Journal of Scientific Computing (2023) 94:71
https://doi.org/10.1007/s10915-023-02133-5

A Complementary Topographic Feature Detection Algorithm Based on Surface Curvature for Three-Dimensional Level-Set Functions

Christoph Lenz[1] · Luiz Felipe Aguinsky[1] · Andreas Hössinger[2] · Josef Weinbub[1]

Received: 25 July 2022 / Accepted: 12 January 2023

Abstract

The level-set method is widely used in expanding front simulations in numerous fields of computational research, such as computer graphics, physics, or microelectronics. In the latter, the level-set method is employed for topography simulations of semiconductor device fabrication processes, being driven by complicated physical and chemical models. These models tend to produce surfaces with critical points where accuracy is paramount. To efficiently increase the accuracy in regions neighboring these critical points, automatic hierarchical domain refinement is required, guided by robust feature detection. Feature detection has to be computationally efficient and sufficiently accurate to reliably detect the critical points. To that end, we present a fast parallel geometric feature detection algorithm for three-dimensional level-set functions. Our approach is based on two different, complementary curvature calculation methods of the zero level-set and an optimized feature detection parameter to detect features. For performance reasons, our algorithm can be in principal linked to different curvature calculation methods, however, as will be discussed, two particularly attractive options are available: (i) A novel extension of the standard curvature calculation method for level-set functions, and (ii) an often disregarded method for calculating the curvature due to its purported low numerical accuracy. We show, however, that the latter is still a viable option, and that our algorithm is able to reliably detect features on geometries stemming from complicated, practically relevant geometries. Our algorithm and findings are applicable to other fields of applications such as surface simplification.

✉ Christoph Lenz
lenz@iue.tuwien.ac.at

Luiz Felipe Aguinsky
aguinsky@iue.tuwien.ac.at

Andreas Hössinger
andreas.hoessinger@silvaco.com

Josef Weinbub
weinbub@iue.tuwien.ac.at

[1] Christian Doppler Laboratory for High Performance TCAD, Institute for Microelectronics, TU Wien, Gußhausstraße 27-29, 1040 Vienna, Austria

[2] Silvaco Europe Ltd., Compass Point, St Ives, Cambridge PE27 5JL, UK

Keywords Level-set method · Curvature · Feature detection · Parallel algorithm

1 Introduction

The level-set method was introduced by Osher and Sethian and has proven itself to be a valuable tool in advancing front simulations [1–3]. The front is described by the zero level-set of a level-set function ϕ in a domain $\Omega \subset \mathbb{R}^n$

$$S = \{\vec{x} \in \Omega : \phi(\vec{x}) = 0\}, \tag{1}$$

which is discretized on a grid with resolutions $(\Delta x_1 \dots \Delta x_n)$. If the grid resolution is equivalent in all coordinate directions then it is called a regular grid with resolution Δx. This implicit representation allows the level-set method to robustly handle the merging of fronts without self-intersections.

In this work we focus in particular on topography simulations for modeling semiconductor device fabrication processes, since the merging of fronts often occurs there [4]. However, the here presented algorithm and findings are applicable to other fields in a similar manner. In general, semiconductor topography simulations are a vital part of the set of process technology computer-aided design (TCAD) simulation toolset. Dedicated simulators are available, such as, commercial (e.g., Silvaco's Victory Process [5]) and open source tools (e.g., ViennaTS [6]). In semiconductor topography simulations, the surface of a wafer undergoing processing is usually described by the zero level-set of a level-set function.

During an entire process TCAD simulation workflow of a practically relevant semiconductor device, several different physical and chemical models are computed to represent, e.g., the formation of different materials (deposition) or the removal of material regions from the wafer (etching). In a multi-layer structure representation, these materials are represented by individual level-set functions and a stacking process using Boolean operations is utilized to keep the description of thin material layers consistent (layer wrapping) [7]. The effects of the different physical and chemical models on each material layer are simulated by propagating the zero level-sets of these material layers. The temporal evolution of the zero level-set of each layer is described by the level-set equation

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

where $V(\vec{x}, t)$ stands for a velocity field that drives the evolution of the zero level-set. The modeling of the physical and chemical processes is achieved through the construction of $V(\vec{x}, t)$ [8, 9]. Certain velocity fields are so intricate that the properties of the Hamiltonian $\mathcal{H} = V(\vec{x}, t)\|\nabla \phi(\vec{x}, t)\|$ in (2) change, e.g., $\mathcal{H}$ only has a weak solution [10]. Thus, different methods of numerically solving (2) using finite differences are required for certain simulation workflows [10–12]. The velocity fields are directly linked to the discrete representation of the zero level-sets of the affected material layers, which can impose strict quality requirements on the feature detection (e.g. noise in the zero level-set).

To calculate the finite differences required to solve the level-set Eq. (2) and to obtain the curvatures of the zero level-set, the level-set values (ϕ-values) are required on all grid points. This is achieved by a re-distancing step [13]. Depending on the used level-set framework, the level-set function may fulfill the *signed distance property*

$$\forall \vec{x} \in \Omega \ \|\nabla \phi(\vec{x})\| = 1, \tag{3}$$

after the re-distancing step. This is achieved by using, for example, the fast marching method (FMM) [14]. Level-set frameworks based on the sparse field approach drop the signed distance property for a faster re-distancing algorithm [15]. To further improve performance, so-called *narrow-band* methods are used that only calculate the ϕ-values for grid points that are required for the calculation of finite differences [16].

Topographies originating from process TCAD simulations are often characterized by large, essentially flat or only slightly bent areas, and small areas with pronounced geometric variation. We refer to these areas with pronounced geometric variation as features. These properties of the topography motivate the use of hierarchical domain refinement such that only areas around a feature are resolved in more detail, to minimize the impact on performance [17]. The selective refinement of the simulation domain is a commonly used strategy for handling practically relevant numerical simulations [18–23]. To that end and the focus of this work, a new algorithm is needed to detect the dynamic changes in the wafer surface (features) during topography simulation, enabling external fast and efficient hierarchical domain refinement.

Geometric feature detection or extraction of three-dimensional (3D) data sets is a widely studied field, where the surface curvature, which describes the local changes in the geometry, is used to achieve feature detection [24–27]. The curvature of surfaces is of high interest in other computational domains as well, such as fluid dynamics, where the relation between surface curvature and surface tension is of interest [28]. However, in these applications the numerical accuracy of the calculated curvature values is the most important factor, whereas for feature detection the computation time of the curvature calculation is of equal or greater concern. Recently, machine learning approaches have been proposed to approximate the curvature of the zero level-set of two dimensional level-set functions [29]. Nonetheless, these methods require the same number of grid points as those based on finite differences. Moreover, the machine learning model has to be re-trained for each value of Δx, which is a user-supplied parameter in most process TCAD simulations and thus of limited practical use.

In this work, we tackle the problem of efficient and automatic feature detection for hierarchical grid refinement. We present a parallel topographical feature detection algorithm based on the curvatures of the zero level-sets of 3D level-set functions. Our algorithm uses complementary methods of calculating the curvatures depending on the applied process model (i.e., the velocity field) to provide robust and accurate feature detection. We show that in particular two curvature calculation methods are of interest due to their complementary application goals: accuracy or performance. We discuss the run-time of the feature detection and evaluate our feature detection algorithm on several geometries originating from process TCAD simulations. Furthermore, we use these geometries to calibrate a numerical feature detection parameter tailored to process TCAD simulations.

2 Feature Detection Algorithm

Before we describe the feature detection algorithm, we discuss the curvatures of a 3D surface. Furthermore, to ensure the robustness of the feature detection algorithm we discuss the impact of minimal surfaces. These considerations allow us to define feature points on the zero level-set.

2.1 Curvatures of a 3D Surface

A point on a 3D orientable surface (e.g., zero level-set) has two principal curvatures κ_1 and κ_2 [30]. The principal curvatures define the *mean curvature*

$$H = \frac{\kappa_1 + \kappa_2}{2}, \tag{4}$$

and the *Gaussian curvature*

$$K = \kappa_1 \kappa_2. \tag{5}$$

The mean and Gaussian curvatures of the zero level-set can be calculated directly from the ϕ-values using various methods which are presented in Sect. 3. The principal curvatures can be calculated from the mean and Gaussian curvatures [30]

$$\kappa_1 = H + \sqrt{H^2 - K}, \tag{6}$$

$$\kappa_2 = H - \sqrt{H^2 - K}. \tag{7}$$

The maximal mean curvature a level-set function can describe is bound by $\pm 1/\Delta x$ [3], since the smallest circle radius which can be represented with a given resolution is Δx.

The mean curvature captures the local geometric variation of the zero level-set for most geometries that typically occur during level-set based simulations. Thus, we will focus our primary discussion on the mean curvature and only return to the Gaussian curvature to improve the robustness of the feature detection.

2.2 Minimal Surfaces

A minimal surface is one that has a mean curvature H of zero. However, κ_1 and κ_2 might not necessarily be equal to zero on each point on the surface (see Eq. 4). In this case, every point on a minimal surface is a saddle point. Examples of such a surface are the catenoid, the helicoid, or the plane [31]. It is possible that segments of the zero level-set locally describe a minimal surface that is not a plane (e.g., a local saddle point). Thus, it is essential to detect these segments of the zero level-set as features even though their mean curvature is zero. Simply considering the mean curvature of the zero level-set of a 3D level-set function is not sufficient to determine if a surface point is a feature or not. Only if a point on the zero level set fulfills $H = K = 0$ its principal curvatures also fulfill $\kappa_1 = \kappa_2 = 0$, thus, the point is part of a plane. Therefore, if a point on the zero level-set has a mean curvature of $H = 0$ and its Gaussian curvature is $K \neq 0$ it is part of a minimal surface and should be considered part of a feature.

2.3 Feature Definition

Before describing the feature detection algorithm, we first need to specify what parts of the zero level-set we consider to be a feature. When the zero level-set locally describes a plane, then $H = K = 0$. These regions have no geometric variation and are *not* features for our purposes. Consequently, a point on the zero level-set is considered to be a feature if it is not part of a plane, i.e., $H \neq 0$ or $K \neq 0$. However, this definition is too strict when considering the following: (i) The curvature values calculated for the zero level-set are numerical approximations, thus small numerical errors can cause the calculated values to deviate from an analytical plane. (ii) If the zero level-set only bends slightly, e.g., has a small numerical mean curvature value, it should not be considered a feature since the geometric variation is smooth enough to not negatively impact the propagation of the zero level-set when solving (2). We thus define a *feature* of a level-set function as a point on the zero-level set that has an absolute mean curvature value higher than a threshold parameter $C > 0$ and is not part of a plane.

This approach is similar to calculating the curvedness of a surface [32]. However, our approach is more computationally efficient since it avoids several additional calculation required for the curvedness. In our approach the principal curvatures do not need to be explicitly calculated, additionally our approach avoids calculating the Gaussian curvature if the mean curvature of a point is not 0.

2.4 Algorithm

The features of the zero level-set are detected by processing the grid points. For each grid point on the zero level-set the absolute mean curvature $|H|$ is calculated. If $|H|$ is bigger than the feature threshold parameter C, then the point on the zero level-set is considered to be a feature. Otherwise, if $|H| < C$ the point is checked if it is part of a minimal surface by calculating the Gaussian curvature K and comparing it against $|K| > C$. Therefore, the algorithm detects a feature when either of the following conditions is met:

$$\begin{cases} |H| > C, \text{ or} \\ |H| < C \text{ and } |K| > C. \end{cases} \tag{8}$$

The feature detection algorithm is parallelized in a straightforward manner since it only depends on the curvature calculation which only needs to access the ϕ-values (provided by the re-distancing step). In this work we use the domain decomposition approach presented in [33].

3 ϕ-Value Based Curvature Calculation Methods

In this section we discuss three established methods and introduce one novel method for calculating the curvatures of the zero level-set with finite differences. To achieve a better understanding of the performance of the different methods we discuss the required stencils to calculate the curvatures. Our novel method of calculating the curvatures of the zero level-set uses more grid points than the standard methods without increasing the size of the finite difference stencil. This is achieved by incorporating under-utilized information from points already present in the stencil. The primary computational effort of the feature detection is the movement of the finite difference stencil over the grid points inside the narrow-band. Thus, the size of the finite difference stencil is significant when considering computational performance. Furthermore, this analysis allows us to choose complementary curvature calculation methods dependent on the surface properties of the level-set functions in the feature detection algorithm (see Sect. 2.4).

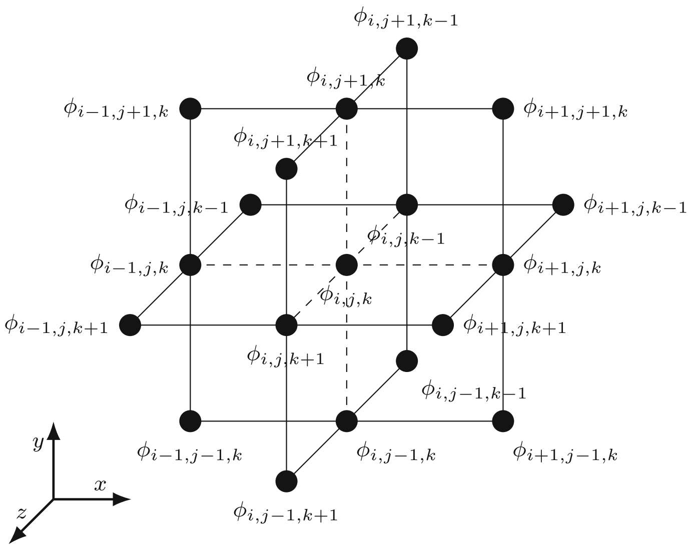


Fig. 1 Plane stencil used to calculate the mean curvatures of the zero level-set

3.1 General Formula

The most common method to calculate the mean curvature of the zero level-set is the general formula for implicit surfaces [34]:

$$H = \frac{\begin{array}{c} D_x(\phi)^2(D_{yy}(\phi) + D_{zz}(\phi)) \\ +D_y(\phi)^2(D_{xx}(\phi) + D_{zz}(\phi)) \\ +D_z(\phi)^2(D_{xx}(\phi) + D_{yy}(\phi)) \\ -2D_x(\phi)D_y(\phi)D_{xy}(\phi) \\ -2D_y(\phi)D_z(\phi)D_{yz}(\phi) \\ - 2D_z(\phi)D_x(\phi)D_{xz}(\phi) \end{array}}{2\|\nabla\phi\|^3}. \tag{9}$$

The derivatives required in (9) are calculated by using the following finite difference approximations

$$D_x(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}, \tag{10}$$

$$D_{xx}(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{\Delta x^2}, \tag{11}$$

$$D_{xy}(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j+1,k} - \phi_{i-1,j+1,k} - \phi_{i+1,j-1,k} + \phi_{i-1,j-1,k}}{4\Delta x \Delta y}, \tag{12}$$

where $\phi_{i,j,k}$ stands for the ϕ-value of the level-set function at the grid point (i, j, k). We will refer to this method of calculating the mean curvature as the *General Formula* method.

These finite differences require the level-set values of 19 different grid points to calculate the mean curvature of the zero level-set. These 19 points lie in the three intersecting planes that are created by the three coordinate direction (plane stencil) as shown in Fig. 1.

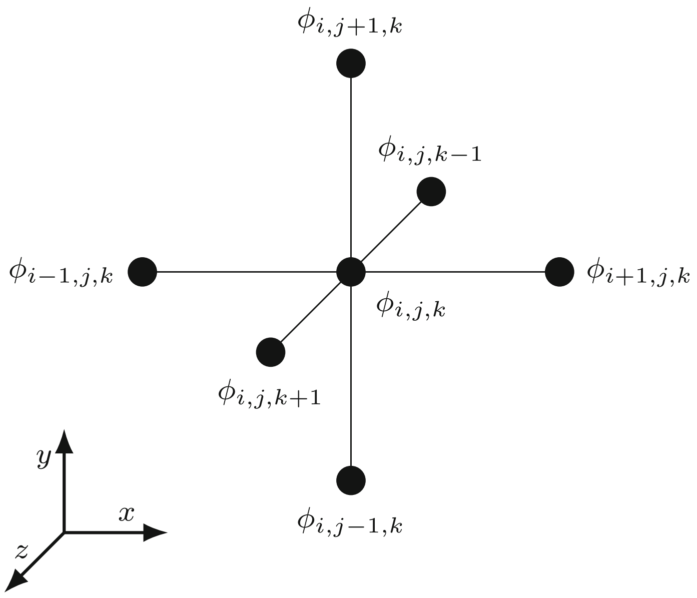


Fig. 2 Star stencil used to calculate curvatures of the zero level-set

The Gaussian curvature can be calculated in a similar way to the mean curvature [34]:

$$K = \frac{\begin{array}{l} D_x(\phi)^2(D_{yy}(\phi)D_{zz}(\phi) - D_{yz}(\phi)^2) \\ +D_y(\phi)^2(D_{xx}(\phi)D_{zz}(\phi) - D_{xz}(\phi)^2) \\ +D_z(\phi)^2(D_{xx}(\phi)D_{yy}(\phi) - D_{xy}(\phi)^2) \\ +2D_x(\phi)D_y(\phi)(D_{xz}(\phi)D_{yz}(\phi) - D_{xy}(\phi)D_{zz}(\phi)) \\ +2D_x(\phi)D_z(\phi)(D_{xy}(\phi)D_{yz}(\phi) - D_{xz}(\phi)D_{yy}(\phi)) \\ +2D_y(\phi)D_z(\phi)(D_{xy}(\phi)D_{xz}(\phi) - D_{yz}(\phi)D_{xx}(\phi)) \end{array}}{(D_x(\phi)^2 + D_y(\phi)^2 + D_z(\phi)^2)^2}. \tag{13}$$

Calculating the Gaussian curvature requires the same derivatives as calculating the mean curvature thus no changes in the stencil are required.

3.2 Shape Operator

When the level-set function ϕ fulfills the signed distance property (see Eq. 3) the mean curvature can be calculated as follows [3]

$$H = \frac{\mathrm{trace}(\mathbf{H}_{\phi_{i,j,k}})}{2} \approx \frac{D_{xx}(\phi_{i,j,k}) + D_{yy}(\phi_{i,j,k}) + D_{zz}(\phi_{i,j,k})}{2}, \tag{14}$$

where $\mathbf{H}_{\phi_{i,j,k}}$ stands for the Hessian of ϕ at the grid point (i, j, k). We will refer to this method as the *Shape Operator* method.

Calculating the mean curvature of the zero level-set using the *Shape Operator* method only requires the level-set values of 7 grid points, which lie on the grid lines that pass trough the grid point (i, j, k) (star stencil) as shown in Fig. 2.

The *Shape Operator* method is an often overlooked method of calculating the mean curvature of a level-set function: In performance-oriented applications, usually the signed distance property of the level-set function is approximated with a first-order scheme (e.g. using FMM). Which, introduces numerical errors into the calculated mean curvature values. However, in Sect. 4 we demonstrate that this error is negligible when this method is used for feature detection.

Alongside with requiring a smaller stencil, the *Shape Operator* method has an additional performance advantage: It is possible to avoid explicitly calculating the Gaussian curvature to determine if a point on the zero level-set is a feature. Both conditions of the feature detection algorithm (8) are simultaneously checked if:

$$|D_{xx}(\phi_{i,j,k})| + |D_{yy}(\phi_{i,j,k})| + |D_{zz}(\phi_{i,j,k})| > 2C. \tag{15}$$

Let the grid point (i, j, k) be part of a minimal surface that is not a plane, so $H = 0$ and without loss of generality:

$$-D_{yy}(\phi_{i,j,k}) = D_{xx}(\phi_{i,j,k}) + D_{zz}(\phi_{i,j,k}) \neq 0. \tag{16}$$

Thus (15) only holds if $D_{xx}(\phi_{i,j,k}) = D_{yy}(\phi_{i,j,k}) = D_{zz}(\phi_{i,j,k}) \approx 0$, which is fulfilled if the considered point on the zero level-set is a feature.

3.3 Variation of Normal

The third method calculates the mean curvature of the zero level-set by calculating the Euler-Lagrange derivative of the normal vector [35, 36]

$$H = \nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|} \approx \frac{1}{2\Delta x}\Bigg(\left(\frac{D_x^+(\phi)}{\|\vec{g}_x^+\|} - \frac{D_x^-(\phi)}{\|\vec{g}_x^-\|}\right) + \left(\frac{D_y^+(\phi)}{\|\vec{g}_y^+\|} - \frac{D_y^-(\phi)}{\|\vec{g}_y^-\|}\right) + \left(\frac{D_z^+(\phi)}{\|\vec{g}_z^+\|} - \frac{D_z^-(\phi)}{\|\vec{g}_z^-\|}\right)\Bigg), \tag{17}$$

where $\vec{g}_x^{\pm}$ is defined as follows:

$$\vec{g}_x^{\pm} := (D_x^{\pm}(\phi_{i,j,k}), \frac{1}{2}(D_y(\phi_{i\ j,k}) + D_y(\phi_{i\pm1,j,k})), \frac{1}{2}(D_z(\phi_{i\pm1,j,k}) + D_z(\phi_{i,j,k}))), \tag{18}$$

$$\vec{g}_y^{\pm} := (D_y^{\pm}(\phi_{i,j,k}), \frac{1}{2}(D_x(\phi_{i,j\pm1,k}) + D_x(\phi_{i,j,k})), \frac{1}{2}(D_z(\phi_{i,j\pm1,k}) + D_z(\phi_{i,j,k}))), \tag{19}$$

$$\vec{g}_z^{\pm} := (D_z^{\pm}(\phi_{i,j,k})), \frac{1}{2}(D_x(\phi_{i,j,k\pm1}) + D_x(\phi_{i,j,k})), \frac{1}{2}(D_y(\phi_{i,j,k\pm1}) + D_y(\phi_{i,j,k}))). \tag{20}$$

To calculate $\vec{g}_x^{\pm}$ the first-order finite forward and backwards differences have to be calculated by using the finite difference formulas:

$$D_x^+(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x}, \tag{21}$$

$$D_x^-(\phi_{i,j,k}) \approx \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\Delta x}. \tag{22}$$

This method again requires a 19-point plane stencil (see Fig. 1) to calculate the mean curvature of the zero level-set. We are going to refer to this method as the *Variation of Normal* method. To calculate the Gaussian curvature, the missing derivatives for (13) have to be calculated.

3.4 Big Stencil

Considering that a 19 point finite difference stencil is present to calculate the finite differences to determine the mean curvature of the zero level-set (see Sect. 3.1 and Fig. 1), we propose using an existing, more accurate finite difference approximation [37] to calculate the mean curvature. The number of grid points used to calculate the finite differences D_x and D_{xx} can be increased without increasing the stencil by using the finite difference formulas

$$\tilde{D}_x(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j+1,k} - \phi_{i-1,j+1,k} + \phi_{i+1,j-1,k} - \phi_{i-1,j-1,k}}{4\Delta x} \text{ and} \tag{23}$$

$$\tilde{D}_{xx}(\phi_{i,j,k}) \approx \frac{\begin{array}{r}\phi_{i+1,j+1,k} - 2\phi_{i,j+1,k} + \phi_{i-1,j+1,k} + \phi_{i+1,j,k} - 2\phi_{i,j,k} \\ + \phi_{i-1,j,k} + \phi_{i+1,j-1,k} - 2\phi_{i,j-1,k} + \phi_{i-1,j-1,k}\end{array}}{3\Delta x^2}. \tag{24}$$

When the mean curvature is calculated using (9) and the finite difference approximations (12), (23), and (24) this novel method is henceforth referred to as the *Big Stencil* method.

This new method uses the same finite difference stencil as the *General Formula* method (9) (see Fig. 1). However, it utilizes more information of the level-set function to calculate the derivatives required in (9). Thus, the numerical accuracy of the calculated mean curvature values is improved. The Gaussian curvature can simply be calculated by using the more accurate finite difference approximations and (13).

4 Results

We investigate the applicability of the curvature calculation methods presented in Sect. 3 and the presented feature detection algorithm in Sect. 2 for several representative geometries (Sect. 4.1). We run a parameter search for the parameter C to find a suitable parameter value for process TCAD simulations (Sect. 4.2) and perform an empirical evaluation of the calculated curvature values and detected features (Sect. 4.3). Furthermore, the performance of the curvature calculation methods is evaluated (Sect. 4.4). We implemented our feature detection algorithm in the open-source topography simulator *ViennaTS* [38] and used OpenMP for parallelization. The presented benchmarks have been compiled with GCC 9.3.0 and executed on a single node of VSC-4[1], which is equipped with two 24-core Intel Xeon Platinum 8174 processors.

4.1 Geometries and Mean Curvature Values

The first geometry we investigate is a sphere, where the mean curvature value can be analytically calculated and subsequently compared to numerical approximations. Most geometries

[1] https://vsc.ac.at//systems/vsc-4/

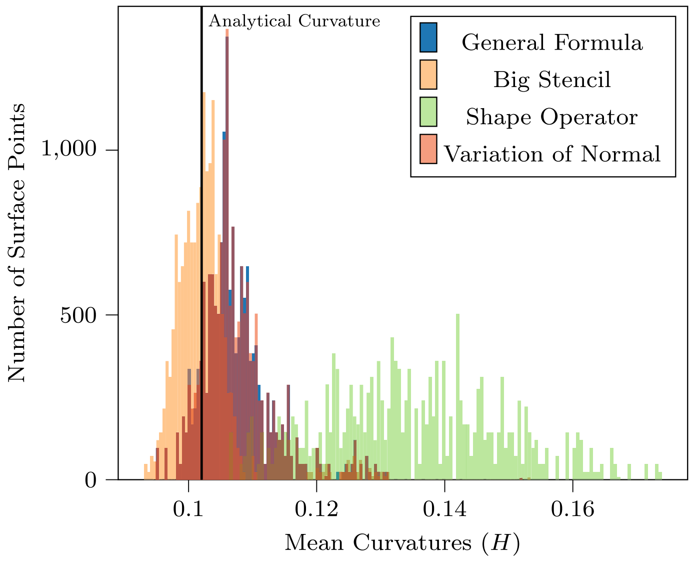


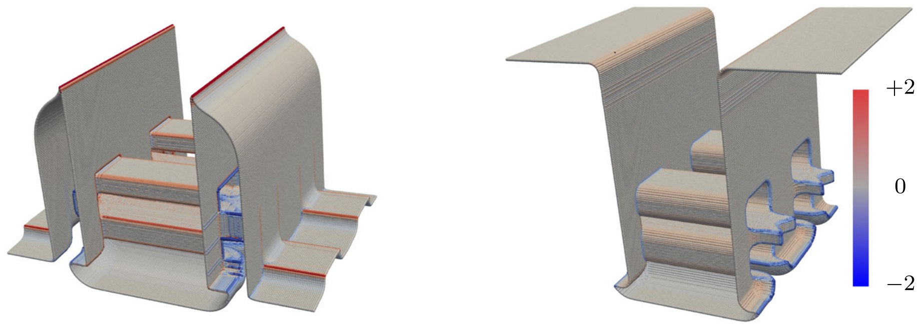
Fig. 3 Mean curvature values for a sphere with radius 9.7 and grid resolution 0.27 after 5 time units

that occur during level-set based simulations, Most geometries that occur during level-set based simulations, however, do not have a readily available analytical solution.

however, do not have a readily available analytical solution. Therefore, to evaluate the applicability of our methods for realistic scenarios, we additionally explore two representative and technologically relevant examples from the field of process TCAD: A stacked nanosheet field-effect transistor (FET) [39] and a selectively grown epitaxial crystal [10].

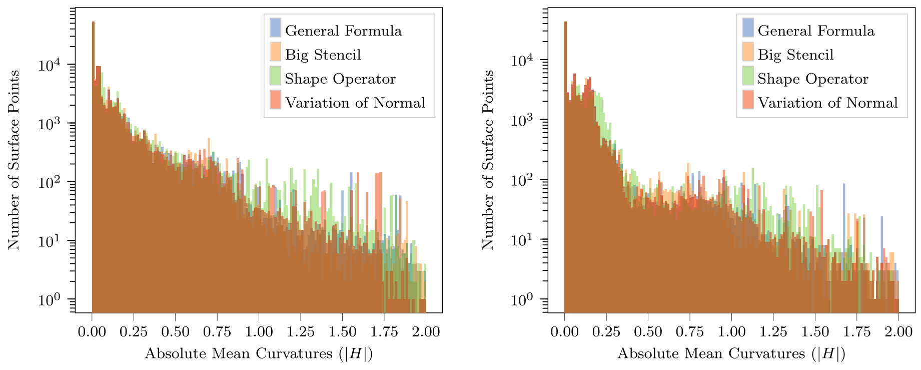
The analytical mean curvature of a sphere with radius r is known to be $1/r$. As previously discussed, every step in a level-set based simulation is a numerical approximation, often of first-order, which introduces small but compounding numerical errors. Thus, we subject the sphere to a velocity field to make the calculated mean curvature values comparable to values that occur during a simulation workflow. We initialize the sphere with a radius of 15. After initialization, the sphere is subjected to a constant velocity field with a velocity of -1 for 5 time units so that, analytically, the final sphere should have a radius of 10 and a mean curvature H of 0.1. A first-order Osher scheme is used to solve the level-set equation [11]. The distribution of the calculated mean curvature values of all points on the zero level-set, for all methods introduced in Sect. 3, is shown in Fig. 3. Due to numerical errors introduced by the used scheme, the distance from all points on the zero level-set to the center of the sphere is between 9.9 and 9.6, with an average distance of 9.7. The smaller radius compared to the analytical sphere is explained by a loss in volume originating from the normalization into the discrete grid. Thus, we expect the calculated mean curvature values to be distributed around the curvature of a sphere with radius 9.7, which is indicate by the black line in Fig. 3.

The mean curvature values computed with the *General Formula* and the *Variation of Normal* methods do not differ in a significant way from each other. The computed values using the *Shape Operator* show the largest deviation from the value of the analytical solution. This is expected from the reduced size of its stencil, as discussed in Sect. 3.2. The results obtained from the *Big Stencil* method provide the best match to the expected values from the analytical solution. All curvature calculation methods except for *Big Stencil* tend to



(a) Material layer 3 (b) Material layer 5

Fig. 4 Calculated mean curvature values of two material layers of a stacked nanosheet FET. Calculated with *Shape Operator*



(a) Material layer 3 (b) Material layer 5

Fig. 5 Distribution of the calculated absolute mean curvature values of the two material layers of a stacked nanosheet FET shown in Fig. 4

overestimate the mean curvature of the shrunken sphere. Calculations done with the *Shape Operator* method overestimate the mean curvature up to 75%. However, from our empirical analysis, we postulate that the *Shape Operator* method is still accurate enough to be viable for the purpose of feature detection, as will be further discussed in Sect. 4.3.

The next geometries under investigation originate from an exemplary process TCAD simulation of a stacked nanosheet FET [39]. A first-order Osher scheme is used to solve the level-set equation. As previously discussed in Sect. 1, different level-set functions representing the individual material layers are combined in a layer wrapping to model the topography of a semiconductor device. Therefore, it is sufficient to only investigate certain material layers, since the features of the underlying layers are already captured in the upper material layer. Figure 4 shows two representative material layers (layer 3 and layer 5) after the 24th process step of a stacked nanosheet FET fabrication simulation (see Table II of [39]). To arrive at the geometries shown in Fig. 4 each material layer has undergone significant geometric changes due to the employed velocity fields. These changes generate complicated geometries with pronounced features. The distribution of the absolute mean curvature values in these layer functions are shown in Fig. 5.

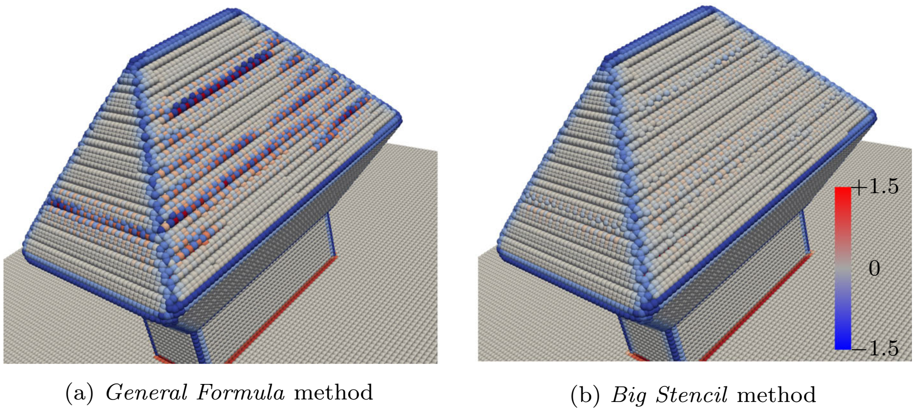


Fig. 6 Calculated mean curvature values of the zero level-set of the material layer of a heteroepitaxially grown SiGe crystal on a Si fin. The noise on the crystal facets introduced by the crystallographic orientation-dependent velocity field is reduced when using the *Big Stencil* method

The calculated mean curvature values for the two material layers of the stacked nanosheet FET shown in Fig. 5 display a similar behavior as on the sphere in Fig. 3. Namely, the *Shape Operator* method overestimates the mean curvature values compared to the other calculation methods which behave in a similar way. A significant amount (about 96%) of the absolute mean curvature values fall between 0.0 and 0.5, which indicates that most of the geometry does not have sharp features and is primarily flat or slightly bent. The absolute mean curvature values in the indicated band between the values of 0.0 and 0.5 agree to a large extent, except for some runaway values when using the *Shape Operator* method. However, as will be discussed later (see Sect. 4.3), the *Shape Operator* method is still able to distinguish between the features and flat parts of the geometries.

Crystallographic orientation-dependent growth requires velocity fields that depend upon special interpolation schemes to calculate the velocities for all points on the zero level-set [40]. Simulating such processes results in level-set functions with sharp corners which have to be maintained throughout a simulation to uphold accuracy. Furthermore, the simulation of such growth processes result in non-convex Hamiltonians that require a special numerical dissipation scheme to solve (2) [10]. Figure 6 shows the SiGe material layer of a heteroepitaxially grown SiGe crystal on a Si fin. The distribution of the calculated absolute mean curvature values for the SiGe material layer are shown in Fig. 7.

Figure 6 shows that the calculated mean curvature values of the facets, i.e., the planes formed by the crystallographic growth, are higher than expected when using the *General Formula* (see Fig. 6a). This is due to noise that is introduced by the numerical technique to solve this physical process. However, when we consider the mean curvature values calculated by the *Big Stencil* method (see Fig. 6b), the effects of the noise on the mean curvature values is reduced significantly. The lesser impact of noise and the thus created difference in calculated mean curvature values are explained by the additional grid points that are taken into consideration when using (23) to approximate the second-order derivatives. Since the level-set function is an approximation to a signed distance function, nearly all information about the curvature is gained from the second-order derivatives in the same coordinate direction (e.g., $\tilde{D}_{xx}, \dots$), see (14). Figure 7 confirms this observation as there are more mean curvature values close to zero and fewer mean curvature values larger than 2 for the SiGe material layer.

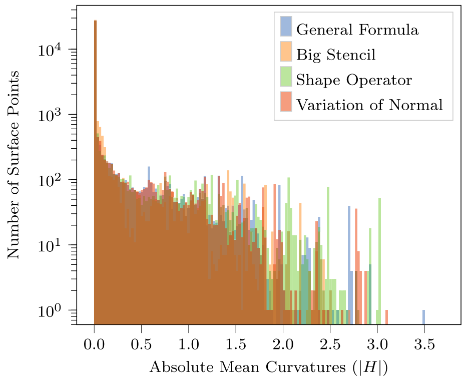


Fig. 7 Distribution of the calculated absolute mean curvature values for the SiGe material layer shown in Fig. 6

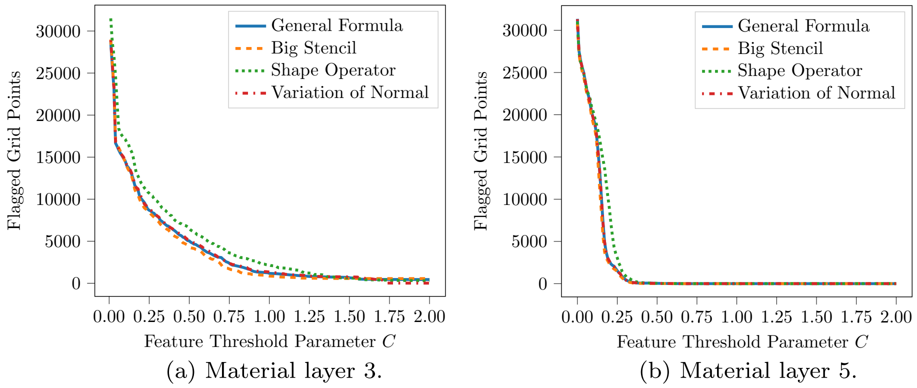


Fig. 8 Flagged points of the zero level-set compared to the chosen feature detection parameter C for the stacked nanosheet FET shown in Fig. 4

4.2 Parameter Study

To determine an appropriate choice for the feature threshold parameter C (8) (which will be used in Sect. 4.3), we conduct a parameter search which starts with a value of 0.01 up to $1/\Delta x$ with a step length of 0.01. Figure 8 shows the number of flagged grid points compared to the chosen feature threshold parameter C for the two representative material layers of the stacked nanosheet FET. Figure 9 shows the results of the feature threshold parameter search for the heteroepitaxially grown SiGe crystal material layer.

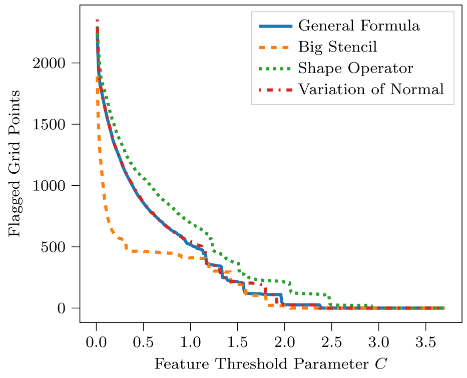


Fig. 9 Flagged points of the zero level-set compared to the chosen flagging parameter C for the heteroepitaxially grown SiGe crystal shown in Fig. 6

As expected, the *Shape Operator* method identifies slightly more surface points as features than the other three methods. Nevertheless, as will be discussed in Sect. 4.3, it still reliably detects features. The other three methods identify approximately the same number of surface points as features with only minor deviations. Figure 8 suggests that the feature detection parameter C should be between 0.1 and 1.0 since with smaller values the entire surface would be flagged as a feature and, conversely, with a larger parameter some features may be missed. Furthermore, Fig. 5 shows that a feature detection parameter of $C = 0.5$ identifies most features and simultaneously avoids capturing only slightly bent parts of the zero level-set, as discussed in Sect. 2.

The parameter search for the feature threshold parameter on the SiGe crystal shows a different behavior (see Fig. 9) than for the stacked nanosheet FET. Using the *Big Stencil* method for the curvature calculation initially identifies fewer points on the zero level-set as features than other methods. As previously discussed (Sect. 4.1), the difference is explained by the error in the mean curvature calculation introduced by the noise of the velocity field. The amount of grid points flagged as features equalizes at a feature threshold parameter of $C = 1.1$ for all methods. However (see Fig. 11), feature detection using this feature threshold parameter is not satisfactory: The noise in the level-set function is so large that the feature detection is not able to detect the edges of the crystal as features. Yet, the numerical noise on the crystal facet is still detected. Reconsidering the results shown in Fig. 9 with these newly gained insights, we infer that the *Big Stencil* method provides a substantially higher quality in the feature detection. The amount of flagged grid points is constant between the feature threshold parameters 0.4 and 1.1 when using the *Big Stencil* method. We thus pick 0.5 (same as in the previous case, the stacked nanosheet FET) as the feature threshold parameter C for the following empirical evaluation.

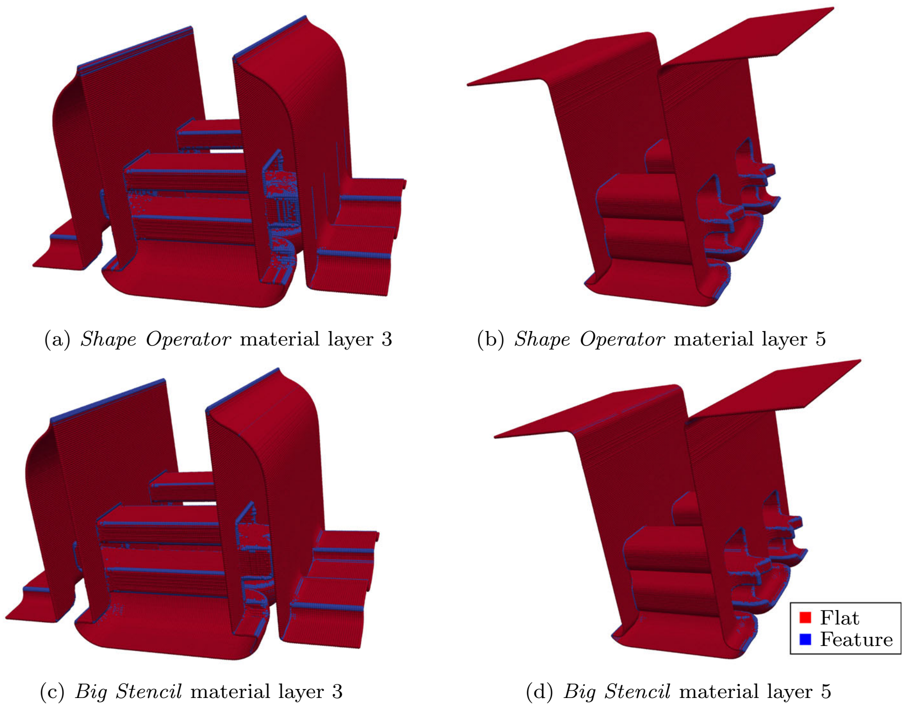

(a) *Shape Operator* material layer 3
(b) *Shape Operator* material layer 5
(c) *Big Stencil* material layer 3
(d) *Big Stencil* material layer 5
🟥 Flat
🟦 Feature

Fig. 10 Detected features of the stacked nanosheet FET using a feature detection parameter C of 0.5

4.3 Empirical Evaluation

This section evaluates if our proposed feature detection algorithm combined with the feature detection parameters C identified in Sect. 4.2 reliably detects the features of the investigated geometries. Figure 10 shows the detected features for the stacked nanosheet FET with a feature detection parameter of 0.5 using the *Shape Operator* and the *Big Stencil* methods, the features detected by the other methods are similar to these two methods. Figures 11 and 12 show the detected features of the SiGe crystal with a feature detection parameter of 1.1 and 0.5, respectively. We only show the results of the feature detection for the *General Formula* and *Big Stencil* method in Fig. 12 as the results of the *Shape Operator*, and *Variation of Normal* are similar to the results of the *General Formula*.

Figure 10 shows that a feature detection parameter of 0.5 is able to distinguish features form flat parts of the geometry, thus, confirming that it is a good choice for reliably detecting the feature of the stacked nanosheet FET.

The features detected in Fig. 11a–c show that the feature detection parameter of 1.1 is too large: This particular parameter choice does not allow for identifying the features at the edges of the SiGe crystal, while still falsely detecting points on the facets as features. The *Big Stencil* method does not detect any features on crystal facets but, due to the high feature detection parameter, it also fails to capture several grid points on the edges that should be detected as features, see Fig. 11d.

Figure 12 shows that the *Big Stencil* method is able to reliably detect the features of the SiGe crystal with a feature detection parameter of 0.5, while the other methods wrongly detect the

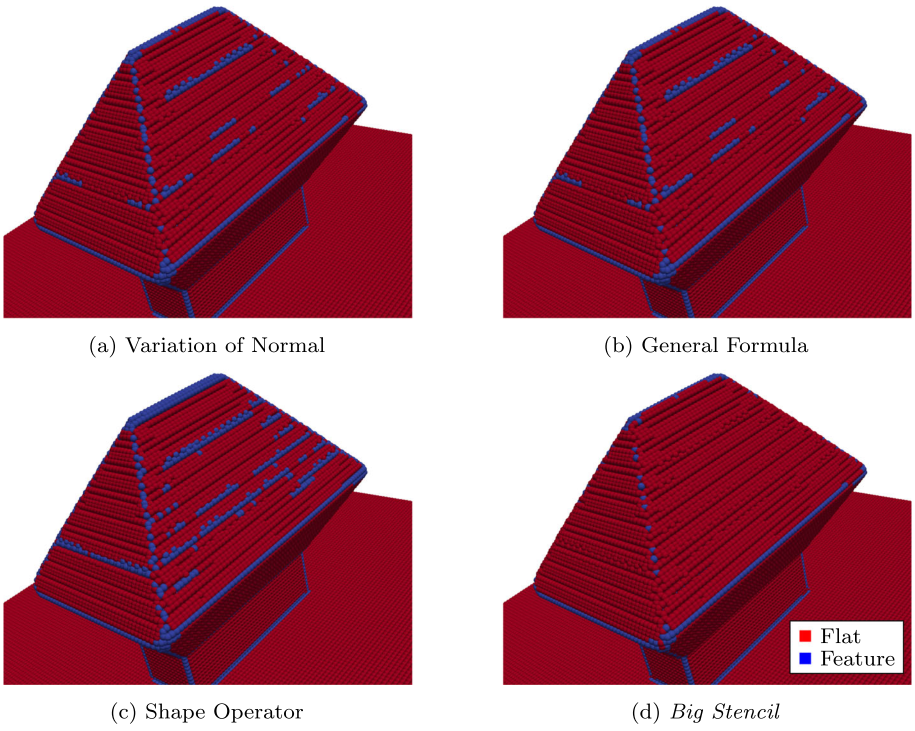


Fig. 11 Detected features of a SiGe crystal with a feature detection parameter C of 1.1

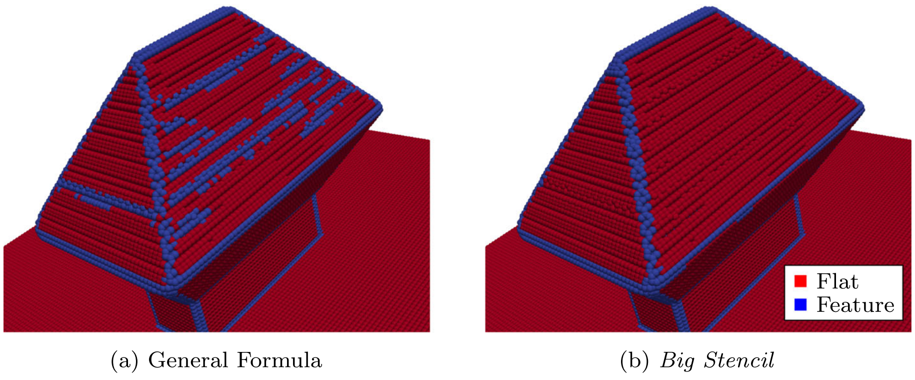


Fig. 12 Detected features of a SiGe crystal with a feature parameter C of 0.5

noise on the crystal facet as features. These observations suggest that it can be advantageous to use complementary curvature calculation methods dependent on the requirements of the application: The *Shape Operator* method reliably detects the features of the geometries relying on simple velocity fields as used, for example, for the fabrication simulation of the stacked nanosheet FET. In applications where the accuracy of the mean curvature is of utmost importance, the *Big Stencil* method should be used in virtue of its higher accuracy. This is the case, e.g., for non-convex Hamiltonians originating from crystallographic orientation-

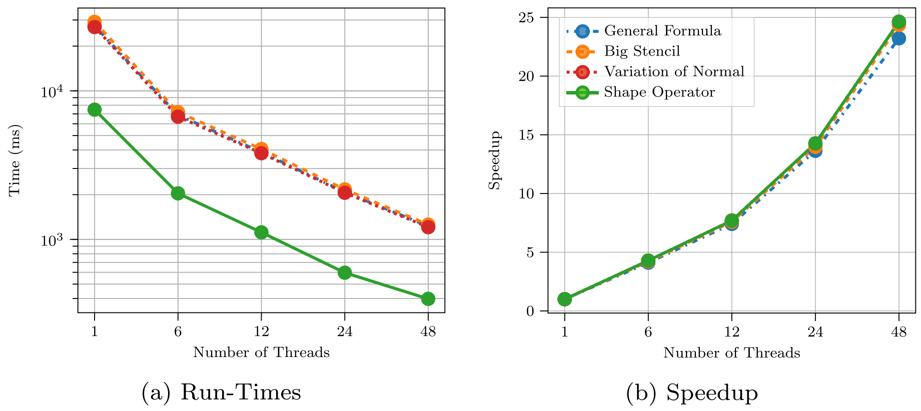


Fig. 13 Run-time and speedup of the feature detection algorithm on a sphere with radius 9.7 and grid resolution 0.27

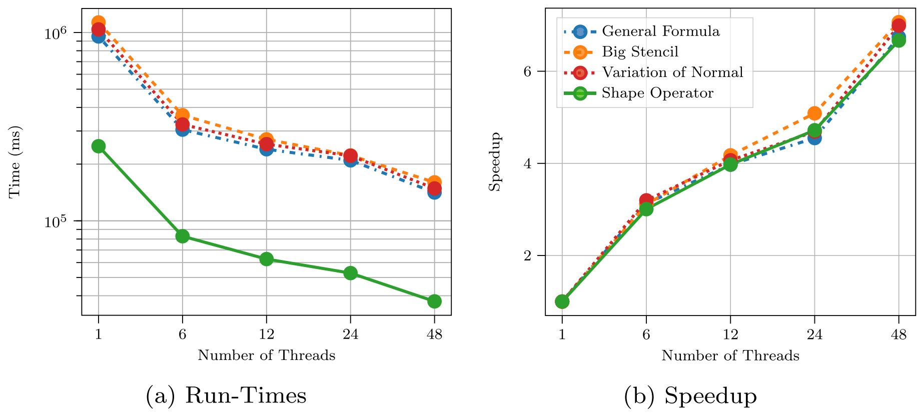


Fig. 14 Run-time and speedup of the feature detection algorithm for all material layers of the stacked nanosheet FET after 24 process steps

dependent velocity fields, where the *Big Stencil* method reduces the effects of noise on the curvature calculation.

4.4 Performance

This section evaluates the averaged performance (run-time and parallel speedup) of our feature detection algorithm based on the geometries introduced in Sect. 4.1. For this analysis all curvature calculation methods presented in Sect. 3 are considered.

As can be seen in Figs. 13, 14, and 15, the *Shape Operator* method is the fastest method while the other three methods perform similarly to each other. The speedup in Fig. 13a is a reflection of the best case scenario, since the convexity of the geometry enables the domain decomposition algorithm to create sub-domains with approximately the same number of surface points. This optimal parallelization scenario is of course not the case for practically relevant cases.

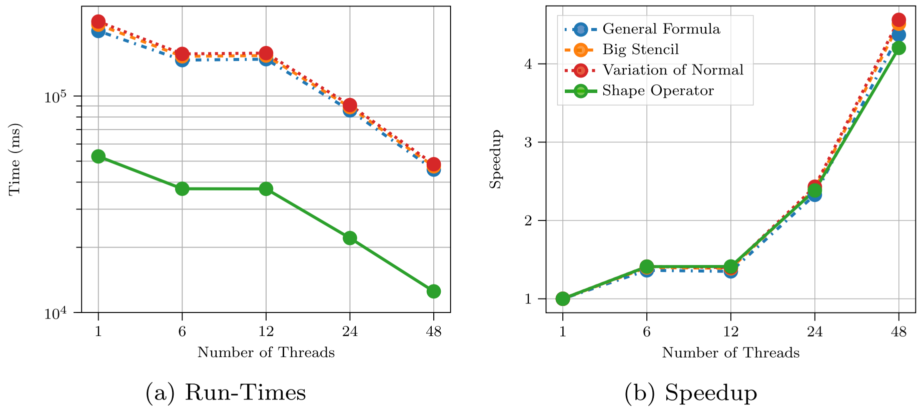


Fig. 15 Run-time and speedup of the feature detection algorithm for the SiGe crystal

The principal differences in run-time and speedup performance between Figs. 14 and 15 as compared to Fig. 13 are due to load-balancing issues in the domain decomposition: Even though the domain decomposition splits the narrow-band into roughly equally sized sub-domains (considering the entire grid), the number of surface points differs in each sub-domain.

In addition, the primary computational effort of the mean curvature calculations of the zero level-set is the application of the finite difference stencil. This fundamental process is inherently memory-bound and thus detrimental for parallel speedup.

The performance analysis reinforces the observations made in Sect. 4.3: Due to its higher accuracy and insignificant increase in run-time compared to the other methods the *Big Stencil* method should be the default choice only for complicated velocity fields where accuracy is paramount. However, for applications where the accuracy is a less significant factor and the primary concern is performance, the *Shape Operator* method is the optimal choice.

5 Conclusion

We introduce a parallel automatic feature detection algorithm for 3D level-set functions with a focus on computational performance, efficiency, and accuracy. The algorithm is based on two different but complementary methods for calculating the curvatures of the zero level-set. Our algorithm can utilize the optimal curvature calculation method based on the properties of the velocity field that is used to propagate the zero level-set. The *Shape Operator* method uses the signed distance property of the level-set function to calculate the mean curvature. This allows the *Shape Operator* method to use the smallest possible finite difference stencil to calculate the mean curvature of the zero level-set. It is, therefore, the fastest method to detect geometric features and is used for structures with little to no noise in the level-set layer functions such as found in the discussed simulation of an exemplary stacked nanosheet FET. The *Big Stencil* method can better cope with the noise introduced by crystallographic orientation-dependent velocity fields to allow for the exact detection of crystal facets. Thus, the *Big Stencil* method should be favored for complicated process models involving, e.g., strong dependencies on the surface normal. Furthermore, we determined that a good choice for the feature detection parameter C is 0.5 in process TCAD simulations for both complementary surface curvature methods, considering the discussed, representative examples.

We compare the methods in our algorithm with two other methods of calculating the mean curvature of the zero level-set in our investigations: The *General Formula* method, and the *Variation of Normal* method. In our experiments, these two methods perform comparably to each other regarding accuracy and performance. However, the *Shape Operator* method significantly outperforms both methods in terms of run-time while being sufficiently accurate for feature detection. Simultaneously, the *Big Stencil* method has an equivalent run-time to both methods yet it has an overall higher numerical accuracy and is less susceptible to models which introduce numerical noise.

Acknowledgements The financial support by the *Austrian Federal Ministry for Digital and Economic Affairs*, the *National Foundation for Research, Technology and Development*, and the *Christian Doppler Research Association* is gratefully acknowledged. The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

Author Contributions All authors contributed to the study conception and design. The first draft of the manuscript was written by CL and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by TU Wien (TUW).

Data Availibility Statement The data sets generated and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Osher, S.J., Sethian, J.A.: Fronts propagating with curvature dependent speed. J. Comput. Phys. **79**(1), 12 (1988). https://doi.org/10.1016/0021-9991(88)90002-2
2. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press, Cambridge (1999)
3. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces, vol. 153. Springer-Verlag, New York (2003)
4. Klemenschits, X., Selberherr, S., Filipovic, L.: Modeling of gate stack patterning for advanced technology nodes: a review. Micromachines **9**(12), 631 (2018). https://doi.org/10.3390/mi9120631
5. Silvaco Victory Process (2022). www.silvaco.com/tcad/victory-process-3d/
6. ViennaTS (2022). https://www.iue.tuwien.ac.at/software/viennats/
7. Ertl, O., Selberherr, S.: A fast level set framework for large three-dimensional topography simulations. Comput. Phys. Commun. **180**(8), 1242 (2009). https://doi.org/10.1016/j.cpc.2009.02.002
8. Aguinsky, L.F., Rodrigues, F., Wachter, G., Trupke, M., Schmid, U., Hössinger, A., Weinbub, J.: Phenomenological modeling of low-bias sulfur hexafluoride plasma etching of silicon. Solid-State Electron. **191**, 108262 (2022). https://doi.org/10.1016/j.sse.2022.108262

9. Rodrigues, F., Aguinsky, L.F., Toifl, A., Scharinger, A., Hössinger, A., Weinbub, J.: In Proceedings of the international conference on simulation of semiconductor processes and devices (SISPAD) (2021), pp. 229–232
10. Toifl, A., Simonka, V., Hössinger, A., Selberherr, S., Grasser, T., Weinbub, J.: Simulation of the effects of postimplantation annealing on silicon carbide DMOSFET characteristics. IEEE Trans. Electron Devices **66**(7), 3060 (2019). https://doi.org/10.1109/TED.2019.2916929
11. Engquist, B., Osher, S.: Stable and entropy satisfying approximations for transonic flow calculations. Math. Comput. **34**(149), 45 (1980). https://doi.org/10.2307/2006220
12. Osher, S., Shu, C.W.: High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. SIAM J. Numer. Anal. **28**(4), 907 (1991). https://doi.org/10.1137/0728049
13. Quell, M., Diamantopoulos, G., Hössinger, A., Weinbub, J.: Shared-memory block-based fast marching method for hierarchical meshes. J. Comput. Appl. Math. **392**, 113488 (2021). https://doi.org/10.1016/j.cam.2021.113488
14. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. **93**(4), 1591 (1996). https://doi.org/10.1073/PNAS.93.4.1591
15. Whitaker, R.T.: A level-set approach to 3D reconstruction from range data. Int. J. Comput. Vision **29**, 203 (1998). https://doi.org/10.1023/A:1008036829907
16. Chopp, D.L.: Computing minimal surfaces via level set curvature flow. J. Comput. Phys. **106**(1), 77 (1993). https://doi.org/10.1006/JCPH.1993.1092
17. Lenz, C., Toifl, A., Quell, M., Rodrigues, F., Hössinger, A., Weinbub, J.: Curvature based feature detection for hierarchical grid refinement in TCAD topography simulations. Solid-State Electron. **191**, 108258 (2022). https://doi.org/10.1016/j.sse.2022.108258
18. Zönnchen, B., Köster, G.: A parallel generator for sparse unstructured meshes to solve the Eikonal equation. J. Comput. Sci. **32**, 141 (2019). https://doi.org/10.1016/j.jocs.2018.09.009
19. Ervik, Å., Lervåg, K.Y., Munkejord, S.T.: A robust method for calculating interface curvature and normal vectors using an extracted local level set. J. Comput. Phys. **257**, 259 (2014). https://doi.org/10.1016/j.jcp.2013.09.053
20. Trompert, R.A., Verwer, J.G.: A Static-Regridding method for two-dimensional parabolic partial differential equations. Appl. Numer. Math. **8**(1), 65 (1991). https://doi.org/10.1016/0168-9274(91)90098-K
21. Lu, P., Xu, X.: A robust multilevel preconditioner based on a domain decomposition method for the Helmholtz equation. J. Sci. Comput. **81**, 291 (2019). https://doi.org/10.1007/s10915-019-01015-z
22. Yang, Z., Ming, J., Qiu, C., Li, M., He, X.: A multigrid multilevel Monte Carlo method for Stokes-Darcy model with random hydraulic conductivity and Beavers-Joseph condition. J. Sci. Comput. (2022). https://doi.org/10.1007/s10915-021-01742-2
23. Wang, C., Wang, W., Pan, S., Zhao, F.: A local curvature based adaptive particle level set method. J. Sci. Comput. (2022). https://doi.org/10.1007/s10915-022-01772-4
24. Ho, H.T., Gibbins, D.: In proceedings of the international conference on digital image computing: techniques and applications (DICTA) (2008), pp. 16–23. https://doi.org/10.1109/DICTA.2008.64
25. Mérigot, Q., Ovsjanikov, M., Guibas, L.J.: Voronoi-based curvature and feature estimation from point clouds. IEEE Trans. Visual Comput. Graphics **17**, 743 (2011). https://doi.org/10.1109/TVCG.2010.261
26. Kim, H.S., Choi, H.K., Lee, K.H.: Feature detection of triangular meshes based on tensor voting theory. CAD Comput. Aid. Des. **41**, 47 (2009). https://doi.org/10.1016/j.cad.2008.12.003
27. Clarenz, U., Rumpf, M., Telea, A.: Robust feature detection and local classification for surfaces based on moment analysis. IEEE Trans. Visual Comput. Graph. **10**, 516 (2004). https://doi.org/10.1109/TVCG.2004.34
28. Popinet, S.: Numerical models of surface tension. Annu. Rev. Fluid Mech. **50**, 49 (2018). https://doi.org/10.1146/annurev-fluid-122316-045034
29. Patel, H., Panda, A., Kuipers, J., Peters, E.: Computing interface curvature from volume fractions: a machine learning approach. Comput. Fluids **193**, 104263 (2019). https://doi.org/10.1016/j.compfluid.2019.104263
30. Stoker, J.J.: Differential Geometry. Pure and Applied Mathematics, v. 20 (Wiley-Interscience, New York, 1989)
31. Dierkes, U., Hildebrandt, S., Küster, A., Wohlrab, O.: Minimal Surfaces I: Boundary Value Problems (Springer. Berlin Heidelberg (1992)
32. Dorai, C., Jain, A.: COSMOS-a representation scheme for 3D free-form objects. IEEE Trans. Pattern Anal. Mach. Intell. **19**(10), 1115 (1997). https://doi.org/10.1109/34.625113
33. Filipović, L., Ertl, O., Selberherr, S.: in Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN) (2011), pp. 131–138. https://doi.org/10.2316/P.2011.719-045

34. Goldman, R.: Curvature formulas for implicit curves and surfaces. Comput. Aid. Geometr. Des. **22**(7), 632 (2005). https://doi.org/10.1016/j.cagd.2005.06.005
35. Whitaker, R.T., Xue, X.: In Proceedings 2001 international conference on image processing, vol. 3 (2001), pp. 142–145
36. Lefohn, A., Whitaker, R.T.: A GPU-Based, Three-Dimensional Level Set Solver with Curvature Flow. Tech. rep., UC Davis: Institute for Data Analysis and Visualization (2002)
37. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables (Dover Publications, 1974)
38. Klemenschits, L.F. X., Selberherr, S.: in Proceedings of the international conference on simulation of semiconductor processes and devices (SISPAD) (2020), pp. 59–62
39. Klemenschits, X., Manstetten, P., Filipovic, L., Selberherr, S.: in Proceedings of the international conference on simulation of semiconductor processes and devices (SISPAD) (2019), pp. 339–342
40. Ted J.: Hubbard, MEMS Design: The Geometry of Silicon Micromachining. Ph.D. thesis, California Institute of Technology (1994)