, if initialized with the Manhattan distance, tri-linear interpolation of the level set function describes correctly the position of the plane after the first time step. The reason for this behavior is the update scheme (6), which also corresponds to a Manhattan distance calculation.

The Manhattan distance computation can be essentially simplified for our purpose. For each grid point $\vec{p}$ we check each grid line going through $\vec{p}$ and with direction $k \in \{x, y, z\}$ for intersections with elements $\triangle$ of the surface triangulation. If this is the case, the signed distance can be calculated according to

$$d(\vec{p}, \Delta, k) = \operatorname{sgn}(n_k) \cdot |p_k - q_k|, \tag{11}$$

where $\vec{q} = (q_x, q_y, q_z)$ is the intersection point with the triangle $\triangle$. Since we only need all grid points with an opposite signed neighbor, triangles can be excluded with an (absolute) distance greater than $1 + \varepsilon_1$. A small positive constant ε_1 is added for numerical reasons.

The signed distance of the closest triangle is finally assigned to the initial level set value $\Phi(\vec{p})$. However, choosing the closest triangle according to $|d(\vec{p}, \triangle, k)| = |p_k - q_k|$ can lead to problems, as depicted in Fig. 3a, where the wrong sign could be assigned to the grid points since both triangles are equally distanced. To get the right sign without additional consideration of neighbor triangles, we measure the distance using

$$d'(\vec{p}, \Delta, k) = \operatorname{sgn}(p_k - q_k) \cdot (p_k - q_k - \varepsilon_2 \cdot t_k) \tag{12}$$

to find the closest triangle. Here $\vec{t} = (t_x, t_y, t_z)$ denotes the unit vector pointing from $\vec{q}$ to the centroid of the triangle and ε_2 is again a small positive constant. However, the distance which is finally assigned to $\Phi(\vec{p})$ is still calculated using (11).

The whole initialization algorithm is realized by an iteration over all triangles. Then the grid lines which may intersect the triangle are confined by calculating the bounding box of the triangle. If a grid line intersects a triangle, all grid points on the grid line with an (absolute) distance smaller than $1+\varepsilon_1$ are determined. For these grid points the indices $\vec{p}$ are stored together with the corresponding values for the distances d and d' in a list. Finally the list is lexicographically sorted according to the indices $\vec{q}$. If there are more entries with same $\vec{q}$ (which is not very often the case), d of that entry with the smallest d' is used for $\Phi(\vec{p})$.

The whole initialization has an algorithmic complexity of $\mathcal{O}(N \log N + M)$, where N denotes the number of grid points nearby the surface and M the number of triangles. The initialization algorithm can produce inefficient sets of active grid points (Fig. 3b) which can be avoided by appending a pruning procedure as mentioned earlier.

3. Hierarchical run-length encoding

To reduce the memory consumption of the level set method various techniques, as for example oct-trees [12], were introduced. Recently, the hierarchical run-length encoded (HRLE) level set data structure was developed [13]. This data structure efficiently stores the level set values of a subset of all grid points in lexicographical order. For all other undefined grid points only the signs of the level set values are stored using run-length compression. Subsequent undefined grid points with same sign are combined in runs as shown in Fig. 4. The run-length encoding is successively applied to all grid directions. The result is a hierarchical data structure with a linear scaling memory consumption with little overhead. The data structure uses arrays for its internal representation. The availability of the signs of all grid points is very useful, since it gives the information on which side of the surface a grid point is located. The sign of the level set function is also important for multi-level-set methods, which are described later.

In the following we show that the HRLE data structure in conjunction with the sparse field level set method are powerful methods to obtain fast linear scaling algorithms useful for three-dimensional topography simulations. We implemented exactly the same data structure as described in [13], where a detailed description of the HRLE data structure can be found.

3.1. Setup of data structure

The HRLE data structure can be setup by inserting all defined grid points (i, j, k) in lexicographical order (with reversed significance (k, j, i) to be consistent with [13]) together with their level set values. Undefined runs are automatically created if grid points are skipped. To avoid ambiguities concerning the sign of undefined runs, our setup algorithm needs at least the level set values of all grid points with an opposite signed neighbor, as obtained by our initialization algorithm (see Section 2.2). In this way, the right sign of undefined runs can be derived from the previous or from the next inserted defined grid point. If the points are already sorted, the data structure can be setup with optimal linear complexity.

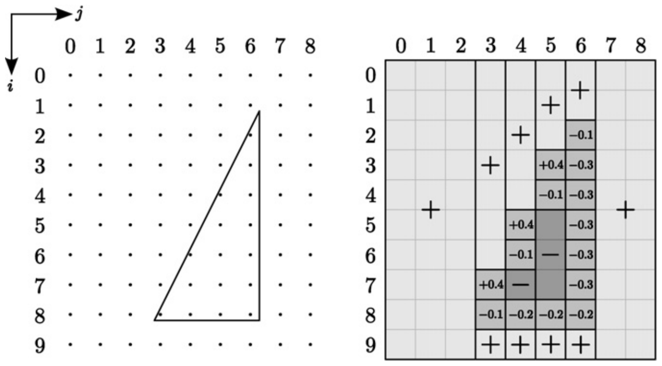

Fig. 4. The level set function of a triangle represented by an HRLE data structure for a 10×9 grid. In this example only the level set values of active grid points are stored explicitly. Undefined grid points with same signs are combined in runs. Light and dark gray runs correspond to positive and negative level set values, respectively.

The successive insertion of grid points at the end of the data structure only requires the modification of the ends of dynamic arrays, which can be realized with constant complexity using, for instance, STL (Standard Template Library) vectors [14].

3.2. Data access

The HRLE data structure allows fast random access to grid points with a worst case complexity of $\mathcal{O}(\log N)$, while sequential access can be performed with constant complexity. For the sequential access we implemented an iterator, which stops at every defined grid point and also at every undefined run. The iterator can be asked for the range of grid point indices, which the current run contains. The iterator provides functions to retrieve the minimum and maximum index vector. If the iterator is at the position of a defined grid point, both index vectors are identical. The iterator gives access to the level set value of the current grid point(s). In case of undefined runs $\pm\infty$ (or an adequate numerical representation) is returned as level set value. After initialization the iterator refers to the first undefined run or defined grid point in the HRLE data structure. Functions for advancing to the next position and for checking if the end is reached, allow a simple iteration over the whole data structure. Fig. 5b shows a HRLE data structure with sequentially numbered undefined runs and defined grid points. The numbering corresponds to the traversal sequence of the iterator. As example, an iterator at the second position refers to the positive undefined run which contains the grid points $(0, 2)$, $(1, 2)$, and $(2, 2)$.

The level set method requires access to neighbor grid points for the calculation of derivatives. To find the correct neighbor grid points the given boundary conditions must be incorporated. We allow periodic, reflective, and infinite boundaries, which can be specified for each axis direction independently. Infinite boundaries are actually realized by reflective boundaries and setting the domain extensions to very large positive and negative values. As example, Fig. 5a shows a small simulation domain with reflective and periodic boundaries for different directions.

In order to obtain linear scaling algorithms, in spite of neighbor access, stencils of iterators can be used and moved simultaneously over the whole data structure [13]. For this purpose our iterator allows the definition of an offset. The behavior of an iterator with a given offset equals an iterator moving over the HRLE data structure, which is obtained by shifting all undefined runs and defined grid points according to the given offset with simultaneous consideration of the boundary conditions. Figs. 5c and 5d show the HRLE data structure as seen for iterators with offsets $(-1, -1)$ and $(2, 1)$, respectively. The $(-1, -1)$-iterator gives access to the $(-1, -1)$-neighbors for all grid points with indices in the range

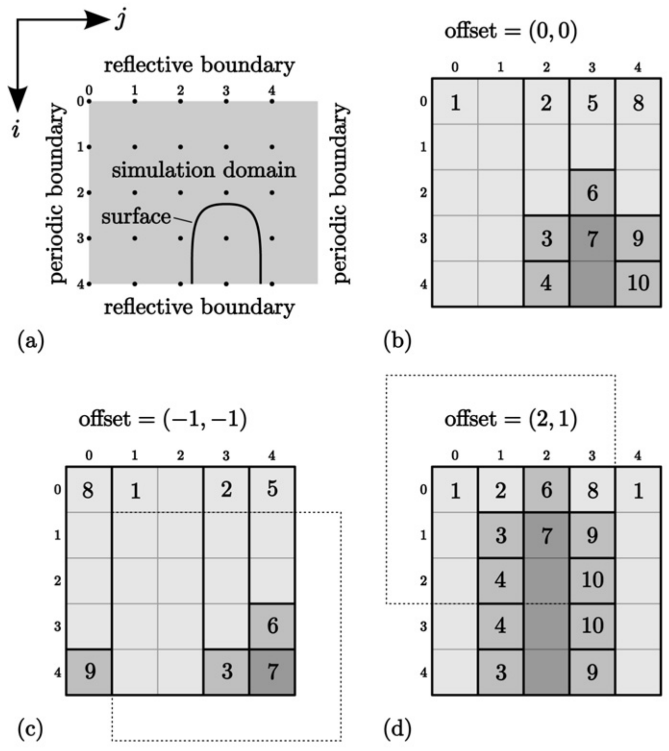

Fig. 5. (a) A surface embedded in a domain with extensions 4×5. However, due to the different boundary conditions in i-direction (reflective) and j-direction (periodic) 5×5 grid points are used for the discretization of the level set function. (b) The corresponding HRLE data structure with serially numbered defined grid points (medium gray), positive (light gray) and negative (dark gray) undefined runs. For iterators without offset the traversal order equals the numbering. (c) The undefined runs and defined grid points as seen by an iterator with offset $(-1, -1)$. (d) The same for an iterator with offset $(2, 1)$.

given by the minimum and maximum indices returned by the iterator. For example, the 4th position of the $(-1, -1)$-iterator refers to the positive defined run with number 2. The minimum and maximum index vector are $(0, 3)$ and $(3, 3)$, respectively, which means that the referenced run contains the $(-1, -1)$-neighbors for the grid points $(0, 3)$, $(1, 3)$, $(2, 3)$, and $(3, 3)$. Using Fig. 5b it can be easily verified that for these grid points the $(-1, -1)$-neighbors really belong to the positive undefined run with number 2.

Our implementation realizes an offset iteration with linear complexity. Due to the offset the iterator does not traverse the HRLE data continuously. For example, there is a jump for the $(-1, -1)$-iterator if it is moved from position 2 to 3 (Fig. 5c). The referenced run changes abruptly from 9 to 1. In such cases a binary search within the HRLE data structure to find the next position may be necessary. However, on average it is possible to advance the iterator in constant time. In case of reflective boundary conditions it might be necessary to reverse the traversal direction, as shown for the transition of the $(2, 1)$-iterator from the 5th to the 6th position (Fig. 5d).

To obtain linear complexities for algorithms requiring neighbor access, several offset iterators can be grouped and moved simultaneously over the data structure. As example, an iterator for first order finite differences can be built from 7 iterators with offsets $\{(0, 0, 0), (\pm1, 0, 0), (0, \pm1, 0), (0, 0, \pm1)\}$. The group of iterator is traversed as follows. In each iteration step the common range of grid point indices, which are contained in the actual runs of all iterators, is determined. This range defines the set of grid points, for which the current iterator positions apply. Then only those iterators, whose maximum index vector is equal to the upper bound of the common range of index vectors, are moved forward and so forth.

The inherent incorporation of different boundary conditions and the possibility to form different stencils of iterators allow a simple implementation of algorithms for the HRLE data structure.

3.3. Sparse field implementation

After each time integration step the level set function has to be rebuilt, because due to the movement of the surface, some active grid points become inactive and vice versa. Therefore, also the layers of defined grid points have to be altered accordingly. Since arrays are used for storing the data structure, it is inefficient to adapt the data structure by inserting and deleting grid points. Instead, we construct a new level set data structure on-the-fly, by iterating over the existing data structure and inserting defined grid points at the end. Thus, the whole data structure is setup from new. Actually, each time step the level set data structure is reconstructed several times. In the following we describe the procedure, which realizes the sparse field level set method using the HRLE data structure with linear complexity.

3.3.1. Time integration

First the level set values of all active grid points are changed according to the applied time integration scheme. A stencil of iterators as previously described is used to enable the calculation of finite differences. A first iteration is necessary to determine the maximum possible time step fulfilling (8). During a second iteration the level set values of active grid points are updated in time and inserted into a new level set data structure. All other defined grid points are skipped. However, the signs of all non-active points, which do not change during the time integration step due to (9), are maintained and stored in the HRLE data structure.

3.3.2. Pruning and consistency check

As previously mentioned a pruning procedure is necessary to avoid dense sets of active grid points. Again, a stencil of iterators, like that for first order differences, is used and moved over the data structure, which is then constructed from new again. The whole stencil stops whenever the central iterator reaches a defined grid point. Grid points are skipped, which do not have a neighbor with opposite signed level set value. The level set values of all other grid points are transfered to the new level set data structure. However, before inserting they are checked, if their level set values are greater than $\frac{1}{2}$, while that of one neighbor grid point is less than $-\frac{1}{2}$ or vice versa. If this is the case, the prerequisite of the sparse field level set method (10) is violated. To guarantee the robustness of the algorithm the level set value must be reduced to $\pm\frac{1}{2}$ (while keeping its sign) before insertion into the new level set data structure.

3.3.3. Dilation

Finally, the set of defined grid points has to be dilated using the update scheme (6). Again a stencil of iterators is moved over the structure simultaneously. However, this time the iterator stops whenever any iterator of the stencil reaches a defined grid point. If the position of the central iterator is a defined grid point, it is passed unchanged to the new data structure. Otherwise, the level set value is first determined using the update scheme (6). Depending on how many layers of defined grid points are needed for the next time integration step this task has to be repeated several times.

For example, if second order approximations of the gradient are necessary, the explicit level set values of all grid points in layers L_0, $L_{\pm1}$, and $L_{\pm2}$ are required. Due to (7) it is sufficient to treat all points with absolute values less than or equal to $\frac{5}{2}$. For this dilation three cycles are necessary. During the first, second, and third cycle all grid points with absolute level set values less or equal to $\frac{1}{2}$, $\frac{3}{2}$, and $\frac{5}{2}$ are added, respectively. Due to (4) and due to (9) it is also possible to expand the level set structure up to absolute level set values of 1 during the first cycle. However, then still 2 more iterations are needed to expand the structure up to $\frac{5}{2}$. The performance is improved, if defined grid points are added as late as possible to speed up the successive iterations.

Furthermore, the pruning and consistency check as described in the previous section can be included during the first dilation cycle to accelerate the algorithm.

3.4. Boolean operations

The intersection or union of regions enclosed by two level set functions can be expressed as the maximum or minimum of both functions [15,16], if the sign convention (2) is used. These operations are useful for more general topography simulations, where consecutive process steps, like etching and/or deposition processes, should be simulated or several materials are involved.

If single iterators for each level set function are used and simultaneously moved over both HRLE data structures, while stopping at each defined grid point, a linear complexity $\mathcal{O}(N_a + N_b)$ can be obtained for these operations. Here N_a and N_b are the number of defined grid points of both level set functions.

Boolean operations can again result in inefficient sets of active grid points in terms of the sparse field level set method. Therefore, we apply the previously described pruning procedure after each boolean operation.

3.5. Unidirectional visibility test

For processes with unidirectional fluxes the HRLE data structure can be used for a very efficient visibility test. To determine the surface velocities, we have to distinguish for all active grid points, whether its surface velocity is affected by the directional flux. If this is the case, the point is called visible. We assume that the flux is in positive i-direction (x-direction). Then, an active grid point with indices (i, j, k) is visible, if the level set values of all grid points in $\{(i', j, k)\colon\ i' < i\}$ are greater than or equal to that of grid point (i, j, k). Hereby, the level set values of undefined grid points are assumed to be $\pm\infty$. Fig. 6 illustrates by means of an example which active grid points are visible. Taking advantage of the lexicographical ordering with reversed significance, it is sufficient to iterate once over the HRLE data structure to determine the visibilities for all active grid points. Hence, this visibility test is of optimal complexity $\mathcal{O}(N)$, where N again denotes the number of defined grid points, which is proportional to the surface size.

If a grid point is visible, the local flux can be determined by

$$F(\vec{p}) = F_{\text{src}} \cdot \max\bigl(0, -n_x(\vec{p})\bigr), \tag{13}$$

where F_{src} denotes the incident flux from the source, and n_x is the component in i-direction (x-direction) of the normal vector calculated by

$$\vec{n}(\vec{p}) = \bigl(n_x(\vec{p}), n_y(\vec{p}), n_z(\vec{p})\bigr) = \frac{\nabla\Phi(\vec{p})}{\|\nabla\Phi(\vec{p})\|}. \tag{14}$$

The local fluxes can then be used to calculate the surface velocities for the active grid points.

3.6. Surface extraction

To visualize a surface represented by the HRLE level set data structure we use the marching cubes algorithm [17]. This algorithm is realized with a cell-shaped stencil of 8 iterators. This group of iterators is moved in parallel over the data structure. Whenever, at

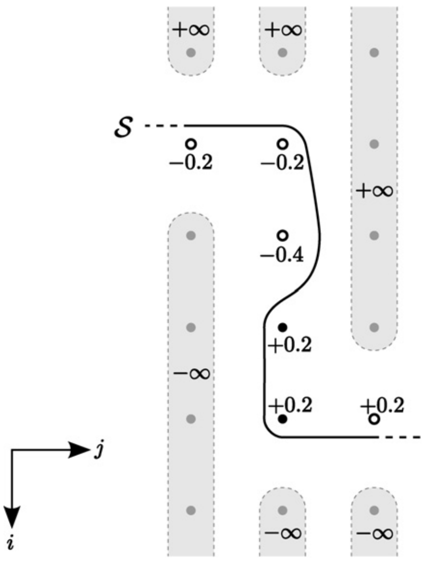

Fig. 6. The surface $\mathcal{S}$ and its HRLE representation are shown. In this example, the level set values are only defined for the active grid points. Positive and negative runs of undefined grid points (gray) are assumed to have level set values $+\infty$ and $-\infty$, respectively. The unfilled grid points fulfill the visibility criterion.

least one iterator reaches the position of an active grid point, the current grid cell is processed by the marching cubes algorithm. The result of this approach is a fast surface extraction method.

4. Multiple level sets

Sometimes topography simulations require the treatment of different materials. Especially, etch processes require the distinction of different material regions, as for example mask and substrate, where different etch rates have to be applied. Hence, the geometric information of material regions is necessary during time evolution. Usually the initial geometry is given as triangulated mesh, where each element is assigned to a certain material. In case of etching, this irregular grid has to be accessed many times to query the material region of surface points in order to calculate the correct surface velocities. Therefore, if search trees are used, where the queries are of logarithmic complexity, a non-linear algorithm for time evolution can be expected. If consecutive deposition and etching processes have to be simulated, as for example in Bosch processes, a costly and challenging modification of the irregular mesh is necessary after each process step.

Instead of using an irregular grid for the material regions and a regular grid for the level set function simultaneously, we propose to use multiple level set functions. The initial geometric information is simply mapped from the irregular to the regular grid by means of additional level set functions. In the following we describe a multi-level-set technique using the sparse field level set method and the HRLE data structure. With the ability to resolve the material dependent surface velocities with sub-time-step accuracy, a more accurate final profile is obtained, especially in the presence of thin layers or large etch rate ratios.

4.1. Level set representation

For our considerations we assume that the whole structure $\mathcal{M}$ is composed of K disjoint material regions $\mathcal{M}_k$

$$\mathcal{M} = \bigcup_{k=1}^{K} \mathcal{M}_k \quad \text{and} \quad \mathcal{M}_k \cap \mathcal{M}_l = \emptyset \quad \text{for all } k \neq l. \tag{15}$$

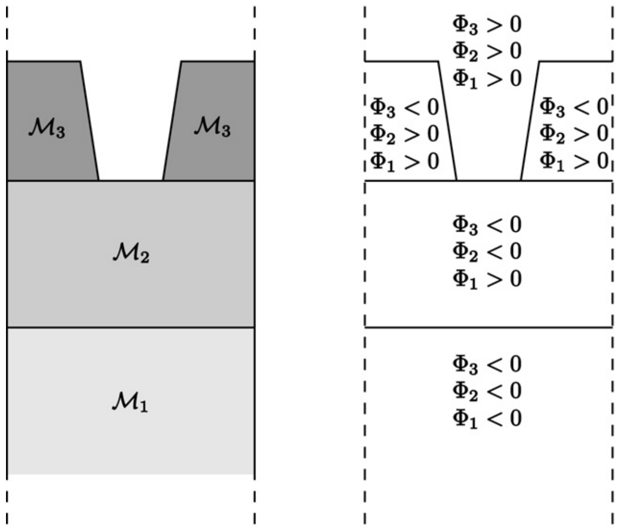

Fig. 7. A geometry consisting of 3 different material and its representation by 3 level set functions. The zero level set of Φ_3 corresponds to the surface. Φ_2 and Φ_1 describe the interfaces $\mathcal{M}_3/\mathcal{M}_2$ and $\mathcal{M}_2/\mathcal{M}_1$, respectively.

There are several possibilities to represent the different material regions by level set functions. One way is to describe each material region $\mathcal{M}_k$ by one enclosing level set function Φ_k [18]

$$\Phi_k(\vec{x}) \leqslant 0 \quad \Leftrightarrow \quad \vec{x} \in \mathcal{M}_k. \tag{16}$$

However, with this representation very thin layers with thicknesses smaller than one grid spacing cannot be resolved, as needed for example to describe thin passivation layers in etching processes. If the passivation layer becomes thinner than the grid spacing, it can vanish abruptly, and the etching of the underlying material would start too early. This can lead to significant errors, especially for large etch rate ratios. To circumvent this problem, we describe a stack of materials $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$, where $\mathcal{M}_1$ denotes the substrate, by choosing K level set functions in such a way that

$$\Phi_k(\vec{x}) \leqslant 0 \quad \Leftrightarrow \quad \vec{x} \in \bigcup_{i=1}^{k} \mathcal{M}_i \tag{17}$$

as demonstrated in Fig. 7. Hence, the level set function Φ_K describes the surface of the whole structure, while the other level set functions represent interfaces. We assume that these level set functions fulfill the inequality

$$\Phi_1(\vec{x}) \geqslant \Phi_2(\vec{x}) \geqslant \cdots \geqslant \Phi_K(\vec{x}). \tag{18}$$

This is the case if they are initialized as distance functions. The number of the material on the surface $\mathcal{S}$ can be obtained by

$$m(\vec{x} \in \mathcal{S}) = \min\{1 \leqslant k \leqslant K\colon\ \Phi_k(\vec{x}) = 0\}. \tag{19}$$

4.2. Time evolution

To calculate the time evolution of the surface, we have to take the different material regions into account. First, we move the surface accordingly to the materials which are on top and which can be obtained from (19). Hereby, we define that a deposition process, which is characterized by positive surface velocities, always increases the thickness of material K. If a new material $K+1$ should be deposited instead, simply a new level set function $\Phi_{K+1}(\vec{x})$ is added and initialized with $\Phi_{K+1}(\vec{x}) = \Phi_K(\vec{x})$.

For etching the surface velocities are negative, and the etch rates of the different materials have to be incorporated during the time evolution of the surface. It is sufficient to update the top most level set function Φ_K in time and to adjust all other level set functions according to the Boolean operation

$$\Phi_k^{(t+\Delta t)}(\vec{x}) = \max\bigl(\Phi_k^{(t)}(\vec{x}), \Phi_K^{(t+\Delta t)}(\vec{x})\bigr). \tag{20}$$

This adaption rule, which also maintains relation (18), is even more general, since it is not restricted to pure etching processes, where the surface velocities are all negative. It is also valid for pure deposition processes and etching processes with simultaneous (re-)deposition of one material type.

There is still the open question, how to change Φ_K under consideration of the different material regions. Common approaches use one single velocity field which is set up in dependence on the material types on the surface [15]. These velocities are used during the whole time integration step. Thus, if another material region is reached, the surface is moved with the wrong velocity. This is especially a problem, if the surface velocities are very different, which is for example the case in presence of masks or etch stop layers.

Our method takes changing velocities with sub-time-step accuracy into account. To explain our approach we define K different velocity fields $V_k(\vec{x})$ $(1 \leqslant k \leqslant K)$, for each material type one. $V_k(\vec{x})$ is the velocity field which would be obtained, if the material type k is on the whole surface, thus $m(\vec{x}) = k$ for all $\vec{x} \in \mathcal{S}$. Many models used in topography simulations use the pseudo-steady state assumption, that the geometry changes slower than the solution of the governing transport equations [19]. Therefore, the solution mainly depends on the current surface profile, which allows an independent determination of the surface velocity fields $V_k(\vec{x})$ for all material types k.

Due to our restriction that only the material type K of the top level region $\mathcal{M}_K$ can be deposited, the surface velocity fields obey

$$\begin{aligned} &V_k(\vec{x}) \in \mathbb{R}_0^- \quad \text{for } 1 \leqslant k < K, \\ &V_K(\vec{x}) \in \mathbb{R}. \end{aligned} \tag{21}$$

For the time evolution of Φ_K we apply for each individual active grid point $\vec{p} \in L_0^{(K)}$ the following update rule:

$$\Phi_K^{(t+\Delta t)}(\vec{p}) = \Phi_K^{(t)}(\vec{p}) - \sum_{k=1}^{K} \Delta t_k(\vec{p}) \cdot \hat{H}\bigl(V_k, \Phi_K^{(t)}, \vec{p}\bigr). \tag{22}$$

Here $\hat{H}(V_k, \Phi_K^{(t)}, \vec{p})$ is the numerical approximation for $V_k(\vec{p}) \times \|\nabla\Phi_K^{(t)}(\vec{p})\|$ and depends on the applied finite differencing scheme [15,20]. $\Delta t_k(\vec{p})$ denotes the time for which the surface velocity $V_k(\vec{p})$ is used during time integration. These times can be approximated for a given time integration step Δt by

$$\Delta t_k(\vec{p}) = \begin{cases} \frac{\Phi_k^{(t)}(\vec{p}) - \Phi_{k-1}^{(t)}(\vec{p})}{\hat{H}(V_k, \Phi_K^{(t)}, \vec{p})} & \text{if } k > k'(\vec{p}), \\ \Delta t - \sum_{l=k+1}^{K} \Delta t_l(\vec{p}) & \text{if } k = k'(\vec{p}), \\ 0 & \text{if } k < k'(\vec{p}). \end{cases} \tag{23}$$

For each active grid point $\vec{p}$ the material number $k'(\vec{p})$ is chosen to be the smallest number $(1 \leqslant k'(\vec{p}) \leqslant K)$ for which all $\Delta t_k(\vec{p})$ are non-negative. If $V_K(\vec{p})$ is positive, $k'(\vec{p})$ is always set to K. $k'(\vec{p})$ can be interpreted as the material type which is locally on the surface after the time integration step.

It should be noted that it is not necessary to evaluate each velocity field V_k at all active grid points $\vec{p} \in L_0^{(K)}$. For an active grid point $\vec{p}$ it is sufficient to calculate $V_k(\vec{p})$ for $k'(\vec{p}) \leqslant k \leqslant K$. $V_k(\vec{p})$ with $k < K$ is also not relevant if $\Phi_k^{(t)}(\vec{p}) = \Phi_{k-1}^{(t)}(\vec{p})$. Moreover, if $V_K(\vec{x})$ is never positive, due to the applied model, then also the determination of $V_K(\vec{p})$ can be skipped in the case of $\Phi_K^{(t)}(\vec{p}) = \Phi_{K-1}^{(t)}(\vec{p})$. Hence, for a certain active grid point only the local surface velocities for those materials have to be determined, which are actually involved during the time step.

The time integration step Δt is chosen in such a way that the CFL condition (8) is fulfilled for Φ_K

$$\max_{\vec{p} \in L_0^{(K)}} \bigl|\Phi_K^{(t+\Delta t)}(\vec{p}) - \Phi_K^{(t)}(\vec{p})\bigr| = \Delta\Phi_{\text{CFL}}. \tag{24}$$

Due to the limitation of $\Delta\Phi_{\text{CFL}}$ (9) it is sufficient for the solution of (23) to store only level set values up to an absolute value of 1 within the HRLE data structures for the level set functions $\Phi_1, \Phi_2, \ldots, \Phi_{K-1}$. Undefined grid points are assumed to have level set values $+\infty$ or $-\infty$ depending on their signs.

The multi-level-set method is realized using the HRLE data structure by adapting the time integration procedure as described in Section 3.3.1. While iterating over the surface level set function Φ_K using a stencil of iterators enabling the calculation of the derivatives of Φ_K, $K-1$ additional single iterators are simultaneously moved over the level set representations of $\Phi_1, \Phi_2, \ldots, \Phi_{K-1}$, stopping at each active grid point in $L_0^{(K)}$. These iterators allow the access to the level set values as needed in (23).

5. Examples

For all examples presented in the following we use the second order upwind differencing Engquist–Osher scheme for time integration [15,20]. Therefore, the level set data structure for the surface level set function is always dilated up to all grid points with level set values less or equal $\frac{5}{2}$. For the CFL condition we chose $\Delta\Phi_{\text{CFL}} = 0.1$. All simulations were performed on an AMD Opteron 2222 processor (3 GHz).

5.1. Isotropic deposition

To demonstrate the capabilities of the sparse field level set method in combination with the HRLE data structure we simulated a conformal deposition process. The surface velocity was set to a constant value. The process was applied to a test structure resolved on various grids with lateral extensions 500 × 500, 1000 × 1000 (Fig. 8), 1500 × 1500, 2000 × 2000, 2500 × 2500, and 3000 × 3000

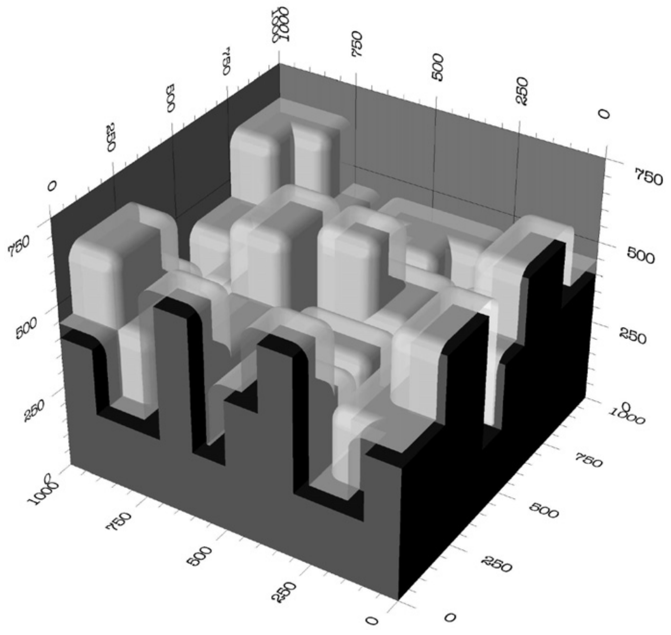

Fig. 8. A conformal deposition process on a test structure (black) resolved on a grid 1000 × 1000. The deposited layer (gray) has a thickness of 45 grid spacings.

Table 1
Performance characteristics of the sparse field level set method using the HRLE data structure. Perfect linear scaling of calculation time and memory consumption with surface size can be observed.

Lateral grid resolution	500 × 500	1000 × 1000	1500 × 1500	2000 × 2000	2500 × 2500	3000 × 3000
Calculation time per time step	4.03 s	15.7 s	36.3 s	64.5 s	101 s	147 s
Used memory	35.3 MB	142 MB	320 MB	569 MB	890 MB	1.25 GB
Number of defined points	3.88M	15.6M	35.2M	62.6M	97.9M	141M
Number of active points	783k	3.14M	7.06M	12.5M	19.6M	28.2M

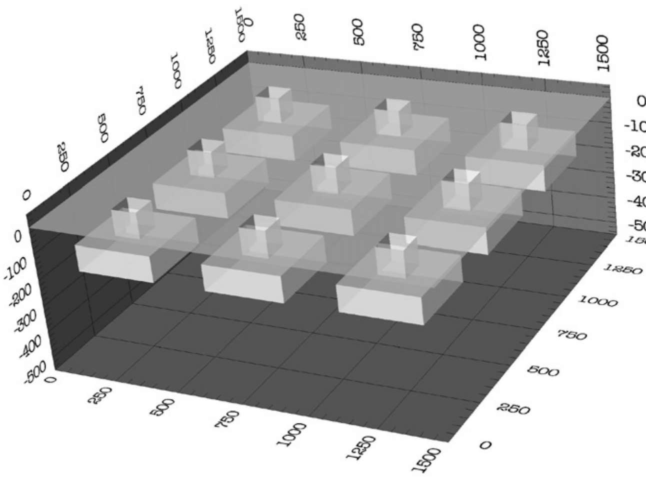
Fig. 9. The initial geometry on which a unidirectional etching process is applied. The geometry is resolved on a grid with lateral dimensions 1500 × 1500.

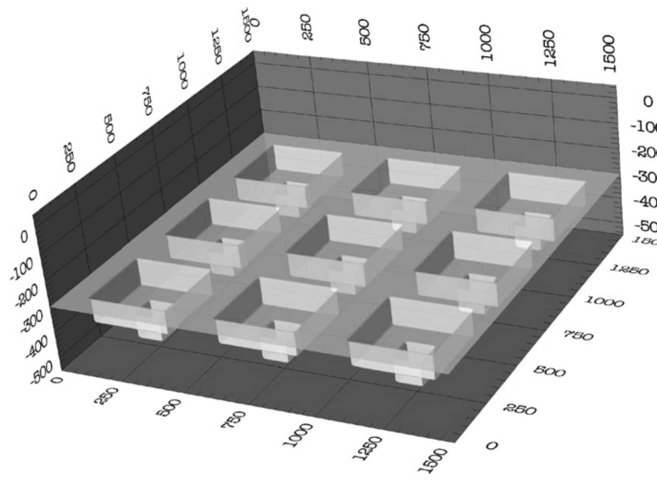
Fig. 10. The profile after exposure to an unidirectional etching process. A layer of 300 grid spacings was removed. The screening of flux is incorporated correctly.

to proof the linear scaling. Hereby we used reflective boundary conditions for the lateral grid directions and infinite boundaries for the vertical direction. In Table 1 the calculation times for one time step are given. Additionally the memory consumption for the HRLE data structure, together with its number of defined and active grid points are listed. Since we use a second order space differencing scheme for time integration, which requires 2 additional layers of defined grid points at the positive and the negative side, respectively, the number of defined grid points is roughly 5 times the number of defined grid points. For storing the defined level set values we use 8 byte floating point numbers.

It should be noted that the HRLE data structure allocates more memory than listed, since our implementation bases on the STL vector [14], which uses more memory to obtain a constant complexity for back insertions. Furthermore, twice the memory is needed during the rebuilding procedure, where the old and the new data structures have to be kept in parallel. Nevertheless, the savings in memory requirements are significant. For example, the memory requirements for storing the level set values for all points of a grid with dimensions $2000 \times 3000 \times 3000$ are 134 GB, whereas using HRLE only 1.25 GB are necessary.

5.2. *Unidirectional etching*

The visibility check is demonstrated on a perfect unidirectional etching process. The geometry is shown in Fig. 9 and resolved on a grid with lateral dimensions 1500 × 1500. The etching rate is set proportional to the incoming flux (13). Fig. 10 shows the final profile after etching away a layer of 300 grid spacings. The calculation time per time step varies from 33 s down to 24 s due to the decreasing surface size.

5.3. *Isotropic etching*

To test the multi-level set method for the description of different materials we simulate an isotropic etching process. The model

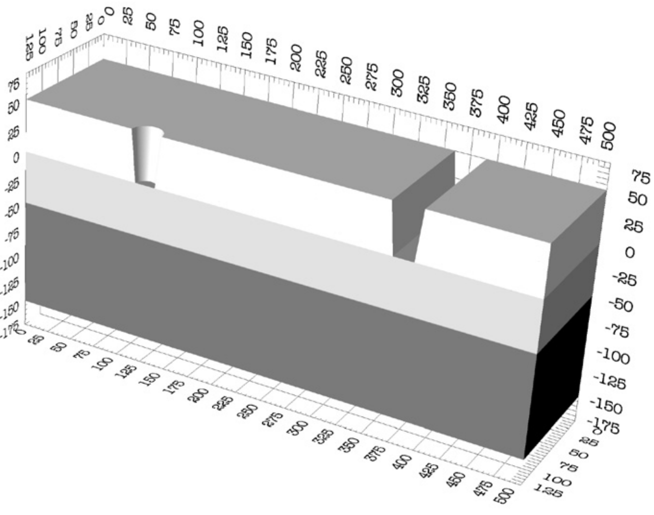
Fig. 11. The test structure with different material regions used for the isotropic etching simulation. Three layers are on top of the substrate (black), with thicknesses 0.5 (dark gray), 50 (light gray), and 50 grid spacings (white). Since the first layer is very thin, it cannot be seen clearly.

shown in Fig. 11 consists of a stack of 4 different material regions. 3 layers with thicknesses 0.5, 50, and 50 grid spacings (mask) are on top of the substrate. The structure is represented by 4 level set function discretized on a regular grid with lateral extensions 500 × 125. For the simulation we use material specific constant etch rates. Their values are, from top to bottom, 0.2/1/0.05/1 (grid spacings per time unit). The profile after 90 time units is shown in Fig. 12. The calculation time per time step is in the range of 0.9 and 1.2 s.

Our multi-level-set technique is able to describe the surface evolution accurately, despite large etch rate ratios and the existence of a layer with thickness of less than one grid spacing.

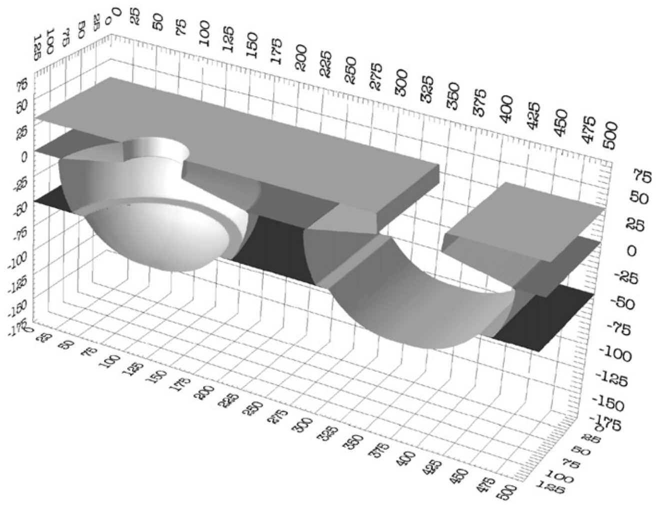

Fig. 12. The final profile after an isotropic etching process with material dependent etch rates. The geometry is represented by 4 level set functions. Due to the thin layer two level set functions (black) almost coincide.

6. Conclusion

We introduced the HRLE data structure to topography simulations, and showed how it can be applied to obtain very efficient algorithms, in terms of run time and memory consumption, in combination with the sparse field level set method. The data structure is also convenient to determine the visibilities with optimal complexity for unidirectional fluxes.

We also presented a multi-level-set method based on the HRLE data structure and the sparse field level set method, which is able to describe the surface evolution in the presence of different material regions. Hereby, very thin layers and large variations in surface velocities can be incorporated.

Since all presented methods have linear scaling laws in terms of the surface size, they are very convenient for topography simulations of large three-dimensional geometries.

References

[1] U.-H. Kwon, W.-J. Lee, Thin Solid Films 445 (2003) 80.
[2] R.E. Jewett, P.I. Hagouel, A.R. Neureuther, T. van Duzer, Polym. Eng. Sci. 17 (1977) 381.
[3] E. Strasser, S. Selberherr, IEEE T. Comput. Aid. D. 14 (1995) 1104.
[4] T. Smy, S.K. Dew, R.V. Joshi, J. Vac. Sci. Technol. A 19 (2001) 251.
[5] M. Fujinaga, N. Kotani, IEEE T. Electron. Dev. 44 (1997) 226.
[6] Z.F. Zhou, Q.A. Huang, W.H. Li, W. Lu, IEEE T. Comput. Aid. D. 26 (2007) 100.
[7] S. Osher, J.A. Sethian, J. Comput. Phys. 79 (1988) 12.
[8] D. Adalsteinsson, J.A. Sethian, J. Comput. Phys. 148 (1999) 2.
[9] D. Adalsteinsson, J.A. Sethian, J. Comput. Phys. 118 (1995) 269.
[10] R.T. Whitaker, Int. J. Comput. Vision 29 (1998) 203.
[11] B. Radjenović, S.J. Kim, J.K. Lee, in: Proc. 12th Int. Congress on Plasma Physics, Nice, France, 2004.
[12] J. Strain, J. Comput. Phys. 151 (1999) 616.
[13] B. Houston, M.B. Nielsen, C. Batty, O. Nilsson, K. Museth, ACM Trans. Graph. 25 (2006) 151.
[14] B. Stroustrup, The C++ Programming Language, 3rd edition, Addison-Wesley, 2000.
[15] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge Univ. Press, 1999.
[16] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, The Visual Computer 11 (1995) 429.
[17] W.E. Lorensen, H.E. Cline, SIGGRAPH Comput. Graph. 21 (1987) 163.
[18] Z.-K. Hsiau, E. Kan, J. McVittie, R. Dutton, IEEE T. Electron. Dev. 44 (1997) 1375.
[19] T.S. Cale, G.B. Raupp, J. Vac. Sci. Technol. B 8 (1990) 1242.
[20] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003.