DISSERTATION

Hierarchical Grid Algorithms for Topography Simulation

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Betreuung von
Associate Prof. Dipl.-Ing. Dr.techn. Josef Weinbub
O.Univ.Prof. Dipl.-Ing. Dr.techn. Dr.h.c. Siegfried Selberherr

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik
von

Dipl.-Ing. Christoph Lenz, BSc

Matrikelnummer 0828641

Wien, im Juni 2023

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek
Your knowledge hub

Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Abstract

The continuously high pace of semiconductor device developments puts massive pressure on novel device designs and on manufacturing processes. Technology computer-aided design tools are critically important to aid these processes and, thereby, help reduce costly conventional experimental efforts. One key tool in the process technology computer-aided design tool-chain is topography simulation. Topography simulations allow to describe a plethora of time-dependent processing steps which change the wafer surface topography (e.g., etching or deposition).

The geometry, i.e., the topography, of modern semiconductor devices has become ever more critical in recent years due to the use of new materials and vertical structures with high aspect ratio. However, semiconductor device topographies are characterized by large areas with little to no geometric variation (e.g., flat parts of the topography) and small areas with pronounced geometric variation (e.g., sharp corners) – so-called features – for example, trench or hole structures which are very common in modern memory devices. This, in principle, requires an increase of the resolution of specific areas of interest of the simulation domain to accurately capture these features, which is efficiently achieved using a hierarchical grid. However, it has to be considered that the higher the resolution of the simulation domain is chosen, the bigger the detrimental impact on simulation performance, potentially leading to unpractical simulation run-times. Therefore, to improve simulation performance, strategies that automatically detect topographical features and locally adapt the configuration of the hierarchical grid accordingly are required.

A well-known metric that measures the geometric variation of a surface is the surface curvature, which is often used for automatic feature detection. This, however, first requires the availability of surfaces, and for that different surface representations are available, which can be broadly categorized into explicit and implicit representations. Depending on the type of simulation, certain surface representations have advantages over others. Implicit surfaces intrinsically handle the merging of materials during a simulation, while explicit surfaces improve the performance of flux calculations.

In this work, new geometry-aware algorithms for surfaces originating from topography simulations are introduced. The algorithms automatically detect features of the device topography. The detected features are then used to improve the simulation performance of topography simulations by locally adapting the resolution of the hierarchical grid.

A general feature detection algorithm for topography simulations was developed and is presented here. Four different methods for calculating the surface curvatures of level-set functions (i.e., implicit surface representation) are investigated. Three of the investigated methods are taken from literature, and one is a novel improvement of the de facto standard method. The novel method has a higher numerical accuracy than the other methods, while only insignificantly increasing the computational effort, making it the optimal choice for feature detection for semiconductor topography simulations.

The feature detection algorithm is used to guide a hierarchical grid placement algorithm, which locally increases the resolution of the simulation domain around features of the topography.

This hierarchical grid placement algorithm is integrated into a topography simulation workflow and used to simulate selective epitaxial growth of SiGe for evaluation purposes. The simulation performance is improved by up to 58% while maintaining accuracy.

Another developed algorithm improves the simulation performance of thin material layer etching, which is a common fabrication technique. The etching process can be simulated with Boolean operations between implicit surfaces. Depending on the thickness of the etched material layer, numerical artifacts develop. A specialized feature detection algorithm is presented that analyzes the thickness of the material layers and calculates a minimal required resolution to prevent the formation of numerical artifacts.

The developed feature detection algorithm can also be used to selectively simplify the simulated topography of a wafer. Therefore, a surface mesh (i.e., explicit surface) simplification algorithm is presented, which considers the features of the surface. This simplification algorithm maintains a high resolution at features of the topography, while simplifying non-features to a greater degree. The simplified surface meshes are then shown to improve the computational performance of flux calculations simulated with Monte Carlo ray tracing by 15%.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Das andauerende, hohe Tempo von Halbleiter-Bauelemente-Entwicklungen erzeugt massiven Druck auf neuartiger Bauelementedesigns und auf Herstellungsprozesse. Technologische, rechnerunterstützte Entwurfswerkzeuge sind von entscheidender Bedeutung, um diese Prozesse zu unterstützen und dadurch kostspielige konventionelle Expermimente zu reduzieren. Ein wichtiges Werkzeug in der Werkzeugkette des technologischen, rechnerunterstützten Entwurfs sind Topographie-Simulationen. Topographie-Simulationen ermöglichen die Beschreibung einer Vielzahl an zeitabhängigen Prozessschritten, die die Wafer-Oberflächen-Topographie verändern (z.B. Ätzen oder Abscheiden).

Die Geometrie, d. h. die Topographie, moderner Halbleiterbauelemente ist in den letzten Jahren durch die Verwendung neuer Materialien und vertikaler Strukturen mit hohem Verhältnis immer kritischer geworden. Allerdings zeichnen sich Topographien moderner Halbleiterbauelementen durch große Flächen mit geringer oder gar keiner geometrischen Variation (z. B. flache Teile der Topographie) und kleine Bereiche mit ausgeprägten geometrischen Variationen (z.B. scharfe Kanten) aus – sogenannte Merkmale – z.B., Graben- oder Lochstrukturen, die in modernen Speicher-Bauelementen sehr verbreitet sind. Dies erfordert prinzipiell eine Erhöhung der Auflösung von spezifischen Bereichen von Interesse des Simulations-Bereichs, um diese Merkmale akkurat zu beschreiben, was effizient durch hierarchische Gitter erreicht wird. Allerdings muss beachtet werden, dass umso höher die Auflösung des Simulations-Bereichs gewählt wird, umso höher ist der nachteilige Effekt auf die Simulationsperformanz, welches potenziell zu unpraktischen Simulationslaufzeiten führen kann. Um Simulationslaufzeiten zu verbessern, sind darum Strategien erforderlich, die automatisch topografische Merkmale erkennen und die Konfiguration des hierarchischen Gitters lokal dementsprechend anpassen.

Eine bekannte Metrik zur Messung der geometrischen Variation einer Oberfläche ist die Oberflächenkrümmung, die häufig zur automatischen Erkennung von Merkmalen verwendet wird. Dies erfordert jedoch zunächst die Verfügbarkeit von Oberflächen, und dafür stehen verschiedene Oberflächenrepräsentationen zur Verfügung, die sich grob in explizite und implizite Repräsentationen kategorisieren lassen. Je nach Art der Simulation, haben bestimmte Oberflächendarstellungen Vorteile gegenüber anderen. Implizite Oberflächen bewältigen von Haus aus das Zusammenwachsen von Materialien während einer Simulation, indes verbessern explizite Oberflächen die Leistung von Flussberechnungen.

In dieser Arbeit werden geometriesensible Algorithmen für Oberflächen aus Topographie-Simulationen entwickelt. Die Algorithmen erkennen automatisch Merkmale der Topographie. Diese erkannten Merkmale werden dann zur Verbesserung der Simulationsleistung von Topographie-Simulationen verwendet, indem die Auflösung des hierarchischen Gitters lokal angepasst wird.

Ein

allgemeiner Algorithmus zur Erkennung von Merkmalen für Topographie-Simulationen wurde entwickelt und wird hier präsentiert. Es werden vier verschiedene Methoden zur Berechnung der Oberflächenkrümmungen von Level-Set-Funktionen (d.h. implizite Oberflächendarstellung) untersucht.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Drei der untersuchten Methoden stammen aus der Literatur, und eine ist eine neuartige Verbesserung der De-facto-Standardmethode. Die neue Methode hat eine höhere numerische Genauigkeit als die anderen Methoden, während sie den Rechenaufwand nur unwesentlich erhöht, was sie zur optimalen Wahl für die Merkmalserkennung in Halbleitertopographie-Simulationen macht.

Der Algorithmus zur Merkmalserkennung wird zur Steuerung eines hierarchischen Gitterplatzierungs-Algorithmus verwendet, der die Auflösung des Simulations-Bereichs rund um Merkmale der Topographie erhöht. Dieser hierarchische Gitterplatzierungs-Algorithmus wird in einen Topographiesimulations-Arbeitsfluss integriert und verwendet, um selektives epitaktisches Wachstum von SiGe zu Evaluierungszwecken, zu simulieren. Die Simulationsgeschwindigkeit wird bei gleichbleibender Genauigkeit um bis zu 58% verbessert.

Ein weiterer entwickelter Algorithmus verbessert die Simulationsgeschwindigkeit des Ätzens dünner Materialschichten, was eine gängige Fertigungstechnik ist. Der Ätzprozess kann mit Booleschen Operationen zwischen impliziten Oberflächen simuliert werden. Abhängig von der Dicke der geätzten Materialschicht entstehen numerische Artefakte. Es wird ein spezieller Algorithmus zur Merkmalserkennung vorgestellt, der die Dicke der Materialschichten analysiert und eine minimal erforderliche Auflösung berechnet, um die Bildung dieser numerischen Artefakte zu verhindern.

Der entwickelte Merkmalserkennungs-Algorithmus kann auch zur selektiven Vereinfachung der simulierten Wafer-Topographie verwendet werden. Daher wird ein Vereinfachungs-Algorithmus für Oberflächengitter (d.h. explizite Oberflächen) vorgestellt, welcher die Merkmale der Oberfläche berücksichtigt. Dieser Vereinfachungs-Algorithmus behält eine hohe Auflösung bei den Merkmalen der Topographie, während die Nicht-Merkmale stärker vereinfacht werden. Es wird gezeigt, dass die vereinfachten Oberflächengitter die Berechnungsperformanz von Flussberechnungen, welche mit Monte Carlo Strahlenverfolgung simuliert wurden, um 15% verbessern.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Acknowledgement

First, I want to thank my primary supervisor Josef Weinbub, who is the head of the Christian Doppler Laboratory for High Performance Technology Computer-Aided Design. His steady support and encouragement throughout my studies provided lots of motivation for my research. Furthermore, I want to thank him for his precise and timely feedback when I needed it. I also want to thank my secondary supervisor Siegfried Selberherr for the excellent working environment and room for discussion he has provided.

In this context, I want to thank Dr. Andreas Hössinger from Silvaco Europe Ltd., who was the primary company partner representative within the Christian Doppler Laboratory for High Performance Technology Computer-Aided Design. His input led to several research ideas and productive discussions with my colleagues.

Additional thanks go to the Institute for microelectronics (iµe) at TU Wien, which provided a great work environment.

Research is a collaborative effort. I want to thus thank my colleagues at the iµe and laboratory. Special thanks go to Luiz Felipe Aguinsky, Michael Quell, and Florian Bogner, who gave me valuable feedback on different aspects of this work. I also want to give my thanks to Paul Manstetten for the numerous, often long, and sometimes infuriating discussions, which almost always lead to good results. My gratitude also goes to Xaver Klemenschits, Frâncio Rodrigues, Alexander Scharinger, Alexander Toifl, and Felipe Riberio for our numerous, very productive discussions.

I want to thank my partner Julia Gutschireiter who always stood by my side and supported me. Furthermore, I want to thank my dog Luci (my light-bringer), who always supported me with her unconditional love when I was frustrated, and gave me a reason to regularly leave my flat and enjoy nature.

My thanks also go to my close friends, who helped me relax and clean my head after long and intense work weeks. Special thanks go to Johannes Gams and Milena Zečević, who helped me navigate these intense times.

Finally, I want to thank my parents, who always supported me during my studies.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Contents

Abstract i

Kurzfassung iii

Acknowledgement v

Contents vi

List of Acronyms ix

List of Symbols x

1 Introduction 1
- 1.1 Research Goals 4
- 1.2 Thesis Outline 5

2 Surface Representations 6
- 2.1 Point Clouds 7
  - 2.1.1 Geometric Properties 7
- 2.2 Surface Meshes 8
  - 2.2.1 Geometric Properties 11
  - 2.2.2 Boolean Operations between Surface Meshes 11
- 2.3 Level-Set Functions (Implicit Surfaces) 12
  - 2.3.1 Geometric Properties 13
  - 2.3.2 Discretization 14
  - 2.3.3 Boolean Operations between Level-Set Functions 16
- 2.4 Switching Between Surface Representations 17
  - 2.4.1 Generating Surface Meshes from Implicit Surfaces (Marching Cubes) 17
  - 2.4.2 Generating Point Clouds from Implicit Surfaces 19
  - 2.4.3 Generating an Implicit Surface from a Surface Mesh 20

3 Surface Classification Methods 24
- 3.1 Curvature of Continuous Surfaces 24
  - 3.1.1 2D Surface (Curves) 24
  - 3.1.2 3D Surface 26
- 3.2 Curvature Calculation for Point Clouds 32

3.3 Curvature Calculation for Surface Meshes . . . 33
3.4 Curvature Calculation for Implicit Surfaces . . . 36
  3.4.1 General Formula . . . 37
  3.4.2 Shape Operator . . . 38
  3.4.3 Variation of Normal . . . 40
  3.4.4 Curvature of 2D Implicit Surfaces (Curves) . . . 40
3.5 Intuitive Approach to Surface Classification . . . 41

4 Topography Simulation and Simulation Platforms 43
4.1 Evolution of Surfaces (Advection) . . . 43
  4.1.1 Solving the Level-Set Equation . . . 44
  4.1.2 Reconstructing the Signed Distance Function (Re-Distancing) 46
  4.1.3 Velocity extension . . . 49
4.2 Surface Flux Calculation . . . 50
  4.2.1 Constant Approach . . . 52
  4.2.2 Bottom-Up Approach . . . 52
  4.2.3 Top-Down Approach (Monte Carlo Ray Tracing) . . . 53
4.3 Multi-Material Simulations . . . 54
4.4 Application of Surface Representations in Topography Simulations . . 56
4.5 Topography Simulation Workflow . . . 56
4.6 Computational Hardware . . . 58
  4.6.1 Caches . . . 59
  4.6.2 Parallelization . . . 60
  4.6.3 Benchmark Systems . . . 61
4.7 Software Tools . . . 61

5 Fast Feature Detection for Level-Set Functions 63
5.1 Feature Detection . . . 64
  5.1.1 Feature Definition . . . 64
  5.1.2 Algorithm . . . 66
  5.1.3 Curvature Based Feature Detection for Level-Set Functions . . 66
  5.1.4 Surface Normal Based Feature Detection . . . 68
  5.1.5 2D Feature Detection . . . 70
5.2 Comparison and Evaluation . . . 70
  5.2.1 Geometries and Mean Curvature Values . . . 71
  5.2.2 Parameter Study . . . 75
  5.2.3 Empirical Evaluation . . . 77
  5.2.4 Parallel Run-Time and Speedup . . . 79
5.3 Summary . . . 81

6 High Accuracy Hierarchical Grids for Topography Simulation 83
6.1 Hierarchical Grids . . . 85
  6.1.1 Nesting . . . 86
  6.1.2 Grid Generation . . . 87
6.2 Hierarchical Grid Placement . . . 89
  6.2.1 Extended Topography Simulation Workflow . . . 91

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

6.3 Benchmark Example Selective Epitaxial Growth 91
6.3.1 Simulation Setup 92
6.3.2 Example 1 94
6.3.3 Example 2 96
6.4 Summary 99

7 Thin Material Layer Refinement for Etching Simulations 100
7.1 Etching Simulations with Boolean Operations 102
7.2 Hierarchical Grid Placement for Thin Material Layers 102
7.2.1 Calculating the Minimal Required Resolution 103
7.2.2 Detection of Affected Material Layers 104
7.2.3 Thin Layer Refinement Algorithm 105
7.3 Benchmark Example LED Pixel Fabrication 106
7.3.1 Simulation Setup 106
7.3.2 Discussion 107
7.4 Summary 109

8 Surface Mesh Simplification for Efficient Top Down Flux Calculation 110
8.1 Surface Mesh Simplification 112
8.1.1 Edge Collapse Algorithm 112
8.1.2 Lindstrom-Turk Algorithm 113
8.2 Region Simplification Algorithm 113
8.2.1 Feature Detection 113
8.2.2 Mesh Partitioning and Extension of Regions 114
8.3 Comparison and Evaluation 116
8.3.1 Distance to Original Geometry 116
8.3.2 Simplification Run-Time 120
8.3.3 Flux Calculation and Monte Carlo Ray Tracing Run-Time . . 120
8.4 Summary 121

9 Summary and Outlook 123

Bibliography 127

Own Publications 141

Curriculum Vitae 143

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

List of Acronyms

2D	two-dimensional
3D	three-dimensional
AMR	adaptive mesh refinement
CPU	central processing unit
CSG	constructive solid geometry
FinFET	fin field-effect transistor
FMM	fast marching method
GGA	gate-all-around
HRLE	hierarchical run length encoding
LED	light-emitting diode
MOSFET	metal-oxide-silicon field-effect transistor
PDE	partial differential equation
SDF	signed distance function
SEG	selective epitaxial growth
SOC	system on a chip
TCAD	technology computer aided design
VSC	Vienna scientific cluster
VTK	visualization toolkit

List of Symbols

$\mathbb{R}$	real numbers
$\mathbb{Z}$	integers
$\mathbb{N}$	natural numbers
$\{\dots\}$	a set
$\vec{x}$	a vector
$\mathbf{x}$	a point
$\cup$	union
$\cap$	intersection
$A \backslash B$	relative complement
$\nabla f(\mathbf{x})$	gradient of a function
$\vec{n}$	normal vector
$\overrightarrow{\mathbf{ab}}$	vector from point **a** to point **b**
$\Vert \cdot \Vert$	L_2-norm
i, j, k, l	indices in $\mathbb{Z}$
$\cdot$	dot product
$\times$	cross product
$A^T, \vec{a}^T$	transposed matrix or vector
$\det(A)$	determinant of a matrix A
$\mathrm{trace}(A)$	trace of a matrix A
$\mathrm{adj}(A)$	adjoint matrix A
I	identity matrix
$\min(X)$	minimum of set X
$\max(X)$	maximum of set X
$\mathrm{conv}(X)$	convex hull
$\mathrm{simplex}(X)$	simplex created by the points in X
ω, σ	simplex
$\mathbf{v}_i$	a vertex (1-simplex)
e_i	a edge (2-simplex)
f_i	a triangle (3-simplex)
$\mathcal{SC}$	a simplical complex
$\mathcal{M}$	a mesh
$\phi(\mathbf{x})$	a level-set function
Δx	grid resolution
$d_E(x, y)$	Euclidean distance

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

$d_1(x, y)$	Manhattan distance
$D_x^+(\phi)$	finite forward difference
$D_x^-(\phi)$	finite backwards difference
$D_x(\phi)$	finite central difference
$D_{xx}(\phi)$	second-order finite central difference in same coordinate direction
$D_{xy}(\phi)$	second-order finite central difference in different coordinate directions
$\mathcal{L}_i$	ith layer of a sparse field level-set function
$\eta_S(\mathbf{x})$	star stencil around point
$\eta_P(\mathbf{x})$	plane stencil
$\eta_B(\mathbf{x})$	box stencil
$\gamma(\mathbf{s})$	regular parametrized curve
$k(\mathbf{t})$	curvature of a regular parametrized curve in s
n	Gauss map
κ_1, κ_2	principle curvatures
H	mean curvature
K	Gaussian curvature
$N_1(\mathbf{x})$	1-ring neighborhood of $\mathbf{x}$
$\mathcal{E}(P)$	efficiency of a patch
$d_{H'}(X, Y)$	one-sided Hausdorff distance
$d_H(X, Y)$	Hausdorff distance

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 1

Introduction

The rapid development of integrated semiconductor devices started with the metal-oxide-silicon field-effect transistor (MOSFET) in the 1960s [1], which was used to put 16 transistors on an integrated device. Over the decades, these integrated devices became more and more complex up to the modern systems of today, such as systems on a chip (SOC). These advancements in chip design are driven by the constant miniaturization of individual devices, which today reached the single-digit nanometer regime [2]. This observation of constant miniaturization is described by *Moore's law*, which was coined by Gordon Moore in an article in 1965 [3].

The constant pressure to design and fabricate ever-shirking semiconductor devices requires considerable and growing resources, ranging from research into material science to optimizing device geometries [4]. To aid the research of novel semiconductor device structures, so-called *technology computer aided design* (TCAD) tools are utilized [5]. TCAD tools are software tools that simulate multiple aspects of a semiconductor device and circuit design process. These tools allow to minimize the costs of conventional experiments [4]. Figure 1.1 shows the three branches of a TCAD toolchain: *process TCAD*, *device TCAD*, and *circuit TCAD* [5]. As is indicated in the figure, the three branches link with each other, allowing iterative cycles of development. The general workflow of designing or improving a semiconductor device starts with process TCAD. During a process TCAD simulation, the entire fabrication process is simulated, either considering an entire wafer or individual devices, depending on the type of simulation. In subsequent device TCAD simulation, the electrical characteristics of a semiconductor device are simulated. Finally, during a circuit TCAD simulation, several semiconductor devices are linked into an electrical circuit, and the behavior of an entire circuit is simulated [5].

The focus of this thesis lies in the process TCAD branch. More precisely, this work focuses on *feature scale* simulations of the device topography, and is in this work referred to as *topography simulation*.

Process TCAD
Device TCAD
Circuit TCAD

Figure 1.1: Three main branches of a TCAD toolchain.

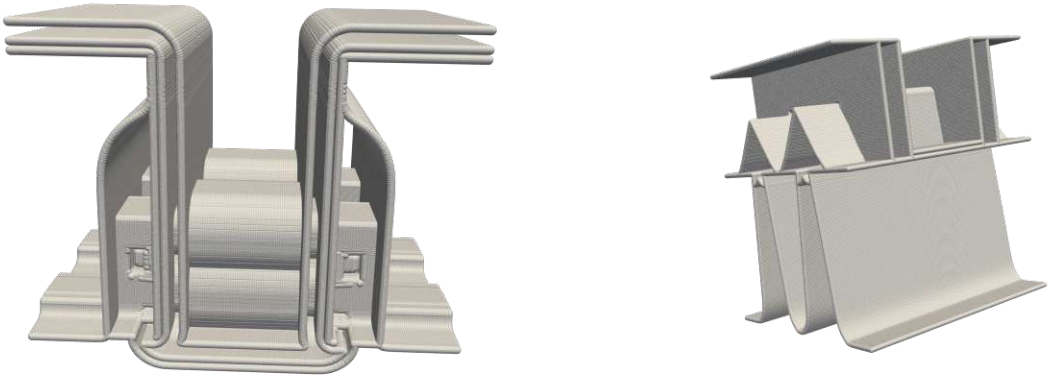
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Example processes which are simulated by topography simulations cover deposition steps that add materials to the simulated structure or, inversely, etching steps that selectively remove materials [6, 7]. The considered topography-changing processing steps are typically modeled with a time-dependent *partial differential equation* (PDE) [8]. Furthermore, there are processing steps that do not change the topography of the wafer but modify its electrical properties, like dopant implantation and diffusion [9].

Historically, the electrical properties of the simulated semiconductor device were the primary focus of process TCAD simulations. However, due to the rapid miniaturization of semiconductor devices, non-planar structures have been introduced like the fin field-effect transistor (FinFET), gate-all-around (GGA) transistors or three-dimensional (3D) NAND flash memories [10]. For these structures, the actual geometries of the devices grew significantly in importance [11], which also puts more emphasis on the representation of the discretized topography and also demands to switch from the previously sufficient two-dimensional (2D) to 3D simulations to uphold accuracy.

Figure 1.2 shows two examples of discretized topographies of modern semiconductor devices. Obviously, the quality of the discretization directly relates to the quality of the resulting topography. However, three additional aspects must be considered: First, not all parts of the topography of a semiconductor device benefit equally from a high-quality discretization (e.g., flat parts). Second, the higher the discretization quality, the higher the impact on computational performance. Third, the choice of discretization is important as it has to be tailored to the actual use case, i.e., different types of subsequent numerical processing steps require a specific discretization for optimal execution, as will be discussed in the following.

Discretization strategies of surfaces can be broadly categorized into parametric, implicit, and explicit representations [14]; this work focuses on the latter two. Implicit representations define a set of points (i.e., a surface) that satisfies an equation of the form $f(x, y, z) = c$, which is used to represent the topography of the semiconductor device. In the here considered topography simulations utilizing the so-called *level-set method*, the implicit function is expressed as a level-set function.



(a) Stacked nanosheet FET [12]

(b) FinFET [13]

Figure 1.2: Examples of two discretized semiconductor device topographies.

Level-set functions are usually discretized on a grid that stores the distance to the surface, because it allows for an efficient and robust handling of surface deformations [8, 15]. Explicit representations directly describe points, or even entire regions, of the discretized domain. An example of an explicit surface representation is a surface mesh which discretizes a surface by covering it with interconnected polygons (e.g., triangles) [16]. As indicated before, depending on the prerequisites of a given simulation, different surface representations have benefits over others [17, 18].

The surfaces depicted in Figure 1.2 show that semiconductor device topographies are characterized by relatively vast areas with little to no geometric variation (e.g., the flat areas of the topography) and comparably fewer parts with significant variation (e.g., the corners of the topography). As discussed previously, the parts of the device topography with significant geometric variation benefit from a higher quality discretization, while the other parts do not, as it would introduce unnecessary considerable computational overhead. These observations motivate the use of a domain discretization that adaptively increases the quality of the discretization depending on the geometric variation of the topography [19, 20, 21, 22, 23, 24]. A typical adaptive data structure used in process TCAD is a hierarchical grid. Hierarchical grids consist of a base grid and a plethora of nested grids at various resolution levels to realize a locally higher resolution of the discretization. These nested grids can be manually or automatically placed inside the simulation domain. As the topography changes during the simulation (e.g., simulating the gradual etching of a trench), the nested grids need to be continuously adapted, requiring a metric which allows for measuring the topography variation.

A well-known metric, originating from the field of differential geometry, is the surface curvature. The surface curvature measures the variation of a surface in a *small* neighborhood around a point on the surface [25, 26, 27, 28, 29]. The surface curvature is thus in principle ideal for the automatic classification of parts of the surface. The concept of the surface curvature of a differentiable surface can be discretized to encompass discrete surfaces, which allows this concept to be used in the here considered context. The surface curvature can be used to develop *geometry-aware* algorithms that allow to focus computational efforts on parts of a discretized surface that benefit from a higher quality of the discretization (i.e., features).

On the one hand these geometry-aware algorithms can be used to increase the resolution of the simulation domain at parts with significant geometric variation, on the other hand this information about the surface can also be used to decrease the resolution of parts of the simulation domain with little to no geometric variation. An important simulation step during a topography simulation is the so-called surface flux calculation. The surface flux describes how much of a reactant (e.g., an etchant) interacts with the surface during a simulation step. One strategy to estimate the surface flux is Monte Carlo ray tracing [30]. Monte Carlo ray tracing can be efficiently performed on surface meshes which can be extracted from level-set functions. However, these contain an impractical amount of triangles in flat regions of the topography, which increases the run-time of Monte Carlo ray tracing.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Therefore, surface mesh simplification algorithms are used to reduce the number of triangles to improve performance.

Furthermore, the adaptation of the resolution of the simulation domain may not only be guided by geometric features of the topography, but may also be guided by other metrics. Many semiconductor devices like 3D NAND flash memories or light-emitting diodes (LEDs) are fabricated by depositing thin material layers on top of each other [31, 32]. These stacked thin material layers are affected by subsequent processing steps like etching processes that create patterns on the device topography, or separates individual devices that have been fabricated on the same wafer. During the simulation of etching process steps on thin material layers, numerical artifacts occur due to an inadequate resolution of the simulation domain, which again can be mitigated by the use of hierarchical grids.

1.1 Research Goals

The primary research goal of this work is the formulation of a general algorithm that automatically detects parts of the wafer surface that benefit from a higher discretization of the simulation domain. This general algorithm is then used to, on the one hand, guide a hierarchical grid placement algorithm to selectively increase the resolution of the simulation domain at critical points to improve simulation performance while minimizing computational overhead. On the other hand, this algorithm is used to selectively coarsen surface meshes to maintain a higher resolution at critical parts of the device topography. The coarsened meshes are then used to accelerate surface flux calculations required to accurately simulate the effects of specific processing steps on the wafer surface. Furthermore, different methods of calculating the curvatures (i.e., the metric used to analyze the wafer surface) on level-set functions are investigated, and two complementary methods (i.e., performance focus, numerical accuracy focus) are obtained. Another important goal is formulating a specialized feature detection algorithm for thin material layers affected by an etching simulation.

Research Setting

The research presented in this work was conducted within the scope of the Christian Doppler Laboratory for High Performance TCAD. The Christian Doppler Association funds cooperation between companies and research institutions pursuing application-orientated basic research. In this case, cooperation was established between the Institute for Microelectronics at the TU Wien and Silvaco Inc., a company developing and providing electronic device automation and TCAD software tools. The computational resources of the Vienna Scientific Cluster have been utilized during this work.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

1.2 Thesis Outline

Chapter 2 gives a theoretical overview and reviews the discrete surface representations used in this thesis. Surface representations are defined, and approaches to calculate geometric properties for each surface presentation are introduced. Furthermore, it is discussed how a surface that is given in one representation can be transformed into another representation.

Chapter 3 introduces the mathematical concept of curvatures of a differentiable surface and their discretization for different surface representations. Several methods for calculating the surface curvature of discrete surfaces are reviewed. The surface curvatures are the primary metric that is used throughout this thesis to detect features of the considered geometries.

Chapter 4 reviews a typical topography simulation workflow as a frame of reference. It is discussed how the surface representations introduced in Chapter 2 are used during a topography simulation. The representation and evolution of different materials are introduced. Moreover, various strategies for surface flux calculations are presented. In the last part of this chapter, a brief overview of the used computer hardware and software tools in this thesis is given.

Chapter 5 introduces a new feature detection algorithm which is based on the surface curvatures. Different strategies for calculating the surface curvature on level-set functions are discussed. The investigation focuses on the qualitative results of the feature detection on the one hand and on the run-time on the other.

In Chapter 6, it is shown how the hierarchical grid placement is directed by the algorithm presented in Chapter 5. Furthermore, it is demonstrated how the performance of a practically relevant epitaxial growth simulation is improved.

Chapter 7 introduces a new algorithm for detecting features that occur when performing Boolean operations with thin material layers. Additionally, this algorithm is able to suggest a required minimal resolution to represent thin material layers after an etching simulation properly.

Chapter 8 introduces a new surface mesh simplification algorithm. This algorithm also uses the feature detection algorithm presented in Chapter 5 to guide the simplification process. The performance of surface meshes simplified with the algorithm is investigated in the context of flux calculation with Monte Carlo ray tracing.

Finally, Chapter 9 concludes with a summary of this thesis and presents ideas for future research.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

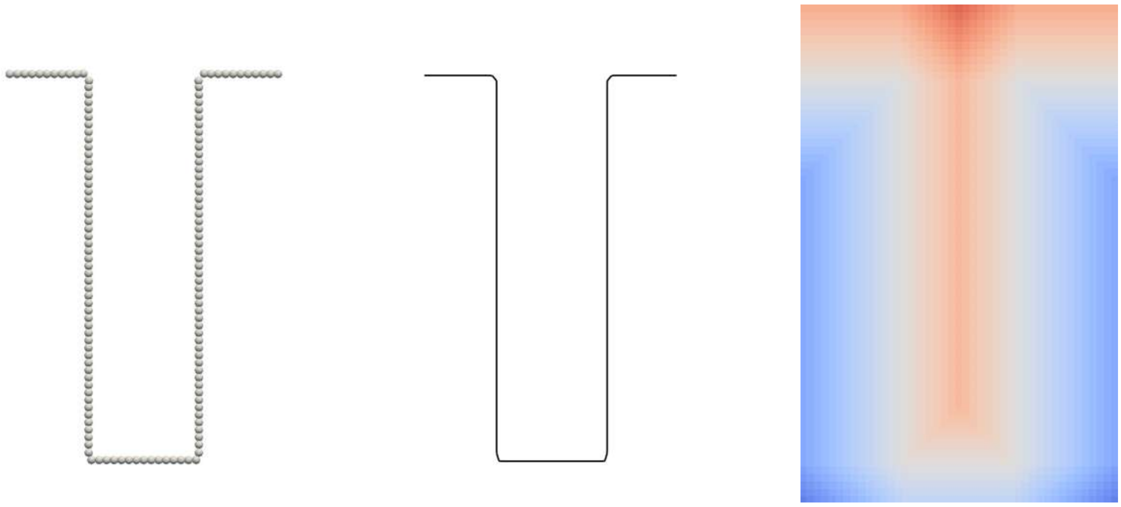
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Chapter 2

Surface Representations

Many problems in engineering investigate the properties or deformation of surfaces. For example, the change of the position of a surface when a physical process deforms it, on what parts of a surface materials accumulate, or how much of the surface area is exposed to a particle source [33, 34, 35]; the interaction at the interfaces of different liquids, gasses, and solid boundaries during a multiphase flow [36, 37]; or the detection of objects in the vicinity of a laser scanner, that creates real time scans of the area around the scanner [38, 39, 40].

For many of these aforementioned engineering problems mathematical models exist that offer solution strategies. One fundamental aspect required to solve these engineering problems is the representation of the investigated surfaces. Most surfaces originating from real world geometries that are used in engineering applications are arbitrary, and thus, do not have a readily available differentiable mathematical description. Thus, the surfaces that describe these geometries have to be discretized to numerically apply these mathematical models. An example of three surface representations (i.e., point clouds, surface meshes, and implicit surfaces) used in this work, are shown in Figure 2.1.



(a) Point Cloud **(b)** Surface Mesh **(c)** Implicit Surface

Figure 2.1: Example of a 2D trench geometry using three different surface representations.

Furthermore, some surface representations offer certain advantages depending on the mathematical models used in a simulation. Certain surface flux calculations use an explicit surface representations [18], since there exist powerful and performant algorithms to accomplish these tasks [41]. On the other hand, the merging of two different surfaces is better handled with an implicit surface representation [17].

In this chapter the discrete surface representations used in this thesis are formally defined. It is discussed how one basic geometric property (i.e., the normal vector) of the discrete surfaces can be defined. Additionally, methods of switching between the discrete surface representations are described.

2.1 Point Clouds

One of the most natural ways to represent a surface in a domain (e.g., $X \subset \mathbb{R}^n$) is to define a coordinate system and then determine a set of points relative to this coordinate system. It is then assumed that these points describe the desired surface. The formalization of this intuition is achieved through the definition of a point cloud as follows [42]:

2.1.1 Definition (Point Cloud) A set of points $\mathbf{x}_i$ with $i = 1, \ldots, k$ embedded in an n-dimensional Cartesian space $\mathbb{R}^n$, is called a *point cloud*, if the points are assumed to have some kind of spatial coherence. Additionally, it is assumed that the points in the point cloud are sampled from a piecewise continuous surface, thus a normal vector and a tangent plane exist.

The term cloud reflects the fact that a priori the relation between the points is not know. Point clouds usually originate from measurement data obtained from real world objects generated by, e.g., laser scanners [43]. However, point clouds can also be generated from other discrete surface representations, which is discussed in detail in Section 2.4.2.

2.1.1 Geometric Properties

As already mentioned in Definition 2.1.1, the spatial coherence of the points in the point cloud is only assumed and not explicit. Thus, before determining the geometric properties of a specific point $\mathbf{x}_i$ in a point cloud a suitable *neighborhood* $M_i = \{\mathbf{x}_i, \mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_m}\}$ of this point has to be determined. There are several methods of determining a neighborhood of a point $\mathbf{x}_i$ in a point cloud, for example, a K-d-tree can be spanned over the entire domain [44]. After a neighborhood has been found a quadratic optimization problem is solved. The optimization problem approximates the tangent plane of the surface by calculating a plane that minimizes the distance from the plane to all points in the neighborhood M_i. So, the normal vector of the point $\mathbf{x}_i$ is approximated by the normal vector of the plane created by the optimization problem [45, 46]. Figure 2.2 shows an illustration of the approximation of the normal vector of a 2D point cloud. The calculation of the curvatures of a point cloud is discussed in Section 3.2.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

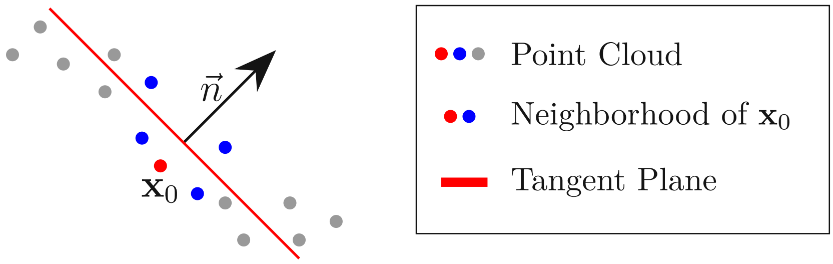


Figure 2.2: Illustration of the approximation of the surface normal of a surface represented by a point cloud. The tangent plane minimizes the distance to all points in the neighborhood.

The previously discussed approach of approximating the surface normal of a point cloud suggests that handling point clouds is computationally expensive, since both required steps (i.e., determining a neighborhood for each point in the point cloud and solving the quadratic optimization problem) are computationally demanding steps. However, when a point cloud is generated from an implicit surface (see Section 2.4.2) it is possible to maintain information about the spatial relation between the points. Furthermore, geometric properties of the implicit surface, e.g., the normal vector or the surface curvatures, can be preserved when a point cloud is generated from an implicit function. Thus, computationally expensive steps such as finding the neighborhood and solving a quadratic optimization problem can be substituted by the comparably cheap calculations on the implicit surface.

2.2 Surface Meshes

One big disadvantage of point clouds is the lack of explicit information of the position of points relative to each other. To obtain a surface representation that also stores information about its local neighborhood only considering points in space is not sufficient. Thus, a surface representation is needed that discretely represents a complex surface as a union of simple elements. The meaning of the term simple element is discussed further down in this section. Intuitively simple elements are the simplest geometries that describe a part of a surface (e.g., a line in 2D). Furthermore, these elements have to be consistently connected to each other, such that there are no partially connected (dangling) elements on the surface. A set of elements that fulfills these properties is called a *mesh*. Especially the condition about the consistent connectivity of the elements is important since otherwise the mesh would not represent a continuous surface and the definition of mathematical surface properties would not be possible.

The elements used to describe an n dimensional mesh are a combination of $1 \dots n-1$ dimensional subsets of $\mathbb{R}^n$. Before these elements can formally be defined some mathematical terminology has to be introduced [47, 48].

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

2.2.2 Definition (Affine Combination) Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \subset \mathbb{R}^n$ be a set of points in $\mathbb{R}^n$, then the linear combination

$$\sum_{i=1}^{k} \omega_i \mathbf{x}_i \text{ with } \omega_i \in \mathbb{R} \text{ and } \sum_{i=1}^{k} \omega_i = 1$$

is called an *affine combination* of the set of points X.
A point $\mathbf{x}_i$ is called *affinely independent* of X if there exists no affine combination of $\mathbf{x}_i$ in X.

2.2.3 Definition (Convex Set) A set $X \subset \mathbb{R}^n$ is called *convex* if it fulfills $\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in X$ for every two points $\mathbf{x}_1, \mathbf{x}_2 \in X$ and all $0 \leq \lambda \leq 1$.

2.2.4 Definition (Convex Hull) The *convex hull* of a set $X \subset \mathbb{R}^n$ is defined as

$$\operatorname{conv}(X) := \bigcap_{X \subset K, K \text{ is convex}} K.$$

The Convex hull of a set of points X describes the smallest convex set that contains all points of X. With these definitions the most basic mesh elements can be defined as follows:

2.2.5 Definition (k-simplex) Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k+1}\}$ be a set of affinely independent points then

$$\operatorname{simplex}(X) = \operatorname{conv}(X)$$

is called a *k-simplex*. A k-simplex has dimension k.

A k-simplex ω contains all $1 \ldots k-1$ simplices of the points used to create ω. Thus, each k-simplex of a set of n points can be described as its own object.

2.2.6 Definition (Face) Let Y be a non-empty subset of points in X then, the k-simplex $\sigma = \operatorname{simplex}(Y)$ is called a *face*; σ is a face of σ. A *proper face of* σ is any face expect σ.

When only the $1 \ldots k-1$ simplices are discussed they are called a facet:

2.2.7 Definition (Facet) Let σ be a face of a set of points X then, all (k-1) faces of σ are the *facets* of σ. Every k-simplex has k+1 facets.

A 0-simplex (vertex) is equivalent to a point in space.

The smallest faces needed to represent a surface are 1-simplices (edges) for 2D surfaces and 2-simplices (triangles) for 3D surfaces. Figure 2.3 shows examples for the first three types of simplices. To consistently represent a 3D surface with 2-simplices certain requirements have to be fulfilled [47].

2.2.8 Definition (Simplical Complex) A finite set of simplices is called a *simplical complex* $\mathcal{SC}$ if

1. $\mathcal{SC}$ contains every face of every simplex in $\mathcal{SC}$.
2. The intersection $\sigma \cap \omega$ of any two simplices $\sigma, \omega \in \mathcal{SC}$ is empty, or a face of σ and ω.

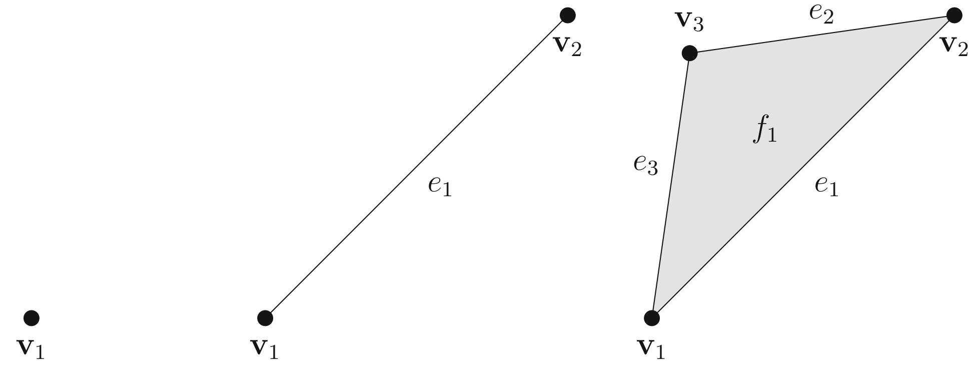
2.2.9 Definition (Triangulation) Let X be a closed set of points in $\mathbb{R}^n$. Then a simplical complex T is called a *triangulation* of X if

1. X is the set of vertices in SC.
2. $\text{conv}(X) = \overline{\bigcup_{\sigma \in \mathcal{T}} \sigma}$.

There are several definitions of meshes in literature, for this work a mesh is considered to be a special triangulation of a closed domain. The definition of a triangulation is expanded to preserve information about the boundary of the domain. Consider for example a domain that represents the geometry of a cube that has a hole, when this geometry is triangulated the holes at the bottom and top of the cube would be filled with elements. Thus, to gain a triangulation that maintains the original geometry (i.e., the hole in the cube) some elements from the triangulation have to be removed. The purpose of a mesh is to preserve the explicit information of the domain while constructing a triangulation [16].

2.2.10 Definition ((Conforming) Mesh) Let Ω be a closed domain in $\mathbb{R}^n$, and σ a simplex, then $\mathcal{M}$ is called a *mesh* of X if

1. $\Omega = \overline{\mathring{\bigcup}_{\sigma \in \mathcal{M}} \sigma}$.
2. The interior of every simplex in $\mathcal{M}$ is non-empty.
3. The intersection of the interior of two faces in $\mathcal{M}$ is empty, or a face



(a) 0-simplex (Vertex) **(b)** 1-simplex (Edge) **(c)** 2-simplex (Triangle)

Figure 2.3: Example of the k-simplices considered in this work.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The last condition in the previous definition guarantees that the faces of the mesh do not overlap with each other. Note that the terms mesh and surface mesh are used interchangeably in this work.

Mesh Quality

The shape of triangles is important for the quality, and robustness of numerical simulations using 3D surface meshes [47]. There are several metrics that measure the quality of triangles [49]. In general, equilateral triangles tend to produce the best results in numerical simulations. Thus, the smaller a single angle of a triangle gets, the worse the results of the numerical simulation. Triangles with one very obtuse angle (i.e., $\alpha > 100$ degree) or equivalently one or two very acute angles (i.e., $\alpha < 40$ degree) are called *needles*. Triangles with a bad quality are particularly problematic for numerical simulations, and a single bad quality triangle may cause the solution of a numerical simulation to diverge [16, 50].

2.2.1 Geometric Properties

A vertex of a surface mesh cannot have a uniquely defined surface normal since in 2D a vertex connects 2 edges which normals may point in different directions. The normal of an edge $\overrightarrow{\mathbf{ab}}$ with vertices $\mathbf{a} = (a_1, a_2), \mathbf{b} = (b_1, b_2)$ of a 2D mesh can be calculated by defining $a^* = a_2 - a_1$ and $b^* = b_2 - b_1$

$$\vec{n} = \frac{(-b^*, a^*)}{\|(-b^*, a^*)\|}. \tag{2.1}$$

These calculations can be interpreted in the following way: the normal vector of the edge is perpendicular to the tangent line, therefore, the normal vector is equivalent to the normalized tangent line (i.e., the edge) rotated by 90 degrees. On a 3D mesh a vertex is part of at least three edges and faces, and thus, again the surface normal is not uniquely defined. Furthermore, a similar problem effects the edges of a 3D mesh, since, each edge is part of two faces. The surface normal of a face (i.e., a triangle) with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and edges $\overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{bc}}, \overrightarrow{\mathbf{ca}}$ is easily calculated with the cross product

$$\vec{n} = \frac{\overrightarrow{\mathbf{ab}} \times \overrightarrow{\mathbf{bc}}}{\|\overrightarrow{\mathbf{ab}} \times \overrightarrow{\mathbf{bc}}\|}. \tag{2.2}$$

A similar discussion to the surface normal is required to define the surface curvatures of a mesh. On a mesh the surface curvatures can only be reasonably defined at vertices, since on edges the surface can only bend in one direction, and triangles are flat, thus they have a curvature of 0. The calculation of the surface curvatures of the vertices of a mesh is discussed in Section 3.3.

2.2.2 Boolean Operations between Surface Meshes

Constructive solid geometry (CSG) is a technique which allows for the creation of complex geometries out of simple geometries utilizing Boolean operations [51].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

This technique can, for example, be used to create the geometry of a hammer by combining a cylinder with a cube. Performing a Boolean operation between two surface meshes requires the following steps [52]:

(a) Calculate the bounding boxes of the two meshes
(b) Use the bounding boxes to identify where the meshes intersect
(c) Identify faces that are affected by the Boolean operation
(d) Determine the domains that are changed by the Boolean operation
(e) Re-triangulate the domains.

Several of the above described steps are computationally expensive. Thus, in performance oriented applications, if possible, it should be avoided to perform CSG on surface meshes. In Sections 2.3.3 and 2.4.3 an efficient strategy for CSG using implicit surfaces is discussed [53].

2.3 Level-Set Functions (Implicit Surfaces)

Surface meshes allow to effectively describe surfaces and simultaneously allow easy access to neighboring elements of the surface. Although, as discussed in Section 2.2.2, topographical changes in a surface mesh require special considerations (e.g., Boolean operations on surface meshes). A surface representation that naturally handles topographical changes without special considerations, and still maintains the easy access to neighboring elements of the surface is an implicit representation of the surface [17].

An implicit surface can be motivated by considering a soap bubble, the soap film separates the inside of the soap bubble from the outside. Thus, the surface of the soap bubble is described by the set of points that separates the inside of the soap bubble from the outside. Implicit surfaces describe a surface by constructing an interface that splits a domain (e.g., $S \subset \mathbb{R}^n$) into an inside and outside, thereby, separating the domain. The *level-set method* introduced by Osher and Sethian utilizes implicit surfaces for a robust simulation of advancing fronts [54].

In an open domain $\Omega \subset \mathbb{R}^n$ a subset $S \subset \Omega$ that describes a curve (2D) or a surface (3D), splits the domain into an inside $\Omega^- \subset \Omega$ and an outside $\Omega^+ \subset \Omega \backslash \Omega^-$. The subset that splits the domain into the inside and the outside is called the *interface* (i.e., S). The interface can be described by an iso-contour of a function ϕ defined in Ω.

2.3.11 Definition (Level-Set function) A continuous function $\phi(\mathbf{x})$ that fulfills

$$\phi(\mathbf{x}) = \begin{cases} > 0 & \mathbf{x} \in \Omega^+ \\ = 0 & \mathbf{x} \in S \\ < 0 & \mathbf{x} \in \Omega^- \end{cases},$$

is called a *level-set function.*

In literature such functions are sometimes also referred to as *signed distance functions.*

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

2.3.12 Definition (Zero Level-Set) The set of points defined by

$$S = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}) = 0\},$$

is called the *zero level-set*.

It should be mentioned here that the choice of negative values for the inside and positive values for the outside is a common arbitrary convention. Furthermore, any other value than zero could be chosen to define the interface of the level-set function. However, choosing 0 allows for a straightforward distinction between the inside and the outside of the domain, by checking the sign of the level-set function. In this work, 2D and 3D level-set functions are considered. An example of a 2D level-set function with the interface S is shown in Figure 2.4.

2.3.1 Geometric Properties

If the gradient of the level-set function ϕ exists it is given by

$$\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right), \tag{2.3}$$

this expression can be naturally reduced to 2 dimensions. The gradient is orthogonal to the iso-contours of the level-set function ϕ [15]. Therefore, normalizing the gradient of the zero level-set yields the normal vector of the surface

$$\vec{n} = \frac{\nabla\phi}{\|\nabla\phi\|}. \tag{2.4}$$

Furthermore, this geometric interpretation motivates the investigation of other geometric properties like the surface curvatures of the zero level-set, which is discussed in Section 3.4.

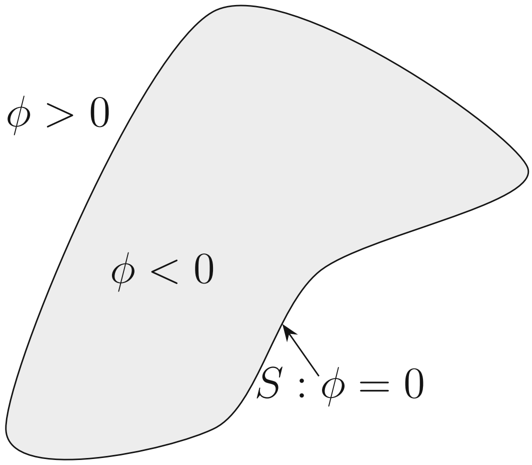


Figure 2.4: Schematic of a 2D level-set function with the zero level-set S.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

2.3.2 Discretization

Modern computer systems only have a limited amount of resources, therefore, it is unfeasible to store every value of an implicit function $\phi(\mathbf{x})$. In the level-set method the level-set function $\phi(\mathbf{x})$ is discretized on a grid with grid resolutions $(\Delta x, \Delta y, \Delta z)$. When the resolution is equal in all coordinate directions and the coordinate directions are orthogonal the grid is called a *regular grid* (or *Cartesian grid*) with grid resolution Δx. The level-set functions considered here are always discretized on a regular grid. In this work the lattice points of the regular grid are indexed as integers and are called *grid points*. The lines that pass through all grid points when one coordinate is fixed are called *grid lines*. Another important concept for discretized level-set functions is the *grid cell*.

2.3.13 Definition (Grid Cell) Let $\mathbf{g} = (i, j) \in \mathbb{Z}^2$ be a grid point. Then, the volume enclosed by

$$[(i - 0.5)\Delta x, (i + 0.5)\Delta x] \times [(j - 0.5)\Delta x, (j + 0.5)\Delta x]$$

is called a *grid cell*.

This definition can naturally be adapted for 3D grid cells.

The values of the level-set function $\phi(\mathbf{g})$ on the grid points $\mathbf{g} = (i, j) \in \mathbb{Z}^2$ are referred to as the *ϕ-values*. The ϕ-values describe the distance from a grid point to the zero level-set. Thus, the level-set function $\phi(\mathbf{g})$ can be written as

$$\phi(\mathbf{g}) = \begin{cases} +d(S, \mathbf{g}) & \mathbf{g} \in \Omega^+ \\ 0 & \mathbf{g} \in S \\ -d(S, \mathbf{g}) & \mathbf{g} \in \Omega^-, \end{cases} \tag{2.5}$$

where $d(p, q), p, q \in \mathbb{R}^n$ is a metric. The level-set method initially presented by Osher and Sethian uses the *Euclidean distance* as the metric [54]:

$$d_E(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}, \tag{2.6}$$

with $p, q \in \mathbb{R}^n$.

However, this is not the only metric that can be used to discretize the level-set function. Whitaker proposed a level-set framework that uses the *Manhattan norm* (also known as the *taxicab metric*) to discretize the level-set function [55]

$$d_1(p, q) = \sum_{i=1}^{n} |p_i - q_i|, \tag{2.7}$$

where again $p, q \in \mathbb{R}^n$.

Furthermore, since not all ϕ-values of the level-set function have to be known all the time, so-called *narrow band* techniques have been introduced [56]. The idea of the narrow band is to only consider a subset of ϕ-values that is sufficient to describe the surface, and to perform calculations to evolve the zero level-set. An integer multiple of the grid resolution $i\Delta x$ is called the width of the narrow band, which characterizes the subset of ϕ-values in the narrow band.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Euclidean Normalization

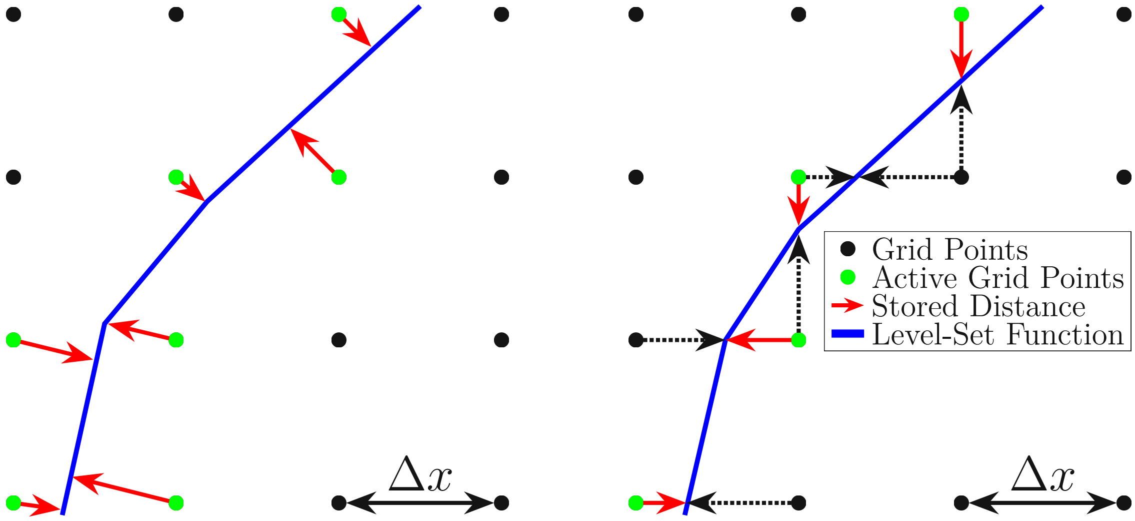
When the Euclidean distance is used to describe the level-set function (see Figure 2.5a) it fulfills the *Eikonal equation* with $F(\mathbf{x}) = 1$ [15]. In an open domain $\Omega \subset \mathbb{R}^n$ the Eikonal equation can be formulated as follows [57]

$$\begin{aligned} \|\nabla\phi(\mathbf{x})\| &= F(\mathbf{x}),\ \forall \mathbf{x} \in \Omega \\ \phi(\mathbf{x}) &= G(\mathbf{x}),\ \forall \mathbf{x} \in S. \end{aligned} \tag{2.8}$$

This equation describes a wave front, where $G(\mathbf{x})$ describes the initial values of the front, emerging form S that travels with a speed of $\frac{1}{F(\mathbf{x})}$ into the domain Ω. The speed has to be positive since otherwise the wave would not propagate into the entire domain, and the problem would not be well-defined.

Manhattan Normalization (Sparse Field Method)

Considering Equation 2.5 every point in $\mathbb{R}^n$ has a distance to the zero level-set. When the level-set function is discretized (see Equation 2.5) only a certain subset of these points is used. This subset of points can be further reduced by the following considerations. A level-set function that passes through a grid cell intersects at least two grid lines, so the distance to the zero level-set can be described by the distance to these intersections (i.e., the Manhattan distance). So the zero level-set is described by the shortest grid line intersection to the adjacent grid point. Furthermore, the zero level-set is the interface between the inside and the outside of a volume. Thus, it is sufficient to only store the shortest distance on one side of the level-set function, except for the rare case in which the level-set function lies directly on a grid point and has a value of 0. Figure 2.5b shows an illustration of this deliberation.



(a) Euclidean normalization **(b)** Manhattan normalization

Figure 2.5: Illustration of the difference between a level-set function normalized with the Euclidean and Manhattan metric.

Which leads to the following description of the level-set function [55, 58]

$$\phi^{\mathcal{L}_i}(\mathbf{x}) = \begin{cases} \mathcal{L}_i = \{\mathbf{x} : i - 0.5 \leq \phi(\mathbf{x}) < i + 0.5\}, & i < 0 \\ \mathcal{L}_i = \{\mathbf{x} : -0.5 \leq \phi(\mathbf{x}) \leq 0.5\}, & i = 0 \\ \mathcal{L}_i = \{\mathbf{x} : i - 0.5 < \phi(\mathbf{x}) \leq i + 0.5\}, & i > 0 \end{cases} . \tag{2.9}$$

Where the level-set values are normalized to a unit cell of length 1. This representation of the level-set function is also referred to as the *sparse field* representation. However, one drawback of using the Manhattan metric is that the level-set function no longer fulfills the Eikonal equation (see Equation 4.12). Thus, $|\nabla\phi(\mathbf{x})| = 1$ can no longer be assumed in the entire domain Ω.

Difference between Manhattan and Euclidean Normalization

The primary difference between Euclidean and Manhattan normalized level-sets is the way the signed distance function is constructed (see Section 4.1.2). When the Manhattan normalization is used it is sufficient to only store grid points at one side of the zero level-set (i.e., on the inside or the outside). This is also possible when the Euclidean normalization is used, however, this negatively impacts the quality of the ϕ-values in the entire domain. Thus, when the Euclidean normalization is used a narrow band around the interface is stored. Figure 2.5 shows an illustration of a discretized level-set function in both normalizations. This example shows that a Manhattan normalized level-set function has to store significantly fewer ϕ-values of grid points to describe the zero level-set. Naturally, a Manhattan normalized level-set has a bigger error in the numerical approximation. However, this error is small enough so that it does not significantly affect the simulation results [55].

2.3.3 Boolean Operations between Level-Set Functions

Boolean operations between implicit surfaces (e.g., level-set functions) are described by the following functions [8]

$$\text{Union:} \qquad \phi_A \cup \phi_B = \min(\phi_A, \phi_B) \tag{2.10}$$

$$\text{Intersection:} \qquad \phi_A \cap \phi_B = \max(\phi_A, \phi_B) \tag{2.11}$$

$$\text{Relative Complement:} \qquad \phi_A \backslash \phi_B = \max(\phi_B, -\phi_A). \tag{2.12}$$

Figure 2.6 shows an illustration of each of the Boolean operation between two level-set functions, which can be interpreted as Boolean operations between two volumes. The big advantage of performing Boolean operations with level-set functions is its performance, since the information about the inside and the outside of the volumes that are part of the Boolean operation is already encoded into the discretization, Boolean operations on level-set functions only need to iterate over the points of the level-set functions and compare ϕ-values [59, 60]. After a Boolean operation has been performed the signed distances of the resulting level-set functions may not be accurate anymore, so the signed distance function has to be reconstructed (see Section 4.1.2).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

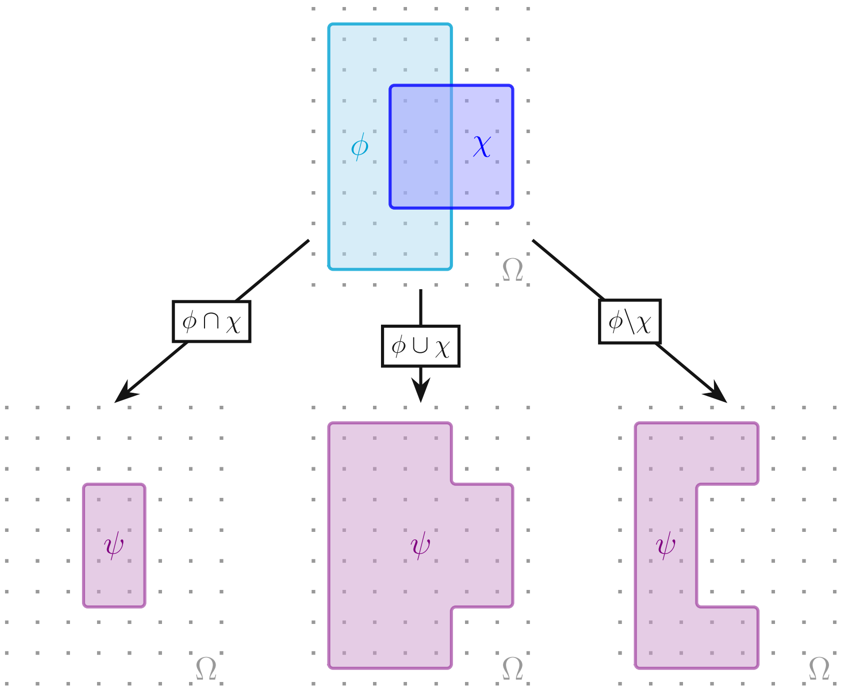


Figure 2.6: Illustration of the Boolean operations union, intersection, and relative complement between two level-set functions and the resulting level-set function. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

2.4 Switching Between Surface Representations

During a single simulation step several different mathematical models may be utilized to achieve the final result of this simulation step. These mathematical models can require different surface representations to achieve complementary goals (e.g., higher performance and more robust results). So a discussion on how to switch between different surface representations used during process TCAD simulations is presented in this section. Figure 2.7, shows the three discussed surface representations and the in this work discussed conversion between them.

2.4.1 Generating Surface Meshes from Implicit Surfaces (Marching Cubes)

A widely used and performant approach to extract a surface mesh from an implicitly defined surface is the marching cubes algorithm [62]. Depending on the dimension of the implicit surface the algorithm is also referred to as the marching squares algorithm (i.e., for 2D implicit surfaces) [63].

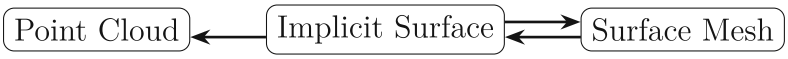
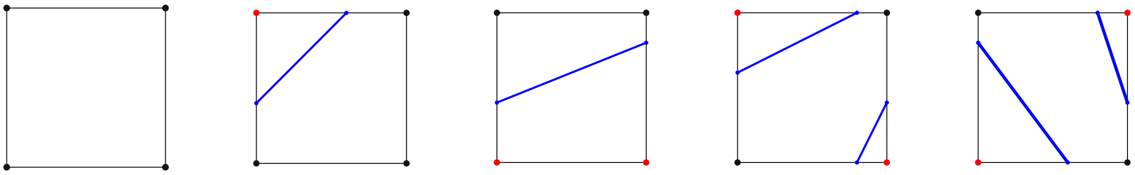


Figure 2.7: The arrows represent the conversions between the surface representations relevant for topography simulations.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The marching cubes algorithm iterates over all chunks (i.e., the 4, or 8 grid points making up the corners of a square, or a cube) of the implicitly defined function. If at least one sign of the ϕ-values making up the chunk is not equal to the others at least one face of the surface lies within the chunk. There are $2^4 = 16$ possible intersections in $2D$ and $2^8 = 256$ in $3D$, which can be reduced to 4, and 14, cases respectively, through the exploitation of symmetries. A lookup table is created that stores all possible grid intersection. Figure 2.8 shows the element of the lookup table for 2D surfaces, and Figure 2.9 shows some examples for 3D surfaces. The required surface elements for the current chunk are retrieved from the lookup table and added to the surface mesh. An example of this process on a 2D level-set function is shown in Figure 2.10.



(a) Case: 1 **(b)** Case: 2 **(c)** Case: 3 **(d)** Case: 4a **(e)** Case: 4b

Figure 2.8: All possible chunks intersections, except for rotations, of the marching squares algorithm. The red vertices indicate the grid points where the signs of the implicit function change.

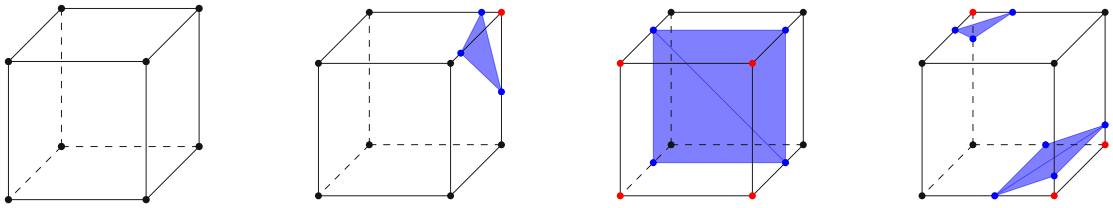


Figure 2.9: Triangulation examples of four chunks created by the marching cubes algorithm. The red vertices indicate the grid points where the signs of the implicit function change.

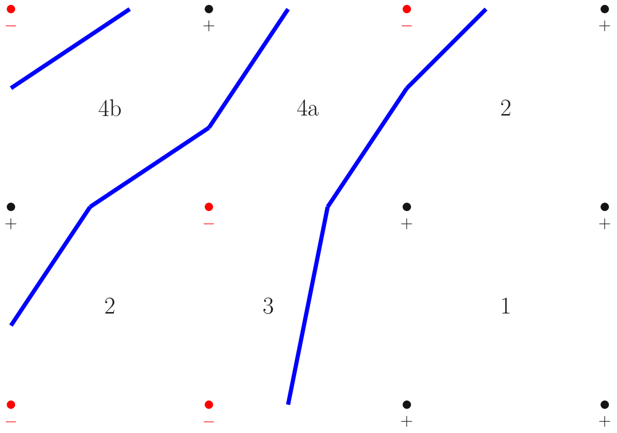


Figure 2.10: Example of the marching squares algorithm operating on a 2D level-set function. The signs of the ϕ-values of each chunk are checked and the corresponding triangulation is inserted. The numbers inside the chunks relate to the possible triangulation shown in Figure 2.8.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 2.11 shows a 3D surface mesh extracted with the marching cubes algorithm. This mesh has several bad mesh elements (see red boxes). Furthermore, the marching cubes algorithm works on the chunks of the grid thus, it creates several mesh elements that may not be required to represent the geometry. In Chapter 8 a surface simplification algorithm is presented that addresses the problem of surface elements that do not contribute information about the geometry.

2.4.2 Generating Point Clouds from Implicit Surfaces

A level-set function can easily be converted into a point cloud, since due to the construction of the level-set function we know the distance from each grid point to the zero level-set. When the level-set function satisfies the Eikonal equation (see Equation 4.12) then the closest point on the zero level-set ($\mathbf{x}_S$) can be calculated with [8]

$$\mathbf{x}_S = \mathbf{x} - \phi(\mathbf{x})\|\nabla(\phi(\mathbf{x}))\|, \tag{2.13}$$

where $\phi(\mathbf{x})$ denotes the ϕ-value of the level-set function on the grid point $\mathbf{x}$.

For level-set functions using the Manhattan normalization Equation 2.13 should not directly be used to create an explicit point cloud. A level-set function utilizing the Manhattan normalization only stores the distance to the intersection point between the level-set function and the closest Cartesian grid line (see Figure 2.5). Thus, a different factor has to be used in Equation 2.13 that takes into account the different normalization of the ϕ-values. Figure 2.12 shows an illustration of the calculation of this factor. To calculate the angle α the Cartesian coordinate direction of the grid line intersection has to be calculated, which is achieved by checking the adjacent grid points. The angle α can now be determined by calculating the inner product of the surface normal in the grid point $\mathbf{x}$ and the previously calculated coordinate direction. Finally, the distance to the surface (d_n) can be calculated with the relation between the trigonometric functions and the right triangle:

$$d_n = \cos(\alpha)\phi(\mathbf{x}). \tag{2.14}$$

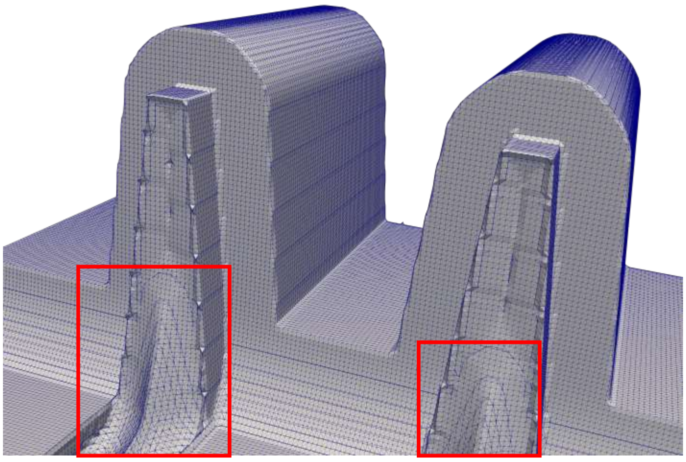


Figure 2.11: Surface mesh extracted with the marching cubes algorithm from the fabrication simulation of a FinFET. The red boxes indicate parts of the surface with bad mesh elements.

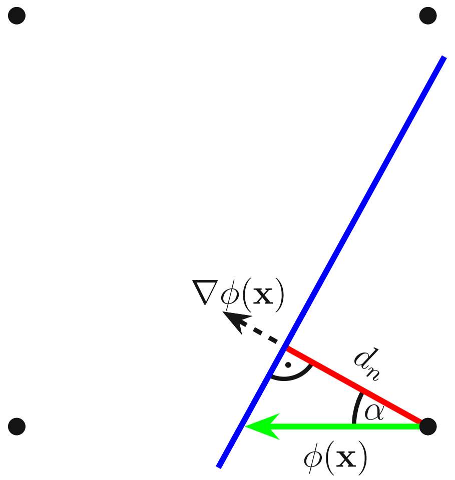


Figure 2.12: Illustration of the normal distance calculation in a Sparse Field level-set.

The point on the surface of the implicit surface can be calculated by using Equation 2.13 and substituting d_n for $\phi(\mathbf{x})$. A performance-focused approximation to the calculations described above is to use the maximum coordinate value of the normal vector $\max(\|\nabla(\phi(\mathbf{x}))\|)$. The normal vector of the implicit surface is required to compute the point cloud, in both cases no additional calculations or memory accesses have to be performed.

One big advantage of point clouds generated from implicit surfaces is that there still exists a relation between a point on the explicit surface (i.e., the point cloud) and the associated grid point of the implicit surface. As long as the topography of the point cloud is not changed, this association allows for a direct mapping of calculated values on the point cloud (e.g., surface flux originating from Monte Carlo ray tracing simulations) back to the grid points of the level-set function.

2.4.3 Generating an Implicit Surface from a Surface Mesh

Another important conversion is the conversion from surface meshes to implicit surfaces. Basic geometries are straightforward to describe as surface meshes and Boolean operations are easily performed with implicit surfaces. Thus, creating initial geometries using CSG is realized by creating surface meshes of simple geometries (e.g., planes or cuboids) and subsequently convert them into implicit surfaces to perform the Boolean operations. The idea of the conversion for Manhattan and Euclidean normalized level sets is similar. However, the implementation differs quite significantly, so the two methods are discussed separately.

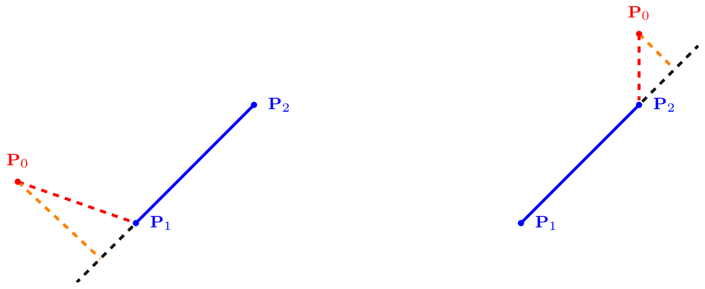
Euclidean Normalization

The conversion algorithm for Euclidean normalized level-set functions starts by calculating a bounding box of the surface mesh. Each grid point in the bounding box is visited and the distance to the surface mesh is calculated.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

If the normal distance to the closest surface mesh face is smaller than a desired threshold parameter (e.g., the thickness of the narrow band) the distance to the surface mesh is then inserted into the level-set data structure at the grid point, otherwise the next grid point is considered. The distance calculation differs depending on the dimension of the surface mesh. Figure 2.13 shows the three cases excluding reflections of how a point can lie in relation to a 2D surface segment (e.g., an edge) between $\mathbf{P}_1$ and $\mathbf{P}_2$. In case 1 the projection of the point $\mathbf{P}_0$ onto the line created by the line segment lies closest to $\mathbf{P}_1$. Thus, the shortest normal distance to the line segment is the distance to $\mathbf{P}_1$. The same argument is used for case 2 concerning the point $\mathbf{P}_2$. When the projection of $\mathbf{P}_0$ lies directly on the line segment the normal distance is given by the distance between the intersection point and P_0.

For 3D surface meshes a more complex approach is required [64]. Consider the triangle created by the points $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$. First a rotation and translation matrix is calculated that rotates one point of the triangle (e.g., $\mathbf{P}_1$) into the origin and the other two points into the YZ-plane. This matrix is then used to also transform the point $\mathbf{P}_0$. Thus, the distance from $\mathbf{P}_0$ to the triangle is given by the x-coordinate of $\mathbf{P}_0$ (see Figure 2.14), if the projection point lies inside the triangle. Otherwise, it has to be determined which edge or vertex is closest to the projection of $\mathbf{P}_0$ and the distance to vertex or edge intersection has to be calculated.



(a) Case: 1 **(b)** Case: 2 **(c)** Case: 3

Figure 2.13: Illustration of the three cases a point can lie relative to a line segment, excluding reflections.

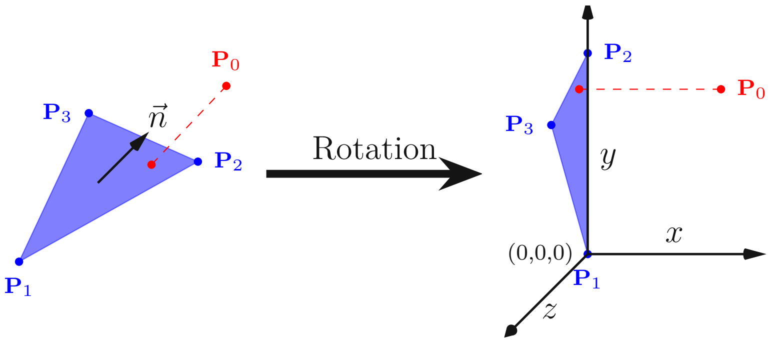


Figure 2.14: Illustration of calculating the normal distance from a point to a triangle in 3D.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The sign of the point $\mathbf{P}_0$ (e.g., if it lies on the inside or outside of the volume) is determined by calculating the normal vector $\vec{n}$ of the triangle and the point $\mathbf{P}_1$ to calculate the sign of:

$$\vec{n} \cdot (\mathbf{P}_0 - \mathbf{P}_1).$$

If the sign is positive $\mathbf{P}_0$ lies on the outside, when the sign is 0 the point lies directly on the face, and on the inside otherwise.

Manhattan Normalization

In a level-set framework that uses the Manhattan distance the distance calculation has to be adapted. It has to be checked if the grid lines in each coordinate direction intersect the faces of the surface mesh. The algorithm for Manhattan normalized level-set functions is an adaptation of the algorithm used for Euclidean level-sets. It is sufficient to calculate a bounding box for each face of the surface mesh and then determine all grid line intersections for that face.

As with the Euclidean normalized level-set the intersection test differs depending on the dimension of the surface. In the 2D case each Cartesian coordinate direction is checked if it intersects the line segment. This is achieved by calculating the difference between the x and y coordinates of the point $\mathbf{P}_0$ and the points $\mathbf{P}_1$ and $\mathbf{P}_2$ of the line segment. Afterwards the sign of the calculated differences is checked. If the sign of the differences changes then the current grid line intersects the line segment and the intersection point q is calculated. Figure 2.15 shows an illustration of the grid line intersection test.

For 3D surfaces the intersection test is more complicated. To determine if a grid line intersects a triangle in 3D the signed areas of the triangle have to be calculated. Given a triangle with the points $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, and a line defined by a point $\mathbf{P}_0$ and a unit vector $\vec{v}$. The signed areas of the triangle projected into the plane perpendicular to the vector $\vec{v}$ are given by [58]

$$A_i = \frac{(\mathbf{P}_{i+1} - \mathbf{P}_0) \times (\mathbf{P}_{i+2} - \mathbf{P}_0)}{2} \cdot \vec{v} = \frac{\det(\mathbf{P}_{i+1} - \mathbf{P}_0, \mathbf{P}_{i+2} - \mathbf{P}_0, \vec{v})}{2} i \in 1, 2, 3.$$

If all three areas have the same sign, then the line intersects the triangle. Figure 2.16 shows an illustration of a succeeding and a failing intersection test. If the intersection test is successful the intersection point can be calculated as follows [58]

$$q = \frac{\sum_{i=1}^{3} A_i \cdot \mathbf{P}_i}{\sum_{i=1}^{3} A_i}. \tag{2.15}$$

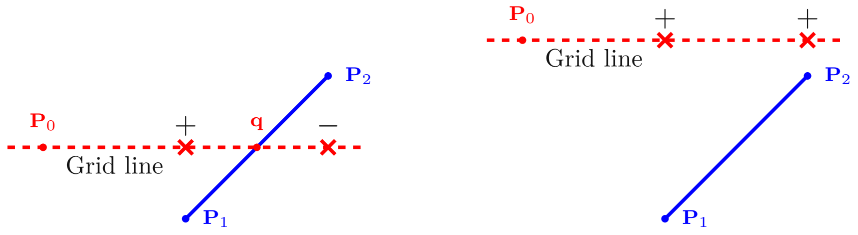
Thus, the ϕ-value of the grid point $\mathbf{P}_0$ can be calculated by evaluating

$$\phi(\mathbf{P}_0) = \pm d_1(\mathbf{P}_0 - \mathbf{q}). \tag{2.16}$$

The sign can be determined by consulting the sign of the coordinate of the normal vector of the face.

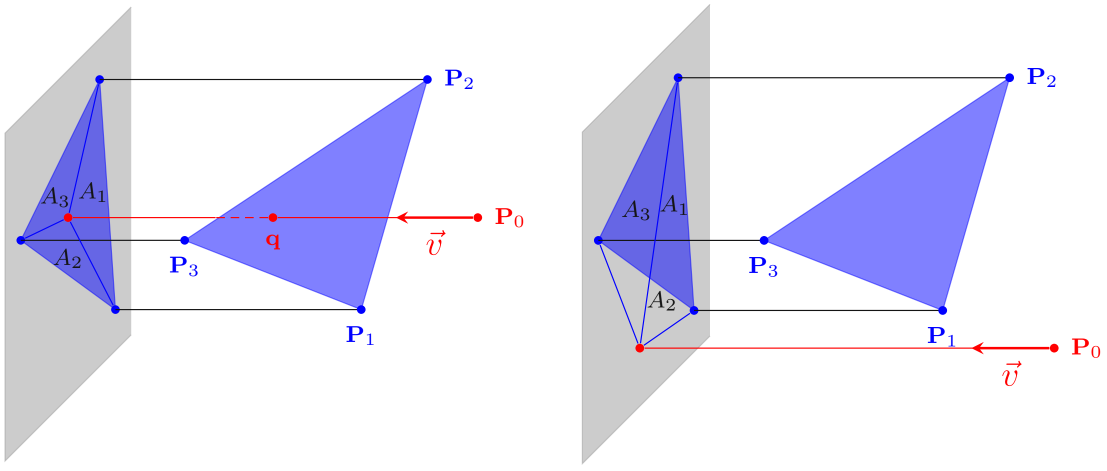
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Succeeding intersection test. **(b)** Failing intersection test.

Figure 2.15: Illustration of the grid line intersection calculation with a line segment in 2D.



(a) The grid line intersects the triangle since all areas A_i have the same sign.

(b) The grid line does not intersect the triangle since the area A_2 has a different sign than A_1 and A_3.

Figure 2.16: Illustration of the grid line intersection calculation with a triangle in 3D. The triangle is projected into the plane perpendicular to the vector $\vec{v}$.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Chapter 3

Surface Classification Methods

Among the goals of this thesis is to devise an algorithm that automatically detects parts of a discretized surface, originating from a simulation workflow, that benefits from a higher resolution of the simulation domain. This requires mathematically analyzing the surfaces so that the regions of interest can be efficiently detected.

In chapter 2 three different surface representations and how to obtain one of their basic geometric properties (i.e., the surface normal) have been discussed. Surfaces have more geometric properties than just the surface normal vector (e.g., the tangent plane). Another property of orientable surfaces, which will be discussed later, is the surface curvature. The surface curvature describes how the direction of the normal vector changes when it is moved from one point on the surface to a neighboring point in an infinitesimal area around the point. Thus, the surface curvature can be used to classify parts of a surface.

In this chapter the mathematical concept of surface curvature on continuous surfaces is introduced. This is followed by a discussion on how these concepts can be discretized for the previously discussed surface representations. For implicit surfaces several methods for calculating surface curvatures are reviewed. The last section in this chapter discusses how discrete surfaces can be analyzed without prior knowledge of surface curvatures.

3.1 Curvature of Continuous Surfaces

The constructions and definitions presented within this section are extracted and simplified from [25]. Only the results needed for further investigations in this work are discussed and proofs are omitted. The goal of this section is to give a rigorous understanding of how the curvatures of a continuous surface are constructed and which properties of the surface they express.

3.1.1 2D Surface (Curves)

First 2D surfaces are discussed. In differential geometry 2D surfaces are referred to as *parametrized curves* and are defined as follows [25]:

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3.1.1 Definition (Parametrized Curve) Let $I = (a, b)$ be an open interval then the differentiable map $\gamma : I \to \mathbb{R}^2$ is called a *parametrized curve*.

For example consider the map $\gamma(t) = (t^3, t^2)$, the parametrized curve created by this expression is shown in Figure 3.1. This curve has a point $\mathbf{p} = (0, 0)$ for which $\gamma'(\mathbf{p}) = 0$, these points are called *singular points*.

For the further investigations in this section it is important that a parametrized curve $\gamma(t)$ does not have any singular points. Thus, the definition of a parametrized curve is extended to a *regular parametrized curve*.

3.1.2 Definition (Regular Parametrized Curve) Let $\gamma : I \to \mathbb{R}^2$ be a parametrized curve. If $\gamma'(t) \neq 0$ for all t, then γ is called a *regular parametrized curve*.

Excluding curves with singular points allows for the definition of the length of a curve. The *arc length* of a regular parametrized curve is defined as follows:

3.1.3 Definition (Arc Length) Let $\gamma : I \to \mathbb{R}^2$ be a regular parametrized curve. Then, for a given $t_0 \in I$ the arc length of the curve γ is defined as

$$s(t) = \int_{t_0}^{t} \|\gamma'(x)\| dx.$$

A regular curve that fulfills $\|\gamma'(t)\| = 1$ for all $t \in I$ is called a *curve parametrized by arc length*. For such curves the arc length is given by $s(t) = t - t_0$.

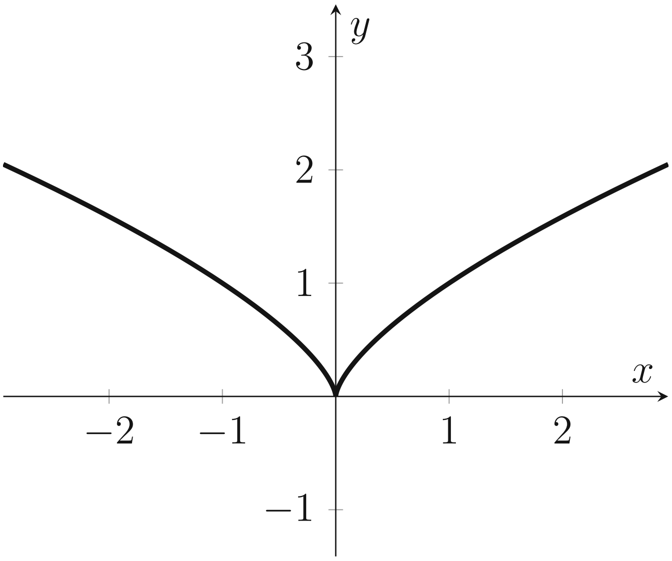


Figure 3.1: Illustration of a parametrized curve (semicubical parabola) with a singular point in $(0, 0)$.

TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The Tangent Line and Curvature of a Curve Parametrized by Arc Length

In each point $\gamma(t)$ on a curve parametrized by arc length there exists a straight line, defined by $\gamma'(t)$ which has a length of $\|\gamma'(t)\| = 1$. This vector passes through the point $\gamma(t)$ and is called the *tangent line* $(\gamma'(t))$. In singular points it is not possible to define the tangent line and so, it is not possible to discuss geometric properties of such curves.

Consider a curve parametrized by arc length γ which in a point $\gamma(t)$ fulfills $\gamma''(t) \neq 0$, in this case a unit vector $\vec{n}(t)$ is well-defined by $\gamma''(t) = \|\gamma''(t)\|\vec{n}(t)$. Furthermore, differentiating $\gamma'(t) \cdot \gamma'(t) = 1$ yields $\gamma''(t) \cdot \gamma'(t) = 0$, which proves that $\gamma''(t)$ and $\gamma'(t)$ are orthogonal. Therefore, the vector $\vec{n}(t)$ is normal to $\gamma'(t)$ and is called the normal vector in $\gamma(t)$.

Since $\gamma'(t)$ has length 1, the norm of the second derivative $|\gamma''(t)|$ measures the rate of change of the angles of the neighboring tangent planes at the tangent plane in $\gamma(t)$. In other words the curve parametrized by arc length $\gamma(t)$ pulls away from the tangent line in $\gamma(t)$ at a rate determined by $\|\gamma''(t)\|$.

3.1.4 Definition (Curvature of a Curve Parametrized by Arc Length) Let $\gamma : I \to \mathbb{R}^3$ be a curve parametrized by arc length and $t \in I$. Then $\kappa(t) := \|\gamma''((t))\|$ is called the *curvature* of γ at t.

A straight line $\gamma(t) = \vec{u}t + \vec{v}$ with constant vectors $\vec{u}, \vec{v}$ has a curvature of 0. Inversely if $\|\gamma''(t)\| = 0$ is integrated the curve $\gamma(t)$ has to be a straight line. Thus, the curvature of a curve can be used as a metric to distinguish between parts of a curve that are flat and parts that are not, e.g., curved.

Furthermore, it can be shown that each regular parametrized curve can be expressed as a curve parametrized by arc length [25]. Thus, the concept of the curvature can be expanded to any regular parametrized curve. The curvature of the regular parametrized curve $\gamma(t)$ is given by the curvature of the reparametrized (i.e., parametrized by arc length) curve $\gamma^*(t)$.

3.1.2 3D Surface

In this section a similar construction to the one presented in the previous section is given for 3D surfaces. However, defining geometric properties like the tangent plane or the curvature for 3D surfaces (which are also referred to as regular surfaces) requires additional considerations. The first major difference is that a regular surface is not defined as a function but as a set [25].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3.1.5 Definition (Regular surface) A subset $S \subset \mathbb{R}^3$ is called a *regular surface*, if for each point $\mathbf{p} \in S$ there exists an open set $U \subset \mathbb{R}^2$, a neighborhood $V \in \mathbb{R}^3$, and a map

$$\mathbf{x} : U \to V \cap S \subset \mathbb{R}^3,$$

such that:

1. $\mathbf{x} := (x(u,v), y(u,v), z(u,v))$ with $u, v \in U$ is differentiable.
2. $\mathbf{x}$ is a homeomorphism: $\mathbf{x}^{-1} : V \cap S \to U$ exists and is continuous.
3. For each $q \in U$, the differential $d\mathbf{x}_q : \mathbb{R}^2 \to \mathbb{R}^3$ is one to one.

To further elaborate the meaning of Definition 3.1.5 consider Figure 3.2. A regular surface is a subset of the plane that gets deformed in $\mathbb{R}^3$. The deformation is handled in such a way that the resulting surface has no sharp points, edges or self-intersections. This definition allows for the definition of a tangent plane in each point on the surface and ensures that the surface is smooth enough to apply methods from calculus. The map $\mathbf{x}$ is called a *parametrization* of the surface S. A surface S may have several parametrizations, however, certain properties of the surface do not depend on the parametrization but only on the set S.

Definition 3.1.5 is a technical definition of a regular surface, which is useful for the theoretical investigation of surfaces and their properties. In this work, the geometric properties of discretized surfaces are investigated, thus, two less technical descriptions of regular surfaces are given which result from Definition 3.1.5.

3.1.6 Proposition If $f : U \subset \mathbb{R}^3 \to \mathbb{R}$ is a differentiable function and $a \in f(U)$ fulfills $0 \notin f'(f^{-1}(\{a\})) = \{f'(\mathbf{x}) | f(\mathbf{x}) = a,\ \mathbf{x} \in U\}$, then $f^{-1}(\{a\})$ is a regular surface in $\mathbb{R}^3$.

Proposition 3.1.6 can be interpreted in a more descriptive way. The set of points in $\mathbb{R}^3$ that fulfill an implicitly defined function $f(x,y,z) = a$, which has a non-vanishing gradient in each point of the image, describes a regular surface. Consider, for example, the following function $f(x,y,z) = x^2 + y^2 + z^2 - 1$, all points except for $\mathbf{p} = (0,0,0)$ have a non-vanishing gradient, however, $f^{-1}(0)$ is not in the preimage of f, thus it is a regular surface (i.e., a sphere with radius 1).

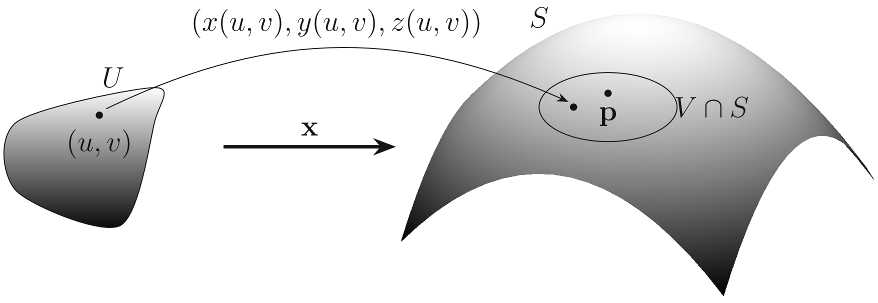


Figure 3.2: Illustration of a regular surface. The local parametrization $\mathbf{x}$ deforms the 2D set U into a 3D set $V \cap S$ around a point $\mathbf{p}$ on a regular surface S.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3.1.7 Proposition Let $S \subset \mathbb{R}^3$ be a regular surface and $\mathbf{p} \in S$ a point on the surface, then there exists a neighborhood V of $\mathbf{p}$ in S in such a way that V is the graph of a differentiable function in one of the forms:
$x = f(y, z)$, $y = f(x, z)$, $z = f(x, y)$.

Proposition 3.1.7 guarantees that in a neighborhood around each point on a regular surface S the surface can be expressed as the graph of a differentiable function.

The Tangent Plane and Curvatures of a Regular Surface

Consider a regular surface S and a point on the surface $\mathbf{p} \in S$, and a regular parametrized curve $\gamma : (-\epsilon, \epsilon) \to S$ with $\gamma(0) = \mathbf{p}$. Since γ is a regular parametrized curve the tangent line $\gamma'(0)$ exists. Let $\mathbf{x} : U \subset \mathbb{R}^2 \to S$ be a parametrization of S and let $\mathbf{q} \in U$. Then the set of all tangent lines of regular parametrized curves on the surface S passing through $\mathbf{p}$ span a 2D subspace. The plane $d\mathbf{x}_q$, that passes through $\mathbf{x}(\mathbf{q}) = \mathbf{p}$, does not depend on the parametrization and is called the *tangent plane* to S denoted by $(T_{\mathbf{p}}(S))$ at the point $\mathbf{p}$. The basis vectors that span $T_{\mathbf{p}}(S)$ are written as $\mathbf{x}_u$ and $\mathbf{x}_v$ and are determined by the parametrization. By choosing a parametrization $\mathbf{x}$, a unit normal vector in the point $\mathbf{p} \in \mathbf{x}(U)$ can be defined as follows

$$\vec{n}(\mathbf{p}) = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}(\mathbf{p}), \tag{3.1}$$

which is called the *normal vector* of the surface S in $\mathbf{p}$ (see Figure 3.3).

It is not always possible to continuously define a unique normal vector $\vec{n}(\mathbf{p})$ on every point of a regular surface, see, for example, the Möbius strip (see Figure 3.4).

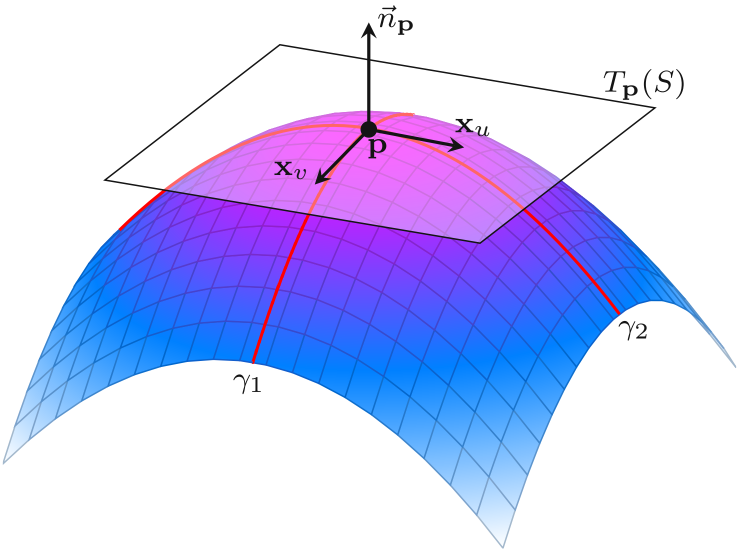


Figure 3.3: Illustration of the tangent plane $T_{\mathbf{p}}(S)$, the corresponding basis vectors $[\mathbf{x}_u, \mathbf{x}_v]$, and two parametrized curves γ_1, γ_2 in the point $\mathbf{p}$ on a regular surface.

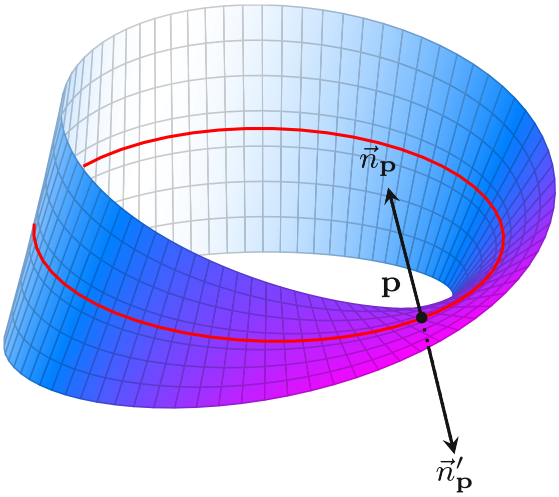


Figure 3.4: Illustration of a Möbius strip, a non-orientable surface. The Möbius strip is non-orientable since the choice of a normal vector in the point $\mathbf{p}$ is not unique. The normal vector $\vec{n}_{\mathbf{p}}$ can be traced along the red line until it ends up in the point $\mathbf{p}$ again, but then it points in the opposite direction (e.g., $\vec{n}'_{\mathbf{p}}$). This holds true for all points on the Möbius strip.

When a normal vector and a closed curve that traces over the Möbius strip is chosen, and the normal is moved along this curve it ends up in the same point, however, the normal now points in the opposite direction. This observation can also be described with the help of parametrizations. On each point of the Möbius strip there exist different parametrizations of the surface, these parametrizations describe the same tangent plane. However, the normal vectors defined by the parametrizations point in opposite directions (see Figure 3.4) and it is not possible to continuously change from one parametrization to the other.

For the following discussions it is essential that the discussed surfaces allow to continuously define a unique normal vector in each surface point. To that end a parametrization of a neighborhood of each point on a regular surface is fixed, the set of all these parametrizations is called a *family of coordinate neighborhood*. If this family of coordinate neighborhood describes a positive movement around each point of the surface it is called an *orientation* of the surface which is formally defined as follows:

3.1.8 Definition (Orientable Surface) A regular surface S is called *orientable* if there exists a family of coordinate neighborhoods such that for each point $\mathbf{p}$ that belongs to two neighborhoods of the family, the change of the parametrization has a positive Jacobian (i.e., the determinant of the Jacobian matrix).
Otherwise, the surface is called *non-orientable*.

Intuitively, Definition 3.1.8 expresses that on an orientable surface the change in the direction of the normal vector is continuous when switching between two neighboring parametrizations.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

If a regular surface S is defined by a differentiable function $f : U \subset \mathbb{R}^3 \rightarrow \mathbb{R}$ as follows $S = \{(x, y, z) \in \mathbb{R}^3 : f(x, y, z) = a\}$ and $f^{-1}(a) \neq 0$ then S is orientable.

3.1.9 Definition (Orientation of a Surface) Let S be a regular orientable surface then a differentiable mapping $\mathcal{n} : U \rightarrow \mathbb{R}^3$ of an open set $U \subset S$ that associates each point $\mathbf{p} \in U$ with the unit normal vector $\vec{n}(\mathbf{p})$ is called an *orientation* of S on U.

It should be mentioned that an orientation can locally be defined on every regular surface, however, orientability is a global property that concerns the entire surface.

To gain further insights into the geometry of a surface the orientation (i.e., $\mathcal{n}$) defined on the entire surface is further investigated.

3.1.10 Definition (Gauss Map) Let S be a regular surface with an orientation $\mathcal{n}$ defined on the entire surface S. Then $\mathcal{n}$ maps all normal vectors on S into the unit sphere and is called the *Gauss map*.

Figure 3.5 visualizes how the Gauss map operates on the hyperbolic paraboloid. Another easily visualized example of the Gauss map is the image of the torus which simply is the entire surface of the unit sphere.

The derivative of the Gauss map $d\mathcal{n}_{\mathbf{p}}$ in a point $\mathbf{p}$ maps the tangent plane $T_{\mathbf{p}}(S)$ into itself. Now consider a curve $\gamma(t)$ in S with $\gamma(0) = \mathbf{p}$, when the normal vector of the surface $\vec{n}_S(\mathbf{p})$ is restricted to the curve $\gamma(t)$ the map $d\mathcal{n}_{\mathbf{p}}(\gamma(0)) = \vec{n}'_S(\mathbf{p})$ results in a tangent vector in $T_{\mathbf{p}}(S)$. So $d\mathcal{n}_{\mathbf{p}}$ measures how fast $\vec{n}_S(\mathbf{p})$ pulls away from the surface in the curve $\gamma(0)$. This is a similar construction to the one discussed for regular parametrized curves with the difference that the curvature is measured by a linear map and not by the norm of a vector. The construction is independent of the chosen regular parametrized curve, so the following definition can be given:

3.1.11 Definition (Normal Curvature) Let S be an orientable surface, $\mathbf{p} \in S$ a point on the surface, γ a regular parametrized curve passing through $\mathbf{p}$ with curvature κ, and $\cos(\theta) = \vec{n}_S(\mathbf{p}) \cdot \vec{n}_\gamma(\mathbf{p})$. Then the number $k_{\vec{n}_\gamma(\mathbf{p})} = \kappa \cos(\theta)$ is called the *normal curvature* of $\gamma \subset S$ in $\mathbf{p}$.

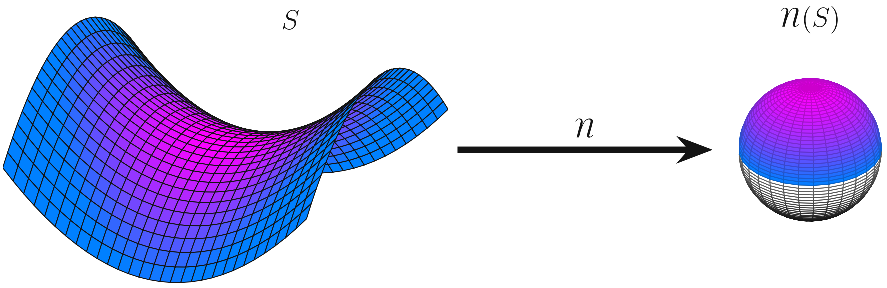


Figure 3.5: Illustration of the Gauss map of the hyperbolic paraboloid. All normal vectors on the surface are mapped into the unit sphere. The image of the Gauss map of the hyperbolic paraboloid is the hemisphere. The colors on both surfaces correspond to their respective normal vectors.

This definition results in a curvature value dependent on the chosen curve on the surface, which leads to the question if these curves can be classified in some way. To get further insights into this question the Gauss map $\mathpzc{n}$ of a surface S is further investigated. The negative differential $-d\mathpzc{n}$ of the Gauss map (sometimes referred to as the *Weingarten map* or *shape operator*) is a self-adjoint linear operator (II). Thus, its Eigenvectors are orthogonal, and the Eigenvalues are real [25]. The matrix II is also known as the *second fundamental form* of the surface S. Recall, that $-d\mathpzc{n}_{\mathbf{p}}$ measures the normal curvature for each regular parametrized curve passing through $\mathbf{p}$. Since the map $-d\mathpzc{n}_{\mathbf{p}}$ is a self-adjoint linear operator the maximum and minimum normal curvatures in each point can be calculated, which leads to the following definition:

3.1.12 Definition (Principal Curvatures) Let S be an orientable surface, and $\mathbf{p} \in S$ a point on the surface S. Then, the maximum normal curvature κ_1 and the minimum normal curvature κ_2 are called the *principal curvatures* of the surface S in the point $\mathbf{p}$.

The corresponding Eigenvectors to the principal curvatures are called the principal directions of S in $\mathbf{p}$. Since the map $-d\mathpzc{n}$ is a linear map the trace and the determinant of the map can be calculated.

3.1.13 Definition (Gaussian Curvature) Let S be an orientable surface, $\mathbf{p} \in S$ be a point on the surface, and $\mathpzc{n}_{\mathbf{p}}$ the Gauss map in $\mathbf{p}$, then

$$\det(-d\mathpzc{n}_{\mathbf{p}}) = K = \kappa_1 \kappa_2$$

is called the *Gaussian curvature* of S in the point $\mathbf{p}$.

3.1.14 Definition (Mean Curvature) Let S be an orientable surface, $\mathbf{p} \in S$ be a point on the surface, and $\mathpzc{n}_{\mathbf{p}}$ the Gauss map in $\mathbf{p}$, then

$$\operatorname{trace}\left(\frac{1}{2}(-d\mathpzc{n}_{\mathbf{p}})\right) = H = \frac{\kappa_1 + \kappa_2}{2}$$

is called the *mean curvature* of S in the point $\mathbf{p}$.

The mean and Gaussian curvature are used to define two special classes of surfaces. Surfaces with a constant Gaussian curvature of 0 are called *developable surfaces*, these surfaces can be "rolled" into the plane, or in other words folded out of paper. A cylinder is an example of a developable surface and the sphere is an example of a non-developable surface.

Surfaces with a constant mean curvature of 0 are called *minimal surfaces*, if these surfaces are bounded they minimize the Dirichlet energy in each point [65]. The plane is a trivial example of a minimal surface, where an example for a non-trivial minimal surface is the helicoid. Minimal surfaces play an important role in the remainder of this work.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Until this point the curvatures of a surface have been derived from the definition of the Gauss map without any explicit reference to a local coordinate system (i.e., a parametrization). For the calculation of the surface curvatures of discrete surfaces a description of the Gauss map in a local coordinate system $\{\mathbf{x}_u, \mathbf{x}_v\}$ is required and is given by [25]

$$d\mathcal{n} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} := -\underbrace{\begin{pmatrix} e & f \\ f & g \end{pmatrix}}_{\mathrm{II}}\underbrace{\begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1}}_{\mathrm{I}}, \tag{3.2}$$

with

$$\begin{array}{ll} E = \mathbf{x}_u \cdot \mathbf{x}_u & -f = \vec{n}_v \cdot \mathbf{x}_u = \vec{n}_u \cdot \mathbf{x}_v \\ F = \mathbf{x}_u \cdot \mathbf{x}_v & -e = \vec{n}_u \cdot \mathbf{x}_u \\ G = \mathbf{x}_v \cdot \mathbf{x}_v & -g = \vec{n}_v \cdot \mathbf{x}_v \end{array}, \tag{3.3}$$

I (see Equation 3.2) is also referred to as the *first fundamental form*. The mean and Gaussian curvature can thus be expressed in any basis as

$$K = \frac{\det(\mathrm{II})}{\det(\mathrm{I})}, \tag{3.4}$$

$$H = \frac{\operatorname{trace}(\mathrm{I}\,\operatorname{adj}(\mathrm{II}))}{2\det(\mathrm{I})} \tag{3.5}$$

where $\operatorname{adj}(M)$ stands for the adjugate matrix.

Some methods of calculating the curvatures of discretized surfaces further down in this chapter only calculate the mean and the Gaussian curvature. The principal curvatures can be calculated from the mean and Gaussian curvature by solving the following equation

$$\kappa_{1,2} = H \pm \sqrt{H^2 - K}. \tag{3.6}$$

3.2 Curvature Calculation for Point Clouds

When estimating the surface curvatures of a point cloud similar obstacles, such as the approximation of the surface normal discussed in Section 2.1.1, have to be overcome. Since there is no information about the spatial coherence of the points in a point cloud, a suitable neighborhood has to be estimated first. After a suitable neighborhood has been identified, recall Proposition 3.1.7, which states that in a local neighborhood a surface can be expressed by a differentiable function $f(x, y)$, which is also called the *height function* [66]. The goal now is to estimate $f(x, y)$ with the points in the previously estimated neighborhood. $f(x, y)$ can be expressed by a Taylor series of order n

$$f(x, y) = J_{\vec{B},n}(x, y) + \mathcal{O}(\|(x, y)\|^{n+1}),$$

$J_{\vec{B},n}(x, y)$ is called a jet of order n or an n-jet.

To obtain an n-jet of the height function defined by the points in a neighborhood around a point $\mathbf{p}$ it has to be approximated. The n-jet is approximated in the least square sense which leads to the following optimization problem

$$\min\left(\sum_{i=1}^{N}(J_{\vec{B},n}(x_i,y_i)-f(x_i,y_i))^2\right).$$

This optimization problem results in a vector $\vec{B}$ that holds the first n coefficients of the Taylor polynomial associated with the n-jet.

$$J_{\vec{B},n}=B_{0,0}+B_{1,0}x+B_{0,1}y+\frac{1}{2}(B_{2,0}x^2+2B_{1,1}xy+B_{0,2}y^2)+\dots. \tag{3.7}$$

To calculate the curvatures of a surface requires at least a 2-jet. The coefficients of the Taylor polynomial can then be used to calculate the Weingarten map (see Equation 3.2) with [66]

$$\begin{aligned} E &= 1+B_{1,0}^2 & -f &= \tfrac{2B_{0,2}}{\sqrt{1+B_{1,0}^2+B_{0,1}^2}} \\ F &= B_{1,0}B_{0,1} & -e &= \tfrac{2B_{1,1}}{\sqrt{1+B_{1,0}^2+B_{0,1}^2}} \\ G &= 1+B_{0,1}^2 & -g &= \tfrac{2B_{0,2}}{\sqrt{1+B_{1,0}^2+B_{0,1}^2}} \end{aligned}\ . \tag{3.8}$$

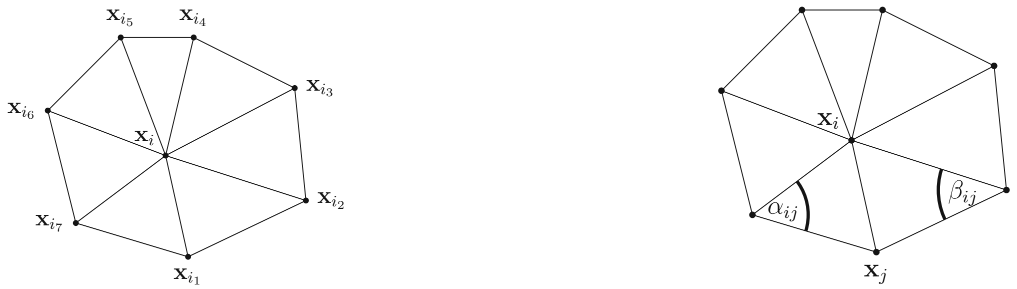
The here presented discussion on 3D surfaces can be similarly adapted for 2D surfaces [66].

3.3 Curvature Calculation for Surface Meshes

Estimating the curvatures of a 3D surface mesh requires less computational effort than for point clouds, since the neighborhood of a vertex and information about the geometry is already present in the discretization. The general idea for calculating the curvatures of a surface mesh is to associate an infinitesimal area around a vertex with the value of an operator, further referred to as *spatial averaging*. Two different operators are required to calculate the Gaussian and mean curvatures. When the area around a vertex converges towards zero the value of the quotient between these operators and the area converges towards the desired curvature value [67]. Thus, first an appropriate area around each vertex on a surface mesh has to be determined. An appropriate area around a given vertex is given by the so-called *Voronoi region* [67]. Furthermore, each Voronoi cell (i.e., the Voronoi area each triangle contributes to the Voronoi region) minimizes the error introduced by the spatial averaging and the discretization [68].

Before the calculation of the Voronoi region is presented another concept has to be introduced. The set of all vertices incident to a vertex $\mathbf{x}_i$ is referred to as the *1-ring neighborhood of* $\mathbf{x}_i$ ($N_1(\mathbf{x}_i)$) (see Figure 3.6a).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) All vertices in the 1-ring neighborhood $N_1(\mathbf{x}_i)$ of the vertex $\mathbf{x}_i$ of an exemplary triangulation.

(b) Voronoi region formula illustration.

Figure 3.6: Illustration of the neighborhood of a vertex.

Thus, the Voronoi region of a vertex $\mathbf{x}_i$ in a triangle mesh is calculated as follows

$$A_{\text{Voronoi}} = \frac{1}{8} \sum_{j \in N_1(\mathbf{x}_i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \|\mathbf{x}_i - \mathbf{x}_j\|^2, \tag{3.9}$$

see Figure 3.6b. This definition only holds for triangle meshes with non-obtuse triangles, which can in general not be guaranteed. The operators discussed further down in this section also result in valid curvature values for 1-ring neighborhoods which consist only of obtuse triangles. Therefore, the calculation of the area has to be adapted (see Algorithm 1) [67]. The algorithm checks each triangle in the 1-ring neighborhood, if a triangle is obtuse a fraction of its area is added to A_{mixed}, otherwise the area of the Voronoi cell is added. A_{mixed} denotes the surface area of the mixed region around a vertex, the set of points making up the mixed region is referred to as A_M.

Algorithm 1: Calculation of the mixed area of the 1-ring neighborhood of a vertex $\mathbf{x}_i$ (A_{mixed}).

```
input  : 1-ring neighborhood of the vertex $\mathbf{x}_i$
output: $A_{\text{mixed}}$

$A_{\text{mixed}} = 0$;
foreach Triangle $T$ in the 1-ring neighborhood of $\mathbf{x}_i$ do
    if $T$ is non-obtuse then
        // Calculate area of Voronoi cell see (3.9)
        $A_{\text{mixed}}$ += Voronoi region of $T$;
    else
        if the angle of $T$ at $\mathbf{x}_i$ is obtuse then
            $A_{\text{mixed}}$ += area($T$)/2;
        else
            $A_{\text{mixed}}$ += area($T$)/4;
```

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Mean Curvature

As previously hinted, surfaces with a constant mean curvature of 0 describe a special class of surfaces: minimal surfaces. Minimal surfaces minimize the surface area of a surface, which lead to an ample body of literature [69, 70]. One of the investigated operators is the *mean curvature normal operator* which establishes a direct relation between the mean curvature flow and the minimization of the surface area [71]

$$\lim_{\mathrm{diam}(A_{\mathbf{p}})\to 0} \frac{\nabla A_{\mathbf{p}}}{A_{\mathbf{p}}} = 2H_{\mathbf{p}}\vec{n}_{\mathbf{p}}, \tag{3.10}$$

where $\nabla A_{\mathbf{p}}$ stands for the gradient, and $A_{\mathbf{p}}$ for the area around the point $\mathbf{p}$. The mean curvature normal operator, also known as the *Laplace-Beltrami operator*, is defined as follows:

3.3.15 Definition (Mean Curvature Normal Operator) Let S be an orientable surface and $\mathbf{p}$ a point on S, further let $\vec{n}_{\mathbf{p}}$ be the normal vector and $H_{\mathbf{p}}$ the mean curvature in that point. Then the *mean curvature normal operator* is defined as

$$\mathbf{H}(\mathbf{p}) = 2H_{\mathbf{p}}\vec{n}_{\mathbf{p}}.$$

The integral of the Laplace-Beltrami operator over the area A_{mixed} in a vertex $\mathbf{x}_i$ on a surface mesh can be discretized as follows [67]

$$\iint_{A_M} \mathbf{H}(\mathbf{x}_i)dA = \frac{1}{2}\sum_{j\in N_1(\mathbf{x}_i)} (\cot\alpha_{ij} + \cot\beta_{ij})(\mathbf{x}_i - \mathbf{x}_j).$$

After inserting the discretized value of the Laplace-Beltrami operator and A_{mixed} into Equation 3.10. The mean curvature of a vertex $\mathbf{x}_i$ on a surface mesh can be expressed as

$$H_{\mathbf{x}_i} = \frac{\|\sum_{j\in N_1(\mathbf{x}_i)}(\cot\alpha_{ij} + \cot\beta_{ij})(\mathbf{x}_i - \mathbf{x}_j)\|}{4A_{\text{mixed}}}. \tag{3.11}$$

Gaussian Curvature

The Gaussian curvature, like the mean curvature, can also be expressed as a fraction of an infinitesimal area around a point as follows [72]

$$\lim_{\mathrm{diam}(A_{\mathbf{p}})\to 0} \frac{A^G_{\mathbf{p}}}{A_{\mathbf{p}}} = K.$$

To elaborate the meaning of $A^G_{\mathbf{p}}$, recall the definition of the Gauss map n, it maps all normal vectors from a surface into the unit sphere. Consider a small surface area around a point $\mathbf{p}$ on an orientable surface, then the image of that area under n describes a small area on the surface of the unit sphere (e.g., $A^G_{\mathbf{p}}$).

Instead of directly calculating the integral over the image of the Gauss map the Gauss-Bonnet theorem is used [67]

$$\iint_{A_M} K\,dA + \sum_{i=0}^{k} \varepsilon_i = 2\pi.$$

This is a simplified version of the theorem since the surface area of the sphere is investigated which has a geodesic curvature of 0 [25]. The ε_i stand for the external angles of the boundary of the Voronoi region. For Voronoi regions it is easy to see that the angle at the vertex $\mathbf{x}_i$ is $\theta_i = \varepsilon_i$, because both edges of the Voronoi cell are perpendicular to the edges of the triangle and, therefore, $\theta_i + \alpha_i = \pi$ (see Figure 3.7). Thus, the Gaussian curvature of a vertex $\mathbf{x}_i$ on a surface mesh can be expressed as

$$K_{\mathbf{x}_i} = \frac{\left(2\pi - \sum_{j=1}^{f} \theta_j\right)}{A_{\text{mixed}}}, \tag{3.12}$$

where f stands for the number of all triangles in $N(\mathbf{x}_i)$.

3.4 Curvature Calculation for Implicit Surfaces

The computationally least expensive way to estimate the surface curvatures of a discrete surface considered in this work is the estimation for implicit surfaces, since, derivatives can be approximated directly from the discretization.

In this section three different approaches of calculating the surface curvatures of an implicitly defined surface are presented: The general formula for implicit surfaces, mean curvature estimation by calculating the variation of the normal vector, and mean curvature estimation through the shape operator. These three approaches are further analyzed in Chapter 5.

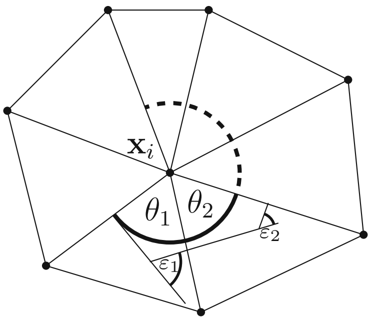


Figure 3.7: Angles in the 1-ring neighborhood required for the calculation of the Gaussian curvature.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3.4.1 General Formula

The goal is to express Equations 3.4 and 3.5 with the derivatives of an implicit function ϕ. This is achieved in the following way, the determinants and traces are transformed such that they are expressed with respect to a local parametrized $\mathbf{x}$ and the normal vector $\vec{n}$ defined by the chosen parametrization [73]

$$K = \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot (\vec{n}_u \times \vec{n}_v)}{\|\mathbf{x}_u \times \mathbf{x}_v\|^2}, \tag{3.13}$$

$$H = \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot ((\mathbf{x}_v \times \vec{n}_u) - (\mathbf{x}_u \times \vec{n}_v))}{2\|\mathbf{x}_u \times \mathbf{x}_v\|^2}. \tag{3.14}$$

Afterwards, the terms of the above equations are transformed such that they are expressed as the derivatives of the implicit function. This is achieved with the help of matrix identities, which transform the above expressions into [73]

$$K = \frac{\nabla\phi \,\mathrm{adj}(\mathrm{Hess}(\phi))\, \nabla\phi^T}{\|\nabla\phi\|^4}, \tag{3.15}$$

$$H = \frac{\nabla\phi \,\mathrm{Hess}(\phi)\, \nabla\phi^T - \|\nabla\phi\|^2\, \mathrm{trace}(\mathrm{Hess}(\phi))}{2\|\nabla\phi\|^3}, \tag{3.16}$$

where $\mathrm{Hess}(\phi)$ stands for the Hessian matrix of ϕ. Furthermore, the mean curvature of an implicit surface can be expressed as

$$H = \nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|}, \tag{3.17}$$

which can be seen by expanding Equation 3.16 and Equation 3.17 and comparing the resulting terms.

Now that an expression for the curvatures of an implicit surface is available it has to be discussed how these formulas can be used on discretized implicit surfaces. Considering a discretized implicit surface as defined in Section 2.3 (e.g., a level-set function), the derivatives of a level-set function can be approximated by central differences as follows

$$D_x(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}, \tag{3.18}$$

$$D_{xx}(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{\Delta x^2}, \tag{3.19}$$

$$D_{xy}(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j+1,k} - \phi_{i-1,j+1,k} - \phi_{i+1,j-1,k} + \phi_{i-1,j-1,k}}{4\Delta x \Delta y}, \tag{3.20}$$

where $\phi_{i,j,k}$ stands for the ϕ-value of the level-set function at the grid point (i, j, k), if the subscript indices are omitted it is assumed that they are (i, j, k).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek, Your knowledge hub

Using these approximations and an expanded form of Equation 3.16, the discretized mean curvature of a level-set function can be approximated as follows

$$H = \frac{\begin{array}{l} D_x(\phi)^2(D_{yy}(\phi) + D_{zz}(\phi)) \\ \quad +D_y(\phi)^2(D_{xx}(\phi) + D_{zz}(\phi)) \\ \quad\quad +D_z(\phi)^2(D_{xx}(\phi) + D_{yy}(\phi)) \\ \quad\quad\quad -2D_x(\phi)D_y(\phi)D_{xy}(\phi) \\ \quad\quad\quad\quad -2D_y(\phi)D_z(\phi)D_{yz}(\phi) \\ \quad\quad\quad\quad\quad -2D_z(\phi)D_x(\phi)D_{xz}(\phi) \end{array}}{2\|\nabla\phi\|^3}. \tag{3.21}$$

This method of calculating the mean curvature will be referred to as the *General Formula* method. The Gaussian curvature can be calculated similarly by using an expanded form of Equation 3.15

$$K = \frac{\begin{array}{l} D_x(\phi)^2(D_{yy}(\phi)D_{zz}(\phi) - D_{yz}(\phi)^2) \\ \quad +D_y(\phi)^2(D_{xx}(\phi)D_{zz}(\phi) - D_{xz}(\phi)^2) \\ \quad\quad +D_z(\phi)^2(D_{xx}(\phi)D_{yy}(\phi) - D_{xy}(\phi)^2) \\ \quad\quad\quad +2D_x(\phi)D_y(\phi)(D_{xz}(\phi)D_{yz}(\phi) - D_{xy}(\phi)D_{zz}(\phi)) \\ \quad\quad\quad\quad +2D_x(\phi)D_z(\phi)(D_{xy}(\phi)D_{yz}(\phi) - D_{xz}(\phi)D_{yy}(\phi)) \\ \quad\quad\quad\quad\quad +2D_y(\phi)D_z(\phi)(D_{xy}(\phi)D_{xz}(\phi) - D_{yz}(\phi)D_{xx}(\phi)) \end{array}}{(D_x(\phi)^2 + D_y(\phi)^2 + D_z(\phi)^2)^2}. \tag{3.22}$$

Calculating the Gaussian curvature requires the same derivatives as calculating the mean curvature.

The finite difference approximations used for calculating the surface curvatures require a certain amount of grid points to be calculated. The set of all grid points required for the calculation of a set of finite differences is referred to as a *finite difference stencil*, or just stencil ($\eta(\mathbf{x})$). A finite difference stencil is always considered with respect to the central grid point $\mathbf{x} = (i, j, k)$. Two finite difference stencils are of particular importance for this work, the 7 point *star stencil* ($\eta_S(\mathbf{x})$), and the 19 point *plane stencil* ($\eta_P(\mathbf{x})$). Figure 3.8 shows an illustration of these stencils. Calculating the surface curvature with the *General Formula* requires a plane stencil.

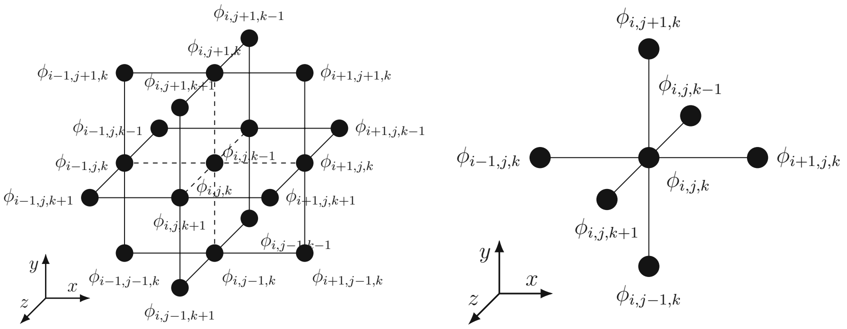
Lastly it should be mentioned that the maximal curvature a level-set function can describe is bound by $\pm 1/\Delta x$ [15], since the smallest circle radius which can be represented with a given resolution is Δx.

3.4.2 Shape Operator

Recalling Equation 3.17, the formula states that the mean curvature of an implicit surface can be expressed as the gradient of the normal vector. This expression can be expanded and leads to the following expression [75]

$$\nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|} = \nabla\vec{n} = -\frac{1}{\|\nabla\phi\|}(I - \vec{n}\vec{n}^T)\mathrm{Hess}(\phi), \tag{3.23}$$

where I stands for the Identity matrix.



(a) Plane stencil ($\eta_P((i,j,k))$) **(b)** Star stencil ($\eta_S((i,j,k))$)

Figure 3.8: 3D finite difference stencils used for curvature calculation. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Assuming that the level-set function fulfills the Eikonal equation (see Equation 2.8) with $F(\mathbf{x}) = 1$, then the term $1/\|\phi\|$ can be ignored. The matrix $(I - \vec{n}\vec{n}^T)$ projects onto the tangent plane of the implicit surface. Restricting $\nabla\vec{n}$ to the tangent plane shows that it is symmetric [75]. Thus, there exists an orthonormal basis $[\vec{p_1}, \vec{p_2}]$ in $\mathbb{R}^2$ of the tangent plane, this basis can be expanded to an orthonormal basis in $\mathbb{R}^3$ as follows $[\vec{p_1}, \vec{p_2}, \vec{n}]$. When the gradient of the normal vector is expressed in this basis it results in the following matrix

$$\nabla\vec{n} = \begin{pmatrix} \kappa_1 & 0 & \sigma_1 \\ 0 & \kappa_2 & \sigma_2 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.24}$$

Here, κ_1 and κ_2 stand for the principal curvatures with the principal directions $\vec{p_1}$ and $\vec{p_2}$, σ_1 and σ_2 stand for the so-called *flow line curvatures* [75]. So the mean curvature of the surface can be expressed as $\text{trace}(\nabla\vec{n})/2$. Furthermore, the trace of a matrix is independent of the chosen basis, thus the mean curvature of a level-set function ϕ with $\|\nabla\phi\| = 1$ can be expressed as

$$H = \frac{\text{trace}(\text{Hess}(\phi_{i,j,k}))}{2} = \frac{D_{xx}(\phi_{i,j,k}) + D_{yy}(\phi_{i,j,k}) + D_{zz}(\phi_{i,j,k})}{2}. \tag{3.25}$$

This method is referred to as the *Shape Operator* method. In Section 4.1.2 a method is discussed that describes how a level-set function can be constructed that fulfills $\|\nabla\phi\| = 1$. Calculating the mean curvature of the zero level-set using the *Shape Operator* method requires a star stencil η_S. It should be mentioned that the quality of the mean curvature calculated with this method on a discretized surface is highly dependent on the quality of the discretization, which is discussed further in Section 5.1.3.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

3.4.3 Variation of Normal

The third method calculates the mean curvature of the level-set function by again using Equation 3.17 which is approximated through the Euler-Lagrange derivative [76, 77]. In addition to the first-order central differences, this approximation requires the first-order forward and backward differences which can be approximated by

$$D_x^+(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x}, \tag{3.26}$$

$$D_x^-(\phi_{i,j,k}) \approx \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\Delta x}. \tag{3.27}$$

In turn, the mean curvature can be approximated as [77]

$$\begin{aligned} H = \nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|} \approx \frac{1}{2\Delta x}\Bigg(&\left(\frac{D_x^+(\phi)}{\|\vec{g}_x^+\|} - \frac{D_x^-(\phi)}{\|\vec{g}_x^-\|}\right) + \\ &\left(\frac{D_y^+(\phi)}{\|\vec{g}_y^+\|} - \frac{D_y^-(\phi)}{\|\vec{g}_y^-\|}\right) + \left(\frac{D_z^+(\phi)}{\|\vec{g}_z^+\|} - \frac{D_z^-(\phi)}{\|\vec{g}_z^-\|}\right)\Bigg), \end{aligned} \tag{3.28}$$

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

$$\begin{aligned} \vec{g}_x^\pm := (D_x^\pm(\phi_{i,j,k}), &\frac{1}{2}(D_y(\phi_{i\,j,k}) + D_y(\phi_{i\pm1,j,k})), \\ &\frac{1}{2}(D_z(\phi_{i\pm1,j,k}) + D_z(\phi_{i,j,k}))), \end{aligned} \tag{3.29}$$

$$\begin{aligned} \vec{g}_y^\pm := (D_y^\pm(\phi_{i,j,k}), &\frac{1}{2}(D_x(\phi_{i,j\pm1,k}) + D_x(\phi_{i,j,k})), \\ &\frac{1}{2}(D_z(\phi_{i,j\pm1,k}) + D_z(\phi_{i,j,k}))), \end{aligned} \tag{3.30}$$

$$\begin{aligned} \vec{g}_z^\pm := (D_z^\pm(\phi_{i,j,k})), &\frac{1}{2}(D_x(\phi_{i,j,k\pm1}) + D_x(\phi_{i,j,k})), \\ &\frac{1}{2}(D_y(\phi_{i,j,k\pm1}) + D_y(\phi_{i,j,k})). \end{aligned} \tag{3.31}$$

To calculate all the required finite differences a plane stencil η_P is required, this method is referred to as the *Variation of Normal* method. This method only calculates the mean curvature, thus, the Gaussian curvature has to be estimated separately by using Equation 3.22.

3.4.4 Curvature of 2D Implicit Surfaces (Curves)

An analogous construction to the one presented in Section 3.4.1 can be done for regular curves (i.e., 2D surfaces) [73]. Recall that regular curves only have one type of curvature which can be approximated for a 2D level-set function with the formula

$$\kappa = \frac{D_y(\phi)^2 D_{xx}(\phi) - 2D_x(\phi)D_y(\phi)D_{xy}(\phi) + D_x(\phi)^2 D_{yy}(\phi)}{\|\nabla\phi\|^3}. \tag{3.32}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3.5 Intuitive Approach to Surface Classification

When considering the construction of the normal curvature presented in Section 3.1.2 one can see that the normal curvatures are defined through the curvature of a parametrized curve and the cosine of the angle between the surface normal and the normal of the considered parametrized curve. It is tempting to drop the formal mathematics behind curvature calculation and try to directly use the angle between the normal vectors of surface points in a neighborhood on a discretized surface for surface classification. However, this approach has several drawbacks which are discussed in the remainder of this section. The most striking drawback is, that this approach can only estimate angles with points of the discretization without any knowledge of the underlying parametrization of the surface.

Point Clouds

Consider a point cloud and a neighborhood $M_i(\mathbf{x}_i)$ of a point $\mathbf{x}_i$. The relative spatial position of the points in the neighborhood with respect to the investigated point $\mathbf{x}_i$ is not known. So, it is possible that two points in the neighborhood describe the same curve on the surface and that one point is farther away from $\mathbf{x}_i$. In this scenario it is not known which of the two points should be chosen for the classification of the surface which can lead to wrong classifications. Therefore, this approach is not suited for point clouds.

Surface Meshes

The triangles of a surface mesh are flat, thus, they have a mean and Gaussian curvature of 0. A similar argument can be used for edges, since the surface can only bend in one direction over an edge. Consequently, using the angle between normal vectors can of course only be used on vertices on a surface mesh, but there is no straightforward way to calculate the surface normal of a vertex. Which again, as with point clouds, makes this approach unsuited for surface meshes.

Level-Set Functions (Implicit Surfaces)

Finally, considering level-set function, some merit can be found in applying the normal vector angle approach. All points on a regular grid have the same distance from each other, so one could check if the points that lie in a *box stencil* (see Figure 3.9) are surface points (i.e., the level-set function changes sign from the point $\mathbf{x}_i$ to this neighboring point). However, depending on the position of the zero level-set more points in the stencil are required, i.e., 21 in 2D and 81 in 3D, which is a lot more than calculating the derivatives for curvature calculation. Further drawbacks of this method on level-set functions are discussed in Section 5.1.4.

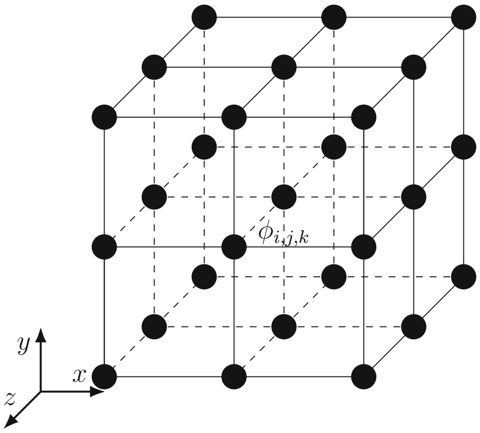


Figure 3.9: 3D box stencil ($\eta_B((i,j,k))$) required for normal based feature detection on implicit surfaces.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 4

Topography Simulation and Simulation Platforms

The fabrication of a semiconductor device consists of many process steps. In some of them materials are stacked on top of the wafer or other materials (i.e., deposited). Others remove parts of the materials on the wafer (i.e., etch selectively) [78]. Furthermore, there are processing steps that do not change the topography of the semiconductor device, yet they change its electrical properties (i.e., ion implantation). When such fabrication processes are simulated they are abstracted into a *process model*. These process models describe how reactants (e.g., molecules or ions) generated inside a reactor interact with the wafer surface.

Fabricating semiconductor devices typically involves a plethora of materials. Consequently, the fabrication process models tend to be material dependent and each material on the simulated wafer has to be uniquely represented. The movement of the materials when they are affected by the process model is described by the level-set equation (see Equation 4.2), a time-dependent PDE.

This chapter gives an overview of the numerical methods used during topography simulations. These methods are subsequently consolidated into the general simulation flow of a topography simulation. Additionally, the computing systems and software tools used in this thesis are introduced.

4.1 Evolution of Surfaces (Advection)

Consider a given process model which describes the movement of the wafer surface in each surface point. Furthermore, assume that the process model has already been evaluated. Consequently, a so-called *velocity* value, based on physical inputs, is produced by the process model for every point on the wafer surface, which expresses the movement of the surface points in time. The set of all velocity values is referred to as the *velocity field* ($V(\mathbf{x}, t)$). Depending on the sign of the velocity field an etching (i.e., positive sign), or a deposition process (i.e., negative sign) is modeled. The objective is to move all surface points according to this velocity field.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

This can be accomplished by solving the ordinary differential equation [15]

$$\frac{d\mathbf{x}}{dt} = V(\mathbf{x}, t) \tag{4.1}$$

for each point on the surface. This description of the surface's movement is referred to as the *Lagrangian* formulation. The Lagrangian formulation for the evolution of the surface requires an explicit representation of the surface, since it describes the movement of individual points on the surface. In addition to the already discussed problems with the deformation of explicit surface representations (see Section 2.2), the Lagrangian formulation also has problems with instabilities and requires topological repairs of the surface [15].

A more stable description which avoids the aforementioned issues of the surface when it evolves in time is an implicit representation of the surface (see Section 2.3). An implicit surface $\phi(\mathbf{x})$ with dimension n is embedded into an $n+1$ dimensional space to describe its evolution in time. The evolution of the surface is accomplished by solving the so called *level-set equation* [54]

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + V(\mathbf{x}, t) \|\nabla \phi(\mathbf{x}, t)\| = 0, \tag{4.2}$$

which is also referred to in literature as the convection or advection equation. This method of tracking the evolution of the zero level-set (i.e., the surface) is commonly referred to as an *Eulerian* formulation. When using an Eulerian formulation the velocity values have to be known in the entire domain instead of only on the surface. Section 4.1.3 discusses how the velocity values are expanded from the surface into the domain Ω.

4.1.1 Solving the Level-Set Equation

Substituting $\mathscr{H}(\mathbf{x}, \phi(\mathbf{x}, t), \nabla \phi(\mathbf{x}, t), t)$ for $V(\mathbf{x}, t)\|\nabla \phi(\mathbf{x}, t)\|$ in Equation 4.2 yields

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + \mathscr{H}(\mathbf{x}, \phi(\mathbf{x}, t), \nabla \phi(\mathbf{x}, t), t) = 0, \tag{4.3}$$

which shows that the level-set equation belongs to the class of *Hamilton-Jacobi* equations, $\mathscr{H}$ is also referred to as the Hamiltonian. The arguments to V and ϕ are omitted for the remainder of this section. Hamilton-Jacobi equations are utilized in classical mechanics to describe mechanical systems, thus there has been substantial research into numerically solving these types of equations [79, 80, 81, 82, 83]. The solution schemes used in this work are discussed further down in this section.

TU Bibliothek, Your knowledge hub. TU WIEN. Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar. The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Upwind Schemes

Assume that V is known in the entire domain, a first-order accurate scheme to solve Equation 4.2 is the so-called *forward Euler* method [84]

$$\frac{\phi(t_{n+1}) - \phi(t_n)}{\partial t} + \mathscr{H}(\mathbf{x}, \phi(t_n), \nabla\phi(t_n), t_n) = 0, \tag{4.4}$$

where t_n is a point in time and $t_{n+1} = t_n + \Delta t$ is the point after one time step. However, simply using this scheme to solve the level-set equation will not result in accurate solutions, as a consequence of numerical instabilities [15, 54]. To improve the solution of the level-set equation Engquist and Osher proposed a modified scheme [81]

$$\mathscr{H} \approx \max(V, 0)\mathscr{D}^{+}(\phi) + \min(V, 0)\mathscr{D}^{-}(\phi), \tag{4.5}$$

with the first-order accurate approximation $\mathscr{D}^{+}(\phi) = \sqrt{(\sum_{l=1}^{n} D_l^{+}(\phi))^2}$ and $\mathscr{D}^{-}(\phi) = \sqrt{(\sum_{l=1}^{n} D_l^{-}(\phi))^2}$. This scheme is referred to as the *Engquist Osher* scheme. The idea is to look at the values of the velocity field and the characteristics that flow out of the zero level-set. This information is subsequently used to bias the one-sided finite difference approximation dependent on the direction in which the information flows [81]. The drawback of this method is that it only works for convex Hamiltonians (i.e., if the Hessian matrix of the Hamiltonian $\frac{\partial^n \mathscr{H}}{\partial x_1 \ldots \partial x_n}$ is positive semi-definite) [8].

Lax-Friedrichs Schemes

In topography simulations the Hamiltonian is not necessarily always convex [85]. Thus, a different scheme is required, to calculate the discretization of the Hamiltonian. The *Lax-Friedrichs* scheme uses first-order central finite difference approximations (see Equation 3.18) to calculate the Hamiltonian. Using central differences to solve Equation 4.2 leads to numerical instabilities, like strong oscillation in the solution, that have to be handled [78]. The Lax-Friedrichs scheme is defined as follows [82]

$$\mathscr{H} \approx V\sqrt{\sum_{l=1}^{n} D_l(\phi)^2} - \underbrace{\sum_{l=1}^{n} \alpha_l \left(\frac{D_l^{+}(\phi) - D_l^{-}(\phi)}{2}\right)}_{DT}, \tag{4.6}$$

with the *dissipation coefficients* α_l and the *dissipation term* DT. The dissipation coefficients have to fulfill

$$\max_{x_i \in \mathbf{x}} \left|\frac{\partial \mathscr{H}}{\partial q_i}\right| \leq \alpha_l, \tag{4.7}$$

with $q_i = \frac{\partial \phi}{\partial x_i}$ for a stable propagation of the solution [82]. The choice of the dissipation coefficients has to be handled with great care.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

When the dissipation term is chosen to be too large it over-smooths the solution of Equation 4.2. This over-smoothing then leads to inconsistently rounded corners with respect to the process model. On the other hand, if it is chosen too small the solution is not stable [79, 86]. To address these problems several strategies have been introduced [13, 15]. In this work a so-called *Stencil Lax-Friedrichs* schemes is used [87]. A Stencil Lax-Friedrich scheme only considers a subset of the domain close to a given point in which the numerical Hamiltonian is solved. Thus, the dissipation coefficients are only calculated in a local domain around the point, which prevents over-smoothing of the solution. Toifl et al. proposed the following calculation of the dissipation terms for a grid point $\mathbf{x}$ in a 9 grid points (2D) and 27 grid points (3D) stencil (e.g., $\eta_B(\mathbf{x})$) for selective epitaxy and wet etching [87]

$$\alpha_l = \max_{\mathbf{p}\in\eta(\mathbf{x})} \left| V n_i + \frac{\partial V}{\partial n_i}\left(\frac{\phi_j^2+\phi_k^2}{\|\nabla\phi\|^2}\right) - \frac{\partial V}{\partial n_j}\left(\frac{\phi_j\phi_i}{\|\nabla\phi\|^2}\right) - \frac{\partial V}{\partial n_k}\left(\frac{\phi_k\phi_i}{\|\nabla\phi\|^2}\right)\right|, \tag{4.8}$$

where n_i stands for the i-th coordinate of the normal vector and $\frac{\partial V}{\partial n_l}$ is defined as follows

$$\frac{\partial V}{\partial n_i} \approx \frac{V(n_i+\Delta n, n_j, n_k, t) - V(n_i-\Delta n, n_j, n_k, t)}{2\Delta n, t}, \tag{4.9}$$

with $\Delta n = \epsilon^{\frac{1}{3}} V$ where ϵ stands for the floating point accuracy.

Courant-Friedrichs-Lewy Condition

When numerically solving time-dependent PDEs (e.g., Equation 4.2) the *stability* of the solution has to be considered when it evolves in time. A solution to a PDE is called stable if small errors in the numerical approximation are not magnified as the solution moves forward in time [88]. This can be achieved by enforcing the Courant-Friedrichs-Lewy (CFL) condition [89], the CFL condition for the forward Euler method can be expressed as [88]

$$\Delta t < \frac{\Delta x}{\max(|V|)}. \tag{4.10}$$

The maximal time step Δt is thus calculated as follows

$$\Delta t = \alpha \frac{\Delta x}{\max(|V|)}, \tag{4.11}$$

with $\alpha \in (0,1)$. In the context of the level-set method the CFL condition restricts how far the zero level-set can move with respect to the grid resolution Δx on a regular grid, during a single time step.

4.1.2 Reconstructing the Signed Distance Function (Re-Distancing)

During simulations using the level-set method the zero level-set is moved through the simulation domain.

Thus, the description of the zero level-set (i.e., the surface described by the level-set function) changes in every time step. The quality of the ϕ-values away from the zero level-set deteriorate with each time step (i.e., the distances between the level-sets of the level-set function move closer together or spread out) [90]. To reduce numerical errors in the propagation of the surface, the signed distance function has to be reconstructed. To that end, depending on the normalization used to describe the level-set function, different strategies must be applied.

Euclidean Normalization

The level-set function is constructed in such a way that it fulfills the Eikonal equation (Equation 2.8 revisited)

$$\begin{aligned} \|\nabla\phi(\mathbf{x})\| &= F(\mathbf{x}),\ \forall\mathbf{x}\in\Omega \\ \phi(\mathbf{x}) &= G(\mathbf{x}),\ \forall\mathbf{x}\in S, \end{aligned}$$

when the Euclidean normalization is used.

To construct a signed distance function originating from the zero level-set S with a departure time of 0 and a constant speed of 1 the following Re-Distancing PDE has to be solved

$$\begin{aligned} \|\nabla\phi(\mathbf{x})\| &= 1,\ \forall\mathbf{x}\in\Omega \\ \phi(\mathbf{x}) &= 0,\ \forall\mathbf{x}\in S. \end{aligned} \tag{4.12}$$

This method only calculates positive signed distance values (i.e., values on the outside of the volume Ω^+). Thus, to calculate the signed distance values on the inside of the volume (i.e., Ω^-) the calculated distance values on the inside have to be multiplied by -1.

Constructing the Signed Distance Function (Fast Marching Method)

The previously described scheme to construct a signed distance function is achieved by using the *fast marching method* (FMM) [91]. The FMM is a numerical method that solves the Eikonal equation on a grid. The numerical solution requires first-order forward and backwards finite difference approximations (see Equation 3.26 and 3.27). Thus, the solution of the Eikonal equation can be discretized as follows [92]

$$\begin{bmatrix} \max(-D_x^+(\phi_{i,j,k}), D_x^-(\phi_{i,j,k}), 0)^2 & + \\ \max(-D_y^+(\phi_{i,j,k}), D_y^-(\phi_{i,j,k}), 0)^2 & + \\ \max(-D_z^+(\phi_{i,j,k}), D_z^-(\phi_{i,j,k}), 0)^2 & \end{bmatrix} = \frac{1}{F_{i,j,k}^2}. \tag{4.13}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

When the following substitutions are used

$$\begin{aligned}
V &= \phi_{i,j,k}, \\
V_1 &= \min(\phi_{i-1,j,k}, \phi_{i+1,j,k}), \\
V_2 &= \min(\phi_{i,j-1,k}, \phi_{i,j+1,k}), \\
V_3 &= \min(\phi_{i,j,k-1}, \phi_{i,j,k+1}),
\end{aligned}$$

Equation 4.13 can be simplified to

$$\max\left(\frac{V-V_1}{\Delta x},0\right)^2 + \max\left(\frac{V-V_2}{\Delta y},0\right)^2 + \max\left(\frac{V-V_3}{\Delta z},0\right)^2 = \frac{1}{F_{i,j,k}^2}. \tag{4.14}$$

Furthermore, since it is assumed that the speed of the waves emerging from the front (e.g., the zero level-set) is positive, $V - V_i$ must be greater than 0. Therefore, the problem can further be simplified to the quadratic equation

$$\left(\frac{V-V_1}{\Delta x}\right)^2 + \left(\frac{V-V_2}{\Delta y}\right)^2 + \left(\frac{V-V_3}{\Delta z}\right)^2 = \frac{1}{F_{i,j,k}^2}. \tag{4.15}$$

When the Eikonal equation is solved on a regular grid, the n dimensional solution to the quadratic equation can be expressed as follows

$$V = \frac{1}{n}\sum_{l=1}^{n} V_l + \frac{1}{n}\sqrt{\left(\sum_{l=1}^{n} V_l\right)^2 - n\left(\sum_{l=1}^{n} V_l^2 - \frac{\Delta x^2}{F_{i,j,k}^2}\right)}. \tag{4.16}$$

It is possible that during this calculation not all grid points needed to solve Equation 4.13 have a finite value or that Equation 4.16 does not have a real solution. In this case a so-called *lower dimensional update* has to be performed. To that end the largest value $V_{\max} = \max_{l\in 1\ldots n}(V_l)$ is removed and Equation 4.16 is solved with one less value. Note that this process always produces a valid solution for the Eikonal equation since a one-dimensional update is $\min_{l\in 1\ldots n}(V_l) + \frac{\Delta x}{F_{i,j,k}}$, which always results in a finite value. The grid points associated with ϕ-values V_i used to solve Equation 2.8 are also referred to as the *upwind neighbors*.

During the FMM a grid point can have one of three states: *accepted*, *calculated*, and *far*. The FMM starts by assigning all grid points neighboring the zero level-set S the state of *accepted* and all other points the state *far*. Afterwards, all grid points adjacent to the zero level-set S are collected, and their distance is calculated using Equation 4.16. The grid points are marked as *calculated* and stored in a minimum heap. Next, the head of the minimum heap is removed, the respective grid point is marked as *accepted* and new values for the grid points in a star stencil (i.e., η_S) that are not yet *accepted* are calculated with Equation 4.16. This process continues until the heap is empty and all points in the domain have a distance value.

The FMM is not the only method for numerically solving Equation 2.8 on a grid [93]. However, it is to this day a widely used method, particularly due its numerical stability [94].

Manhattan Normalization

In the sparse field method the level-set function is considered in layers. The layer $\mathcal{L}_0$ represents the zero level-set and the layers $\mathcal{L}_j$ with $j \in \mathbb{Z}\backslash\{0\}$ represent the iso-contours further away from the zero level-set. Calculating the next layers $\mathcal{L}_{i+1}$ relative to an already known layer $\mathcal{L}_i$ is achieved by updating all points in the new layer as follows [55, 58]

$$\phi^{\mathcal{L}_i}(\mathbf{x}) = \begin{cases} \min_{\mathbf{y}\in\eta_S(\mathbf{x})\cap\mathcal{L}_{i-1}} \phi^{\mathcal{L}_{i-1}}(\mathbf{y}) + 1 & \text{if } i > 0 \\ \max_{\mathbf{y}\in\eta_S(\mathbf{x})\cap\mathcal{L}_{i+1}} \phi^{\mathcal{L}_{i+1}}(\mathbf{y}) - 1 & \text{if } i < 0 \end{cases}, \tag{4.17}$$

where $\eta_S(\mathbf{x})$ describes a star stencil in the point $\mathbf{x}$ and $\phi^{\mathcal{L}_i}(\mathbf{x})$ the ϕ-value of the level-set function on the layer $\mathcal{L}_i$ in the grid point $\mathbf{x}$.

Constructing the signed distance function with this approach is computationally much more efficient than using the FMM. Since, the computational overhead of managing the heap and solving the quadratic Equation 4.16 is replaced by determining the minimum or maximum in $\eta_S(\mathbf{x})$ and a summation.

4.1.3 Velocity extension

In Section 4.1 methods to numerically solve Equation 4.2 have been discussed. Until now, it has been assumed that the values of the velocity field $V(\mathbf{x}, t)$ are known in the entire domain. However, the process models used to describe the propagation of the zero level-set in topography simulations can only be meaningfully defined directly on the surface [78, 95, 96, 97]. In order to solve Equation 4.2, the values of $V(\mathbf{x}, t)$ have to be known in the entire domain. Dependent on the level-set normalization different strategies have to be employed to propagate the surface information as far into the domain as necessary.

Euclidean Normalization

Equation 2.8 describes the waves that propagate outwards from the zero level-set in normal direction with a given speed of $\frac{1}{F(x)}$. So, any information defined on the zero level-set can be propagated into the entire domain using this method. This propagation of information (e.g., velocity values) is described by the so-called velocity extension equation [95]

$$\begin{aligned} \nabla\Phi(\vec{x}) \cdot \nabla V(\vec{x}) &= 0 & \vec{x} &\in \Omega \\ V(\vec{x}) &= V_S(\vec{x}) & \vec{x} &\in S, \end{aligned} \tag{4.18}$$

where V_S stands for the calculated velocity values at the cross points of the zero level-set [95].

The solution of Equation 4.18 can be calculated using the FMM. Yet, during this simulation step (see Figure 4.7) it can be assumed that the FMM has already been performed to construct the signed distance function [15]. Thus, the direction of the information propagating outwards, orthogonal to the zero level-set, is already known.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

This available information can be reused during the velocity extension by constructing the heap in such a way that the new velocity values are calculated only once for each grid point. This is possible since all the upwind neighbors for each grid point of the level-set function have already been calculated. Furthermore, this process can be parallelized by considering the problem in the context of graph theory [98].

Manhattan Normalization

The previously described extension process is not necessary for Manhattan normalized level-set functions. In this case, these level-set functions directly describe the cross points of the zero level-set (i.e., $\mathcal{L}_0$) on which the models are defined. Nevertheless, to achieve the movement of the layer $\mathcal{L}_0$ a rebuilding step is required [55]. This rebuilding step is in essence the same as the construction of the signed distance function discussed in Section 4.1.2 (see Equation 4.17) combined with the pruning of larger values (i.e., $\phi(\mathbf{x}) \geq 0.5$; see discussion in Section 2.3.2). The velocity values are calculated for all surface points (i.e., $\mathcal{L}_0$) and stored in this layer. Consequently, the layers $\mathcal{L}_1$ and $\mathcal{L}_{-1}$ are calculated. Note that only these two layers are needed since the maximal distance the zero level-set can move during one time step is bound by the CFL condition. To reconstruct the layer $\mathcal{L}_0$ all values of the signed distance function that are larger than 0.5 are removed (since in the here considered case the ϕ-values are normalized between -0.5 and 0.5; see Section 2.3.2). This results in the zero level-set (i.e., layer $\mathcal{L}_0$) after one time step.

4.2 Surface Flux Calculation

During the fabrication of a semiconductor device the wafer is placed inside a reactor that produces reactants that interact with the wafer surface. These interactions can remove (i.e., etching) add material (i.e., deposition) from/to the wafer surface [78, 96, 97, 99, 100, 101]. In a topography simulation the above discussed surface interactions are captured by the process model. The two main parts that have to be considered in the process model are:

- *Reactor Scale Transport Model*: The chamber in which the reactants for the process are created is modeled. This model has to take into account the gases or plasma put into the chamber in addition to the temperature, pressure, chemical reactions, and the geometry of the chamber [102]. The output is the concentration, energy distribution, and the velocities of different reactant species.
- *Feature Scale Transport Model*: The transport through the feature geometry until a reactant species interacts with the wafer surface is modeled. Furthermore, the actual interaction of the different reactant species with the wafer surface is described.

In topography simulations the results of the reactor scale model are used as inputs for the feature scale model.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

This is achieved through the virtual so-called *source plane* ($\mathscr{P}$), which allows to model the distribution of the reactant species created during the reactor scale simulation. The source plane separates the reactor scale from the feature scale. The reactant (source) fluxes and their energy and angular distributions are fixed on this plane. Feature scale transport is modeled by combining several reactants of the same species into *particles*. Furthermore, a particle may also represent the aggregation of multiple species with similar chemical behavior. This is a necessary abstraction since there could be an order of 10^{20} reactants present in the reactor, easily overcoming available computing resources and thus not allowing for practically relevant simulations. Additionally, it is assumed that the surface stays constant during the modeling of the particle transport and that the particles are distributed according to the reactor scale transport model in the source plane. During this approach a surface coverage (Θ) of the reactants on the wafer surface is calculated, which is subsequently used to determine the velocity field $V(\mathbf{x}, t)$ according to the process model. The surface coverage is calculated by estimating the surface fluxes $\Gamma(\mathbf{x})$ for the particles according to the process model. The surface flux a point $\mathbf{x}$ on the surface receives is modeled by the following equation [103]

$$\begin{aligned}\Gamma(\boldsymbol{x}) &= \int_{\Omega} \Gamma_{in}(\boldsymbol{x}, \boldsymbol{\omega}_{d\Omega}) d\Omega \\ &= \int_{\mathscr{P}_{vis}} \frac{\boldsymbol{\omega}_{\boldsymbol{x}_{src}} \cdot \boldsymbol{n}_{\boldsymbol{x}}}{\|\boldsymbol{x} - \boldsymbol{x}_{src}\|^2} \left(\Gamma_{src}(\boldsymbol{x}_{src}, -\boldsymbol{\omega}_{\boldsymbol{x}_{src}})\right) d\boldsymbol{x}_{src} \\ &+ \int_{\mathscr{S}_{vis}} \frac{\boldsymbol{\omega}_{\boldsymbol{x}_{re}} \cdot \boldsymbol{n}_{\boldsymbol{x}}}{\|\boldsymbol{x} - \boldsymbol{x}_{re}\|^2} \left(\Gamma_{re}(\boldsymbol{x}_{re}, -\boldsymbol{\omega}_{\boldsymbol{x}_{re}})\right) d\boldsymbol{x}_{re},\end{aligned} \tag{4.19}$$

where $\mathscr{P}_{vis}$ stands for the visible part of the source plane and $\mathscr{S}_{vis}$ stands for the visible part of the surface. Equation 4.19 describes the flux on the entire surface by integrating over the amount of flux each point on the surface receives (see Figure 4.1).

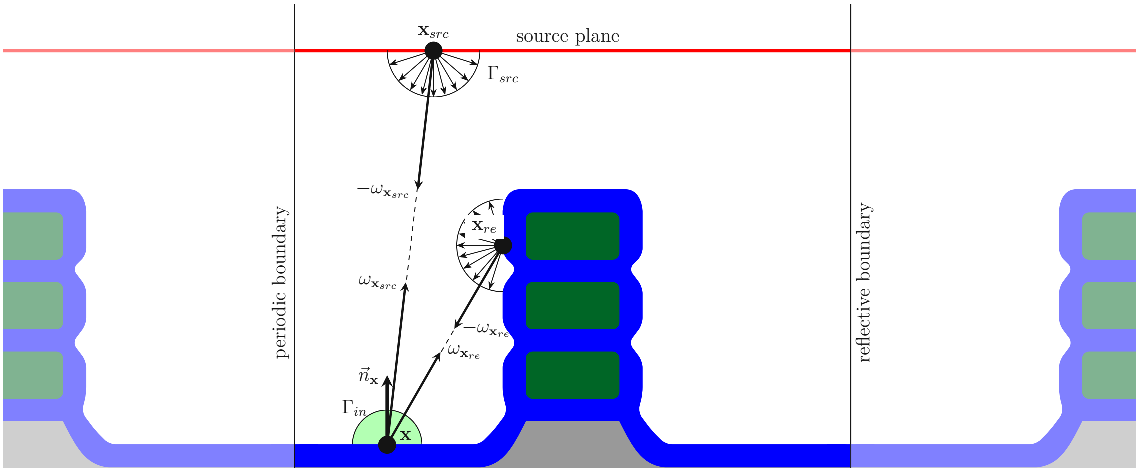


Figure 4.1: Illustration of the surface flux calculation. A point $\mathbf{x}$ on the surface is affected by two types of fluxes. The flux it receives from the source Γ_{src} and the flux that is reflected from the geometry Γ_{re}.

A point on the surface $\mathbf{x}$ receives fluxes from two sources, first, the direct flux received from the source Γ_{src} and second, the flux from reflections of particles in the domain Γ_{re}. The amount of flux a point on the surface receives is affected by how much of the source plane is visible $\boldsymbol{x}_{src}$ and how many other points on the surface that reflect particles are visible $\boldsymbol{x}_{re}$.

Since not the entire wafer is simulated some particles will leave the simulation domain. To avoid loosing the information of these particles so-called boundary conditions are employed. There are two kinds of boundary conditions: *periodic* and *reflective*. Periodic boundary conditions check where a particle leaves the domain and re-emit this particle on the opposite side of the domain, with the same trajectory. Reflective boundary conditions re-emit the particles that hit the boundary back into the domain at the point of impact.

For computational reasons, it is not feasible to calculate the surface flux directly from Equation 4.19, thus, the surface flux has to be estimated. The following sections discuss different approaches to estimate the surface flux (i.e., Equation 4.19): constant, bottom-up, and top-down [101].

4.2.1 Constant Approach

The straightforward way to approximate the surface flux on the wafer surface is to assume that it is constant. In this case it is assumed that each point on the surface receives the same amount of flux from $\mathscr{P}$. However, the simulated structure still has to be checked for eventual voids that are not exposed to $\mathscr{P}$ [104].

This simplistic surface flux approach is valid for some cases. For example in anisotropic wet etching the surface is exposed to enough reactant that the assumption of constant flux holds [105, 106]. However, this assumption does not hold in general and more sophisticated approaches for calculating the surface flux are required (e.g., for processes where a flux distribution is observed on the wafer surface) [101].

4.2.2 Bottom-Up Approach

The constant flux approach does not take shadowing and other geometry-dependent effects into consideration, e.g., points on the surface that receive fewer particles due to parts of the geometry blocking a direct path to the source plane $\mathscr{P}$ (see Figure 4.2). To alleviate this problem, a bottom-up approach is used. In a bottom-up approach each point on the wafer surface (i.e., bottom) is considered, and it is calculated how much of the source plane $\mathscr{P}$ is visible (i.e., up) from this point. Computationally the performance of this approach can be improved by adaptively sampling the hemisphere and only calculating the flux on certain surface elements [107]. These calculations can be performed directly on the level-set function $\phi(\mathbf{x}, t)$. This approach lends itself to model processes in which the reactants interact with the surface with little to no re-sputtering [6, 108].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

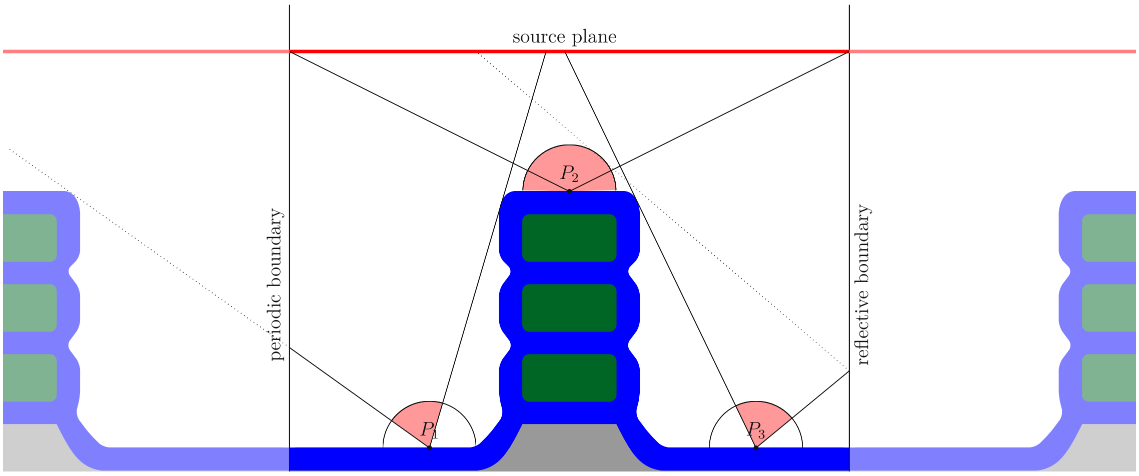


Figure 4.2: Illustration of the bottom up flux calculation approach. Not all surface points receive the same amount of flux. The flux each of the illustrated surface point receives is indicated by the red cones.

4.2.3 Top-Down Approach (Monte Carlo Ray Tracing)

The bottom-up approach does not take the reflection of individual particles from the geometry back into the domain into consideration. Although it is possible to model these reflections with a bottom up approach, it would require multiple iterations of bottom-up flux calculations on the entire surface, which would thus be unfeasible since it would require a lot of computational resources. To efficiently model the reflections of particles from the surface back into the domain, so-called ray tracing is utilized, which is a Monte Carlo based strategy to model the surface flux [103, 109]. During a ray tracing simulation, the source plane $\mathscr{P}$ is split into equally sized areas. The to be emitted particles are grouped together into packages which subsequently are emitted into the domain with a direction $\vec{t}$ given by the distribution of the particle source. The particles are traced from the source plane (i.e., top) until they interact with the wafer surface (i.e., down). When a particle interacts with the wafer surface a part of its flux's payload is absorbed by the surface and, depending on the remaining flux payload, is subsequently re-emitted into the domain. The amount of particles that are absorbed by a surface element from all particles are summed up and result in the flux at the respective surface element. Figure 4.3 shows an illustration of this process.

To efficiently compute the top-down ray tracing an explicit representation of the geometry of the wafer surface (e.g., a surface mesh or a point cloud) is preferable [110]. Furthermore, since ray tracing is a Monte Carlo process, numerical noise is introduced into the resulting flux, values which does not have a physical meaning. The magnitude of the numerical noise can be reduced by increasing the number of simulated particles. However, the numerical noise can never be entirely removed, and the more particles are simulated, the more computational resources are required.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek, WIEN Your knowledge hub

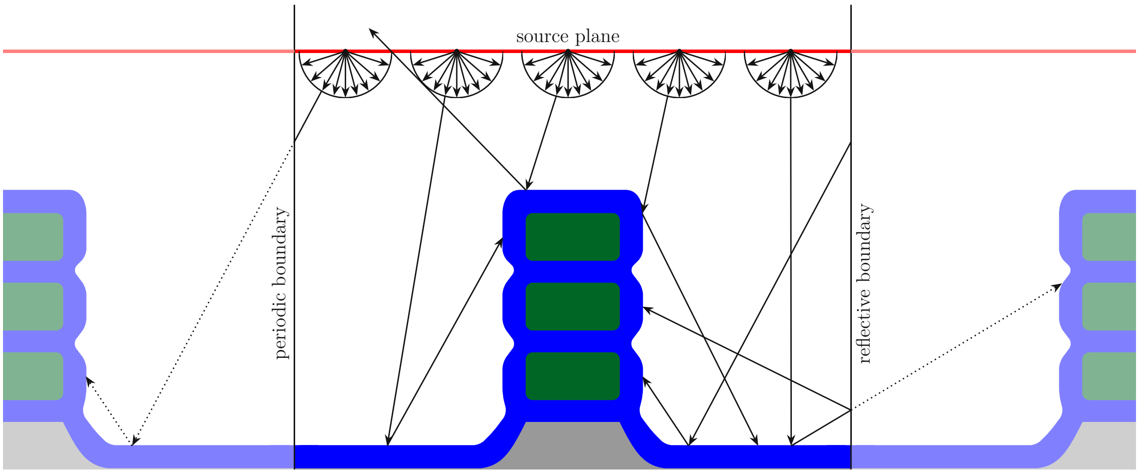


Figure 4.3: Illustration of the top-down flux calculation approach. In this approach, a particle can be re-emitted from the surface and thus simulate specular effects.

4.3 Multi-Material Simulations

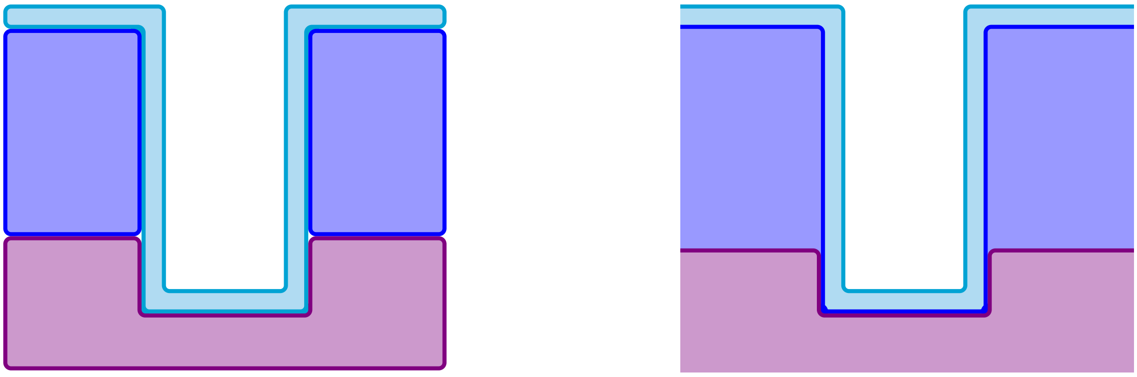
A typical topography simulation consists of several materials that are stacked on top of a wafer. To model different interactions of the process model with different materials each of these materials has to be represented by its own level-set function. The most natural approach would be to define a level-set function that envelopes each material (see Figure 4.4a).

However, this approach has some disadvantages concerning the interfaces between materials. Figure 4.5 illustrates the problems that can occur with such a description of the material layers. In Figure 4.5a an overlap of the two material layers is shown. These non-physical overlaps can form due to small numerical errors in the discretization of the level-set function. The same problem can also lead to non-physical voids shown in Figure 4.5b. A special kind of void is shown in Figure 4.5c (a triple junction). This void occurs because of the discretization of the level-set function, which leads to rounded corners. Reducing the grid resolution also reduces the rounding effect.

Most of the research on circumventing the formation of the previously discussed voids or overlaps originates from the study of multiphase flow [36, 111, 112, 113]. The strategies employed during such simulations are, on the one hand, to artificially keep the level-set functions together by changing the velocities of the velocity field. On the other hand, Boolean operations are employed to prevent the level-set functions from overlapping.

In multi-material topography simulations consisting of M materials usually $M-1$ level-set functions are required, since the vacuum or gas above the simulated device is usually not represented explicitly. The most common approach to handle multiple materials in topography simulations is the so-called additive or wrapping approach [114].

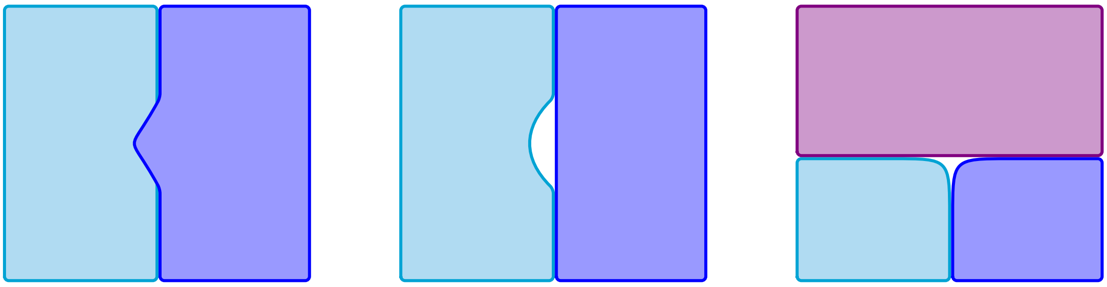
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) Level-set functions representing materials.

(b) Additive level-set functions (wrapping approach).

Figure 4.4: Illustration of two strategies of representing three material layers with level-set functions.



(a) Overlap

(b) Void

(c) Triple Juncture

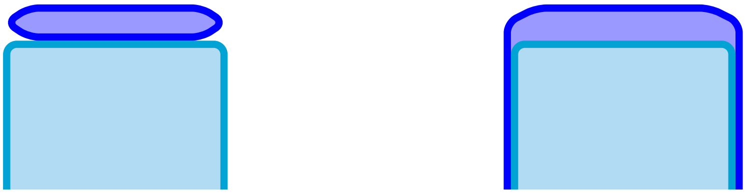
Figure 4.5: Illustration of problems that occur when enveloping material layers with level-set functions.

In the additive approach, each level-set function ϕ_i describing a material M_i is designed in such a way that it represents the union of all underlying materials $M_j \neq \emptyset$

$$M_i = \bigcup_{j=1}^{i-1} M_j \text{ where } M_j = \phi_j. \tag{4.20}$$

Figure 4.4b shows an illustration of this approach. The additive approach fixes another problem that often occurs in topography simulations (see Figure 4.6): When a material layer is stacked on top of another material, the rounding that occurs at the edges of the material interfaces leads to the formation of non-physical voids. Figure 4.6b illustrates how this problem is resolved by using the wrapping approach.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) Without the additive approach non-physical voids occur at the border of the materials.

(b) Using the additive approach avoids the formation of these non-physical voids.

Figure 4.6: Illustration of a thin material layer on top of another material layer.

4.4 Application of Surface Representations in Topography Simulations

During a topography simulation it can be advantageous, or in some cases even essential, to utilize different surface representations. As discussed in Section 4.1, an implicit surface representation is preferred during a topography simulation for the evolution of the surface. An implicit surface representation intrinsically handles topographical changes of the surface, such as the merging of two fronts, due to material merging which often occures during topography simulations [17]. In Chapter 7 a strategy for simulating etching processes with Boolean operations is discussed where again an implicit surface representation is advantageous (see Section 2.3.3). Furthermore, it is easier to perform CSG on implicit surfaces, so the generation of initial geometries is handled with implicit surfaces.

On the other hand, some flux calculation methods require an explicit surface representation (e.g., Monte Carlo ray tracing) [18]. Moreover, the visualization of surfaces, especially 3D surfaces, is much simpler with an explicit surface representation than with an implicit surface representation. Therefore, for flux calculations and the visualization of the surface the implicit surface is converted into an explicit surface as discussed in Section 2.4.

4.5 Topography Simulation Workflow

The effects of the process models on different materials on the wafer surface during a topography simulation is governed by the level-set equation (see Equation 4.2). In a discrete setting, a time-dependent PDE like the level-set equation is solved by calculating small time-steps that allow for a stable propagation of the solution according to the CFL condition [88], and is from heron referred to as level-set method. A simulation based on the level-set method is discretized in time as well as space.

An example of a topography simulation workflow based on the level-set method is shown in Figure 4.7, the stimulation steps this thesis focuses on are highlighted in red.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

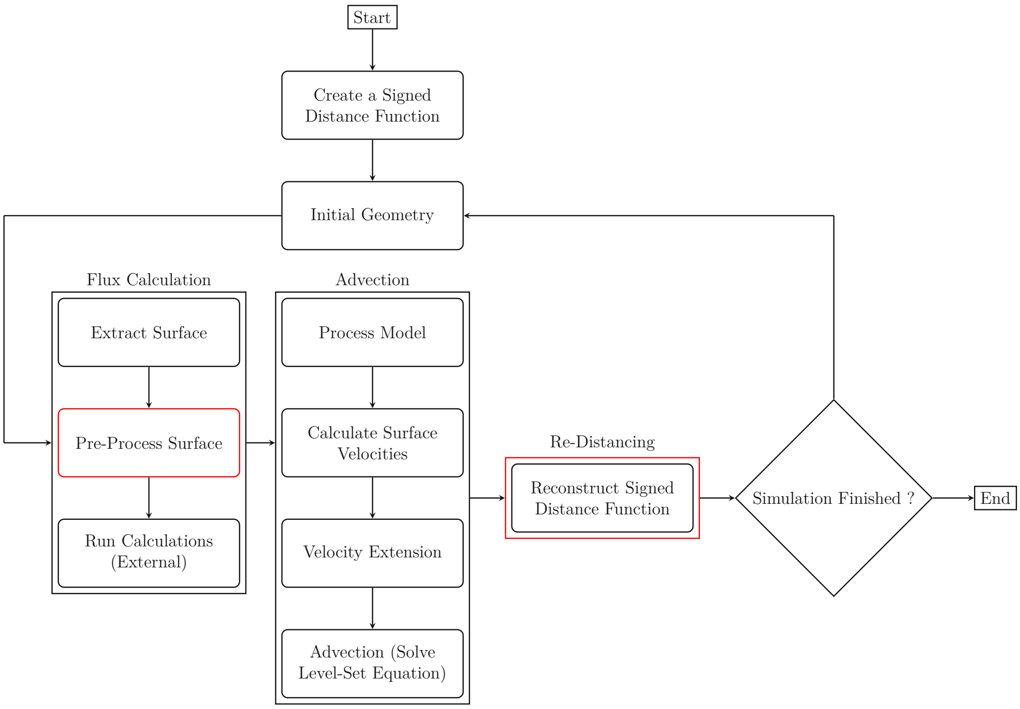

Start
Create a Signed Distance Function
Initial Geometry
Flux Calculation
Extract Surface
Pre-Process Surface
Run Calculations (External)
Advection
Process Model
Calculate Surface Velocities
Velocity Extension
Advection (Solve Level-Set Equation)
Re-Distancing
Reconstruct Signed Distance Function
Simulation Finished ?
End

Figure 4.7: Flowchart describing the typical topography simulation workflow of a simulation based on the level-set method. The steps marked in red are discussed in this work.

The general flow of a simulation is independent of the chosen level-set normalization. The simulation starts with the initialization of the geometry. The initial geometry usually originates from a previous simulation step or is created by CSG using Boolean operations of simple level-set functions (e.g., planes and boxes). After the initialization the main time-loop starts, and repeats the following three steps until the process model concludes.

Flux Calculation

Dependent on the process model used during the simulation certain pre-processing steps have to be performed (e.g., surface flux calculations). The surface flux obtained form this pre-processing step is then used during the evaluation of the process model. The extraction of the surface has been discussed in Section 2.4, a pre-processing step to improve the performance of Monte Carlo ray-tracing based flux calculation is discussed in Chapter 8. This step is not necessary for process models where a constant flux can be assumed.

Advection

The advection step is the core part of a level-set based topography simulation. During the advection step the process model is evaluated by taking into account the values generated by the flux calculation. The level-set equation is solved, and the velocities calculated by the process model are extended into the domain to propagate the surface. In a level-set framework based on the Manhattan normalization the velocity extension step is not required.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Re-Distancing

This step is required to prevent the propagation of numerical errors which emerge during the advection step. In Chapter 6 this step is further discussed to allow for simulations utilizing hierarchical grids.

4.6 Computational Hardware

The computational performance results discussed in the following chapters are computed on commercially available computer systems. Therefore, the performance of the to be presented experiments is bound by the limitations of these computer systems. To get a more precise understanding of these limitations the relevant concepts of modern computer systems are discussed in this section.

Modern microprocessor are schematically composed of three parts, see Figure 4.8 [115]. The *central processing unit* (CPU) handles all computations and interactions with data stored in the main memory, the main memory holds all information required for a program to run, and input/output allows for interactions between the program and the user. A *core* is the combination of the control and arithmetic logic unit. Modern CPUs consist of several cores on a single chip, which allows for the parallel execution of instructions. Furthermore, modern CPUs offer simultaneous multithreading and thereby offering for each *physical* core two *logical cores*, further increasing parallel computing capabilities. In turn, compute clusters are composed of *nodes*, each offering at least one CPU and potentially also additional accelerators, such as general purpose graphics processing units.

The speed at which the CPU can fetch information from the main memory is the primary limiting factor for performance also known as the *von Neumann bottleneck*. To partially overcome this issue so-called *caches* have been introduced to the CPU chip. Caches are small memories with very fast read and write speeds. They store small amounts of information and act as a buffer between CPU and main memory. Figure 4.9 shows an illustration of a modern computer system, which, conceptually, also represents one form of the previously mentioned node.

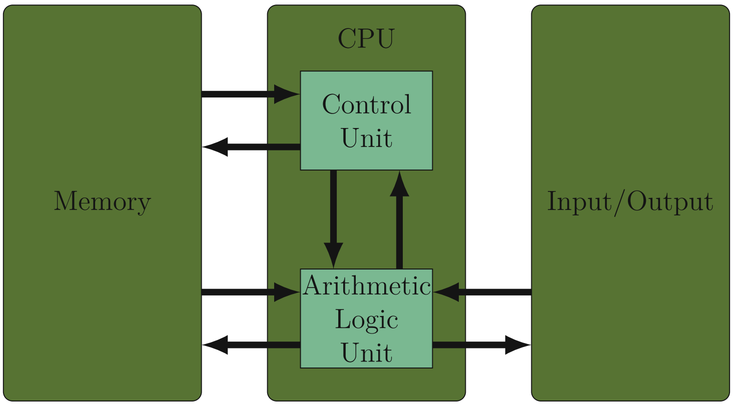


Figure 4.8: Illustration of the basic theoretical parts of a digital computer and their interactions.

This illustration shows an extended version of the basic principle described by Alan Turing in the 1930s [115]. The use of different cache levels is a historical development, since CPU clock speed (i.e., the amount of instruction the CPU can process during a certain time interval) grew much faster than the memory bandwidth (i.e., the speed at which data can be moved from the main memory to the CPU). This discrepancy is also commonly referred to as the DRAM gap [116].

4.6.1 Caches

As can be seen in Figure 4.9 there are different kinds of caches, usually the closer a cache is to the CPU register the faster it is [115]. Caches have different responsibilities, the *instruction cache* (i.e., L1 I) stores the order of instructions the CPU has to perform. In the *data cache* (i.e., L1 D) the data is stored on which the CPU wants to perform its instructions. If the data is not present in the cache the data has to be fetched from a higher level cache (e.g., L2 or L3), in the worst case it has to be fetched from the main memory. Some caches are exclusive to one core (e.g., L1 and L2), others are shared between all cores (e.g., L3).

As discussed in the previous section when the CPU requests data from the memory, it is first checked if the data resides in the cache. If the requested data is present in a cache it is called a *cache hit*, otherwise it is called a *cache miss* and the data has to be fetched from the main memory.

Cache memory is built with SRAM cells, SRAM cells take up a huge amount of physical space on a CPU. Furthermore, the fabrication of SRAM cells is more expensive than DRAM thus, it is not feasible to create huge caches [117].

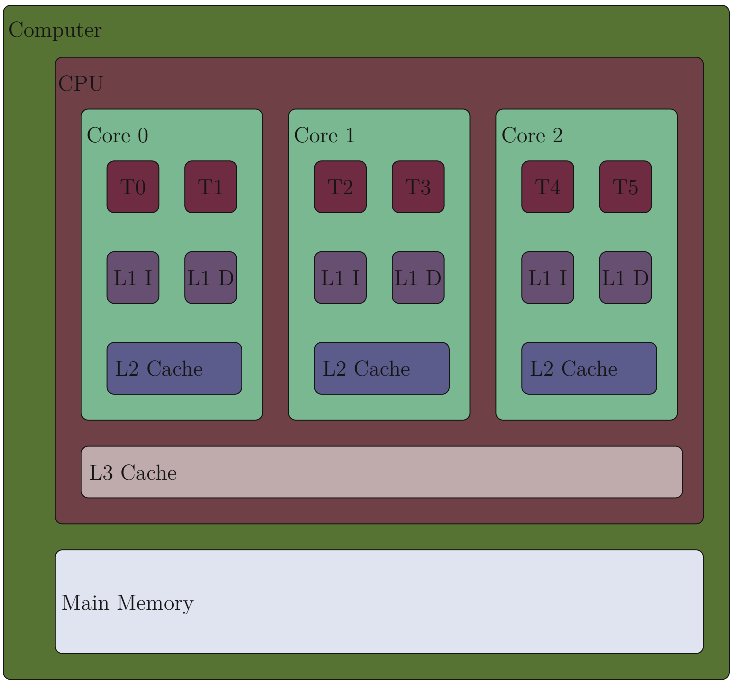


Figure 4.9: Illustration of a modern computer system. It is made up of a single CPU with three cores. Three different cache levels and the main memory.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

TU Bibliothek, WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

In a scientific framework the geometries that are simulated are large (i.e., large in the sense that the geometries do not fit into the cache of a CPU). Thus, much of the run-time required for a topography simulation is dedicated to the fetching of data from memory or higher cache levels. When the primary effort in executing a program consists of memory access and not calculation speed the problem is called *memory bound*.

4.6.2 Parallelization

In the mid 2000s CPUs hit the so-called *heat barrier*, which describes the effect that an increase in clock speed yields a disproportional amount of heat that has to be dissipated [115]. Thus, different strategies had to be developed to increase the performance of CPUs. Since Moore's law is still observable the size of semiconductor devices is still shrinking which frees up space on a wafer and ultimately the individual chips. To utilize this additional space vendors started introducing additional cores to the CPU. However, the speedup that can be expected by adding n cores to a CPU is in general smaller than n. This can have several reasons, ranging from limited memory bandwidth over non-optimal cache use up to load balancing issues (i.e., when a problem cannot efficiently be split into n parts).

The efficiency of a parallel program can be measured by putting the serial (s) and the parallel (p) parts into relation [115]

$$s + p = 1. \tag{4.21}$$

Let's first assume that there is one worker (e.g., a core) that works on a task (e.g., a program) the time it takes to finish the task can be expressed as

$$T_s = s + p. \tag{4.22}$$

Now assume that in the perfect scenario the run-time of a parallel task T is reduced to $\frac{T}{n}$ when using n workers. As previously mentioned there are limitations to the speedup a parallel program running on n cores can achieve. The ideal achievable speedup trough parallelization (also known as *strong scaling*) can be expressed as

$$T_p(n) = s + \frac{p}{n}. \tag{4.23}$$

Conversely, in case of *weak scaling* the number of processes and the problem size is increased, leading to a constant workload per process.

To define a measurement for the speedup of a program when it is parallelized a metric for the performance (i.e., the work that is done during the run-time of a program) is required. The performance of a serial program can be defined as [115]

$$P^s = \frac{s + p}{T_s} = 1, \tag{4.24}$$

and the performance of a parallel program as

$$P^p = \frac{s + p}{T_p(n)} = \frac{1}{s + \frac{1-s}{n}}. \tag{4.25}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Combining these two performance metrics leads to the speedup

$$S = \frac{P^p}{P^s} = \frac{1}{s + \frac{1-s}{n}} \tag{4.26}$$

of a parallel program. Equation 4.26 is also known as *Amdahl's Law*, which describes an upper limit of speedup for a problem of fixed size [118]. When $n \to \infty$ it is easily seen that the speedup that can be achieved trough parallelization is bound by the run-time of the serial parts of the program. Furthermore, there is *Gustafson's law* which estimates the parallel speedup of weak scaling programs [119].

4.6.3 Benchmark Systems

To evaluate the run-times of the algorithms presented in this work the used *benchmarking systems* are introduced in this section. The Benchmarking systems are a single node from the *Vienna Scientific Cluster* (VSC). The VSC is a collaboration of several Austrian universities that provides supercomputer resources and corresponding services to their users [120]. Furthermore, an industrial computer system (ICS) and a workstation (Workstation 1) are used. Table 4.1 summarizes the key metrics of the three used benchmark systems.

4.7 Software Tools

The research results presented in this work have been produced by the use of several open- and closed-source software tools. In this section all used tools are introduced and a brief description of the relevant features is given.

Victory Process

Victory Process is commercial process TCAD simulation tool developed by *Silvaco* [121]. It features several etching, deposition and oxidation models for semiconductor fabrication simulations as well as an interface to implement user specific process models.

	Workstation 1	Workstation 2	VSC4	ICS
Frequency (GHz)	4.4	4.6	3.1	2.8
Sockets	1	1	2	2
Cores per CPU	4	12	24	10
Logical cores per CPU	8	24	48	20
L1i cache	32 KByte	384 KByte	32 KByte	32 KByte
L1d cache	32 KByte	384 KByte	32 KByte	32 KByte
L2 cache	256 KByte	6 MB	1024 KByte	256 KByte
L3 cache	8 MByte	64 MB	33 MByte	26 MByte
Main memory	32 GByte	32 GByte	96 GByte	226 GByte

Table 4.1: Benchmarking systems used to run simulations in this work.

The modeling of the process models is based on the level-set method. Victory Process is written in C++, parallelization is achieved with pThreads and OpenMP. The results presented in Chapter 6 and Chapter 7 where produced with Victory Process. Victory Process uses a hierarchical grid data structure to store the level-set functions representing the different material layers in a topography simulation.

ViennaLS

ViennaLS is an open source level-set library which can be used for process TCAD simulations developed at the *Institute for Microelectronics TU Wien* [13]. It is developed in C++, uses OpenMP for parallelization and is accessible as a Python library. The results presented in Chapter 5 where produced with ViennaLS. ViennaLS uses a Hierarchical Run-Length Encoding (HRLE) data structure to store the level-set function.

Visualization Toolkit (VTK)

The Visualization Toolkit (VTK) is an open source collection of algorithms to manipulate and visualize scientific data [122]. It is developed in C++, however, it is available on several other platforms like Java or Python. In this work, VTK is used to store and visualize data. Furthermore, it is used to calculate the Hausdorff distance between two meshes in Chapter 8.

Computational Geometry Algorithms Library (CGAL)

The Computational Geometry Algorithms Library (CGAL) is a high performance library for geometric algorithms developed in C++ [123]. The surface mesh simplification algorithm presented in Chapter 8 is based on the implementation of the edge-collapse algorithm of CGAL. CGAL uses a *half-edge* data structure to store meshes. It is a high performance data structure that allows for fast access to neighboring vertices, edges faces and an efficient method to iterate over the unstructured mesh data sets.

Vienna Mesh

Vienna Mesh is a meshing framework developed at the *Institute for Microelectronics TU Wien* [124]. It is developed in C++ and uses a modular approach to combine multiple external mesh generation and adaptation tools (e.g., CGAL or VTK). The surface simplification algorithm presented in Chapter 8 has been developed with ViennaMesh.

Embree

Embree is an open source high performance CPU based ray tracing kernel library [41, 125]. It is used in Chapter 8 to perform Monte Carlo ray tracing.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Chapter 5

Fast Feature Detection for Level-Set Functions

Surface geometries originating from process TCAD simulations are often characterized by small areas with pronounced geometric variations, and large areas which are only slightly bent or even entirely flat areas. Consider, for example, the corners and side walls of a trench, respectively.

The areas with pronounced geometric variations are referred to as features. During a process TCAD simulation the wafer surface evolves in time, which dynamically changes the surface geometry during each simulation time step. Therefore, an algorithm is needed that detects where on the surface new features emerge or dissolve. Additionally, depending on the used process model strict quality requirements are imposed on the feature detection (e.g., ignoring noise in the level-set function). Geometric feature detection or extraction of 2D and 3D data sets is a widely studied field, where the surface curvature is the primary metric [26, 27, 28, 29]. In the context of level-set method, the curvatures of the zero level-set have also been used in numerous ways to gain more insights into the geometry of the surface [126, 127, 128]. Moreover, the surface curvature is of high interest in other fields of computer science, such as fluid dynamics, where the relation between surface curvature and surface tension is of interest [129]. These simulations require a high numerical accuracy of the calculated curvature values, which in turn increases the run-time. However, since in this work the surface curvature is used to indicate parts of the simulation domain that can benefit from a higher resolution the quality of the calculated curvature values is less of a concern, as long as the numerical error is not *too big*. Thus, the dominant metric for feature detection is the run-time.

In this chapter, a general feature detection algorithm based on the geometric properties of a discrete surface is introduced (Section 5.1). The algorithm is independent of the chosen surface representation and is used throughout the remainder of this thesis. Due to the context of this thesis in topography simulation the primarily used surface representation are level-set functions 2D feature detection is a simplified version of the 3D case, thus, the discussion focuses on the latter and the adaptations for the 2D case are presented at the end of the discussion.

Furthermore, in Section 5.2 the different curvature calculation methods for implicit surfaces presented in Section 3.4, and a novel method presented in this section, are tested for their feasibility considering geometries originating from topography simulations. These geometries are then further investigated with respect to the run-times of the feature detection algorithm and are used to calibrate a numerical feature detection parameter for topography simulations.

Own Contributions

The original contributions in this chapter are the formulation of a feature detection algorithm for discrete surface representations (see Chapter 2) for process TCAD simulations. Additionally, an alternative way of calculating the finite differences for curvature calculation has been developed. This work was partially presented at the ASHPC 2021 conference [130] and was published as an article in the Journal of Scientific Computing [74].

5.1 Feature Detection

This section introduces a formal definition of which parts of a surface are considered to be a feature. To that end, the effects of minimal surfaces on the definition of a feature are discussed. Furthermore, two feature detection algorithms based on the two classes of surface classifications (i.e., surface curvature and angle between normal vectors) are introduced [22, 127].

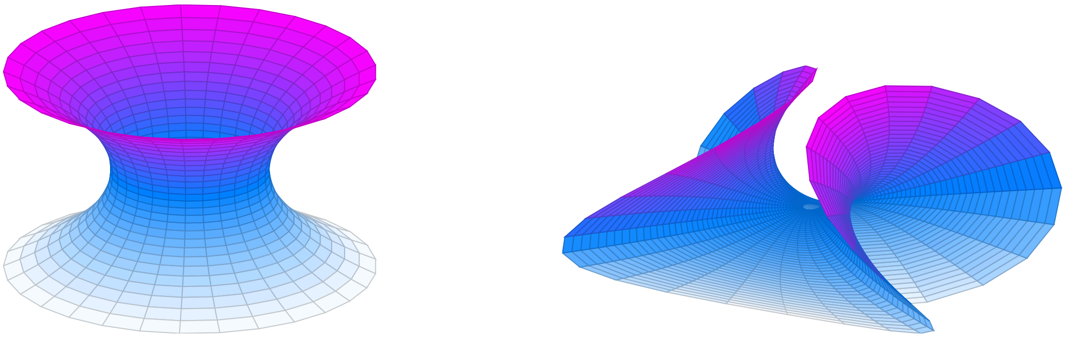
In Section 5.1.1 a feature of a 3D surface is formally defined. Next, in Section 5.1.2 the feature detection algorithm for 3D discrete surfaces using the surface curvatures is introduced. The for this thesis developed *Big Stencil* method is presented in Section 5.1.3. Furthermore, it is discussed how the calculation of the Gaussian curvature can be avoided when using the *Shape Operator* method for feature detection and still obtain a robust detection of the features of the surface. In Section 5.1.4 a variation of the feature detection algorithm is discussed that uses the surface classification method based on the normals of the surface. Furthermore, some preliminary performance tests are discussed. Finally, in Section 5.1.5 the adaptations to the feature detection algorithm for 2D surfaces are presented.

5.1.1 Feature Definition

As already mentioned in Section 3.1.2, a minimal surface is one that has a mean curvature H of 0. However, the principal curvatures κ_1 and κ_2 might not necessarily be equal to 0 on each point of the surface (see Definition 3.1.14). Another way of describing minimal surfaces is the following: Minimal surfaces are surfaces on which every point on the surface is a saddle point. The plane is a trivial example of a minimal surface, however, there are several non-trivial examples. See, for example, the surfaces depicted in Figure 5.1. For the purpose of this work it is important to distinguish parts of a surface that locally describe a minimal surface that is not a plane (e.g., a local saddle point) from an actual plane.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Catenoid **(b)** Enneper surface

Figure 5.1: Examples of non-trivial minimal surfaces which represent a curved geometry.

These observations indicate that simply considering the mean curvature of a discrete surface is not sufficient to distinguish flat parts of the surface from parts that are curved. The surfaces depicted in Figure 5.1 are obviously not a plane and bend, thus, the points on these surfaces describe features of the geometry. So, a necessary and sufficient condition for a surface point to be part of a plane is $H = K = 0$, which implies that its principal curvatures also fulfill $\kappa_1 = \kappa_2 = 0$. Therefore, if a point on a surface only fulfills $H = 0$ and its Gaussian curvature fulfills $K \neq 0$ it should be considered a feature.

If a point on a discrete surface is considered to be a feature and if it fulfills $H \neq 0$ or $K \neq 0$ the following two problems arise:

- The curvature values calculated for a discrete surface are numerical approximations, thus, small numerical errors can cause the calculated values to deviate from an analytical plane.
- If the discrete surface only bends slightly, e.g., has a small numerical mean curvature value, it should not be considered a feature since only sharp geometric variations lead to problems in the discretization of the surface.

Therefore, a feature can be defined as follows:

5.1.1 Definition (Feature) A *feature* of a discrete surface is a point on the discrete surface that has an absolute mean curvature value higher than a threshold parameter $C > 0$, and $|H| > C$ or $|H| < C$ and $|K| > C$.

Definition 5.1.1 is similar to calculating the curvedness of a surface [131]. However, the here discussed approach of defining a feature is computationally more efficient. The curvature calculation methods for surface meshes and discrete implicit functions discussed in Chapter 3 calculate the mean and Gaussian curvature of a surface. Thus, additional computational resources have to be devoted to calculating the principal curvatures in addition to calculating the curvedness. Additionally, the in Definition 5.1.1 given description of a feature avoids calculating the Gaussian curvature if the mean curvature is already sufficient to identify a surface point as a feature.

5.1.2 Algorithm

The features of a discrete surface are detected by iterating over all surface points:

- Point Clouds: All points of the point cloud.
- Surface Meshes: All vertices of the surface Mesh.
- Level-Set Functions: The grid points in the narrow band around the zero level-set.

For each surface point the absolute mean curvature $|H|$ is calculated. If $|H|$ is bigger than the feature threshold parameter (i.e., feature detection parameter) C, then the point on the surface 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:

$$\begin{cases} |H| > C, \text{ or} \\ |H| < C \text{ and } |K| > C. \end{cases} \tag{5.1}$$

The parallelization of the feature detection algorithm is straightforward since the curvatures of each point on a discrete surface have to be calculated independently of each other. Thus, the parallel speed-up is memory-bound and limited by the quality of the used domain decomposition of the discrete surface.

5.1.3 Curvature Based Feature Detection for Level-Set Functions

In Section 3.4 three different methods of calculating the surface curvatures of an implicit surface (e.g., a level-set function) are introduced. The *General Formula* and *Variation of Normal* methods can be implemented directly as they are presented in Section 3.4. The *Shape Operator* method, however, requires further discussion before it can be utilized for feature detection, which is done later in this section. Furthermore, a novel way of calculating the derivatives required for the curvature calculation is presented that does not increase the stencil size but uses underutilized grid points already present in the stencil to improve accuracy.

Big Stencil

Consider that a 19 point plane stencil η_P is present to calculate the finite differences required to determine the mean curvature of the zero level-set (c.f., Section 3.4.1 and Figure 3.8a). There exist more accurate finite difference approximations to calculate the finite differences D_x and D_{xx} [132]. These finite differences require no additional grid points, thus, they do not increase the size of the finite difference stencil. They can be calculated by the following finite difference formulas

$$\tilde{D}_x(\phi_{i,j,k}) \approx \frac{\phi_{i+1,j+1,k} - \phi_{i-1,j+1,k} + \phi_{i+1,j-1,k} - \phi_{i-1,j-1,k}}{4\Delta x}, \tag{5.2}$$

$$\tilde{D}_{xx}(\phi_{i,j,k}) \approx \frac{\begin{array}{r}\phi_{i+1,j+1,k} - 2\phi_{i,j+1,k} + \phi_{i-1,j+1,k} + \phi_{i+1,j,k} - 2\phi_{i,j,k} \\ + \phi_{i-1,j,k} + \phi_{i+1,j-1,k} - 2\phi_{i,j-1,k} + \phi_{i-1,j-1,k}\end{array}}{3\Delta x^2}. \tag{5.3}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

When the mean curvature is calculated using Equation 3.21 utilizing the finite difference approximations D_{xy}, $\tilde{D}_x$, and $\tilde{D}_{xx}$, it is henceforth referred to as the *Big Stencil* method. The here suggested method uses the same finite difference stencil as the *General Formula* method. However, it utilizes more information of the level-set function (e.g., more ϕ-values) to calculate the derivatives required in Equation 3.21. Therefore, the numerical accuracy of the calculated mean curvature values is improved. The Gaussian curvature can be calculated by using the more accurate finite difference approximations and Equation 3.22. Therefore, no additional calculations are required when using this method for feature detection.

Addendum Shape Operator

The *Shape Operator* method uses the trace of the Hessian to calculate the mean curvature of the discrete surface. By only requiring the trace this method can be calculated with a 7-point star stencil η_S (see Section 3.4.1), however, calculating the Gaussian curvature still requires a plane stencil η_P. That larger stencil η_P would negate the performance gains achieved through the smaller stencil and fewer finite differences that have to be evaluated. Therefore, a different strategy to determine if a surface point is part of a minimal surface is desired. This can be achieved by utilizing the fact that the value of the mean curvature is expressed by half the trace of the Hessian. Thus, conditions of the feature detection algorithm (see Equation 5.1) are simultaneously checked if

$$|D_{xx}(\phi_{i,j,k})| + |D_{yy}(\phi_{i,j,k})| + |D_{zz}(\phi_{i,j,k})| > 2C. \tag{5.4}$$

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

$$-D_{yy}(\phi_{i,j,k}) = D_{xx}(\phi_{i,j,k}) + D_{zz}(\phi_{i,j,k}) \neq 0. \tag{5.5}$$

Thus, $|D_{yy}(\phi_{i,j,k})| \neq 0$ and Equation 5.4 holds for points on a minimal surface that describe a feature of the zero level-set.

Nonetheless, it is possible that each element in the trace of the Hessian is equal to 0 when a point on a minimal surface that is not a plane is investigated. However, this case can be ignored considering the following deliberations. If all second-order derivatives of the trace are 0, then the basis in which the shape operator in the point $\mathbf{x}$ is expressed has a specific form. For example, let $\mathbf{x}$ be the saddle point of a hyperbolic paraboloid parametrized by $(x^2 - y^2) - (x - y)^2 = z$. When the basis vectors spanning the tangent plane of the shape operator in $\mathbf{x}$, point in a 45 degree angle between the cartesian coordinate axes x and y, then the second-order derivatives along these basis vectors are 0 since the curves are straight lines (see Figure 5.2). However, this implies that the basis of the shape operator in the adjacent points to $\mathbf{x}$ has to be different, thus, the trace of the Hessian in these adjacent points does not contain only zero entries.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

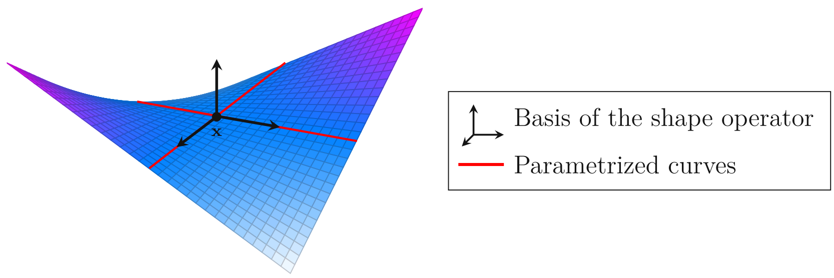


Figure 5.2: Illustration of a hyperbolic paraboloid and a basis of the shape operator in the point $\mathbf{x}$ where all second-order derivatives are 0.

The *Shape Operator* method is an often neglected method of calculating the mean curvature of a level-set function. Its numerical accuracy is highly dependent on the numerical accuracy of the signed distance function, which in performance oriented applications is only first-order accurate (see Equation 4.13). However, in Section 5.2.3 it is shown that the numerical accuracy is high enough to use this method for feature detection.

5.1.4 Surface Normal Based Feature Detection

As mentioned in Section 3.5, when considering level-set functions the angle between the surface normal vectors can be used for surface classification. To that end, the definition of a feature (see Definition 5.1.1) has to be adapted.

5.1.2 Definition (Angle Feature) A point on the surface of a level-set function is called an *angle feature*, if the angle between the normal vector $\vec{n}_{\mathbf{x}}$ and one normal vector $\vec{n}_{\mathbf{x}_i}$ on each surface point in a box stencil ($\eta_B(\mathbf{x})$) around the point $\mathbf{x}$ is bigger than an angle C_{angle}.

This method intrinsically handles minimal surfaces since it approximates the angle between the normal vector of a parametrized curve and the central normal vector in the direction of the maximum curvature κ_1 and not the mean curvature.

Algorithm Adaptation

The algorithm presented in Section 5.1.2 must be adapted to work with angle features. Instead of calculating the curvatures in each point on the surface, the surface normal in each surface point is calculated. Afterwards, a box stencil is moved over each surface point. Each point in the box stencil is checked if it is a surface point, when this point is a surface point the angle α between the central normal and the normal of the surface point is calculated, and is checked if $\alpha < C_{angle}$. If $\alpha > C_{angle}$ the central point is marked as a feature and the box stencil is moved to the next surface point, otherwise the next point in the box stencil is checked.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Run-Time Evaluation of Surface Normal Based Feature Detection

The primary drawbacks of a surface normal based feature detection on level-set functions are discussed here. Depending on the position of a surface normal on the surface it is possible that the angle between the central normal and an adjacent normal is greater than the parameter C_{angle} although the central surface normal is not part of a feature. Figure 5.3 shows an illustration of such a problem. The point $\mathbf{p}_1$ would be flagged as a feature even though it is part of a plane.

Furthermore, the computational performance (See Section 5.2) of the surface normal and curvature based feature detection are experimentally compared. In what follows, the studies have been performed on a trench geometry shown in Figure 5.4.

The run-times and speedup of the curvature based feature detection algorithm using the *General Formula*, the *Shape Operator* method to approximate the surface curvature, and the surface normal based feature detection algorithms presented in Section 5.1 are investigated.

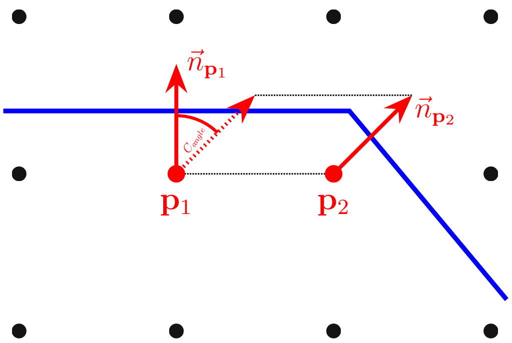


Figure 5.3: Illustration of the problems that may occur when using a surface normal based feature detection. If $\mathbf{p}_1$ is the central grid point, the angle between $\vec{n}_{\mathbf{p}_1}$ and $\vec{n}_{\mathbf{p}_2}$ would be bigger than C_{angle}, thus $\mathbf{p}_1$ is flagged as a feature, although it is part of a plane.

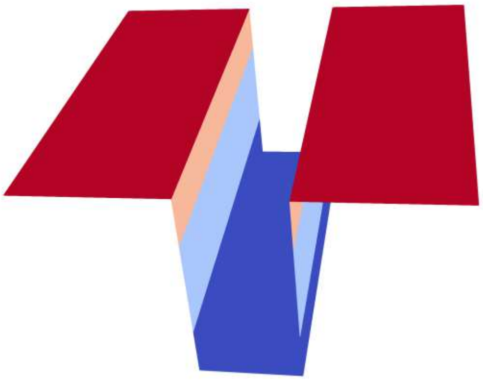
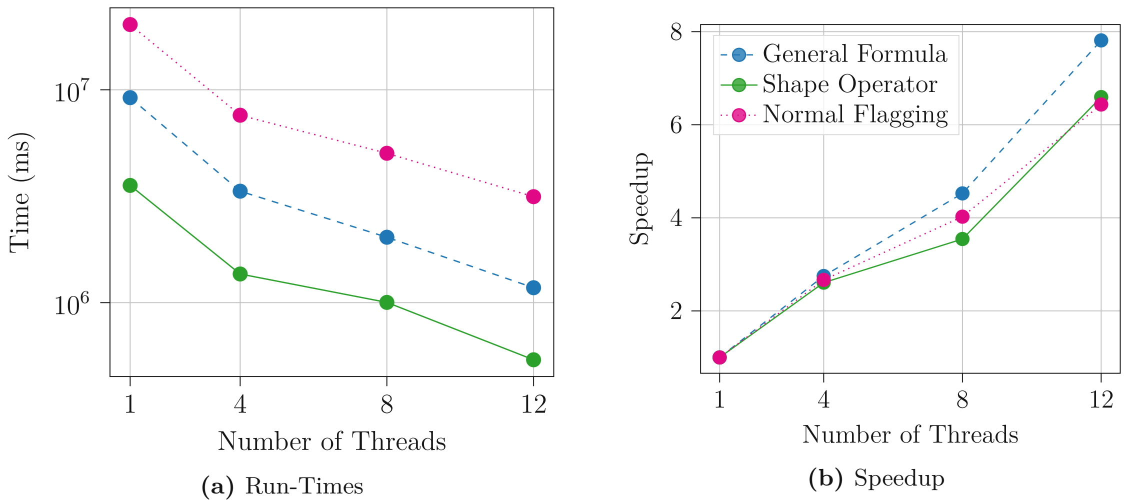


Figure 5.4: Trench geometry with 4 subdomains which hold an approximately equal amount of grid points.



(a) Run-Times **(b)** Speedup

Figure 5.5: Run-time and speedup of the feature detection algorithms on the Trench geometry.

The feature detection parameters C and C_{angle} have been chosen in such a way that they flag the same amount of grid points as features. Figure 5.5 shows the benchmarks executed on Workstation 2. The results show that the performance of the surface normal based feature detection is far worse compared to using the surface curvatures. The additional drawback of potentially flagging more surface points than needed yields the conclusion that surface normal based feature detection is the vastly inferior choice to detect features of a level-set function compared to feature detection based on surface curvature.

5.1.5 2D Feature Detection

Feature detection on 2D surfaces is a simplified version of the feature detection algorithm presented in Section 5.1.2. The definition of a feature (see Definition 5.1.1) also holds for 2D surfaces, when the term mean curvature is substituted by curvature (i.e., κ). Furthermore, in the 2D case the calculation of the Gaussian curvature and discussion about minimal surfaces is immaterial, since minimal surfaces only occur in 3D. Thus, the algorithm presented in Section 5.1.2 is easily adapted for 2D discrete surfaces. In this case, the algorithm simplifies to calculating the absolute curvature $|\kappa|$ and comparing it to the feature threshold parameter C.

Surface normal based feature detection can also be used on 2D surfaces, by modifying the box stencil which only consists of 9 grid points in 2D.

5.2 Comparison and Evaluation

The simulation results presented in this section have been generated with ViennaLS and where executed on a single node of VSC4 (see Section 4.6.3).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Hierarchical Run-Length Encoding

ViennaLS uses a Hierarchical Run-Length Encoding (HRLE) data structure to store the level-set function. The HRLE data structure only stores the narrow band around the zero level-set up to a predetermined thickness (see Section 2.3) [133]. This is achieved through a segmentation of the coordinate directions. This data structure naturally lends itself to be used in a level-set framework based on the sparse field approach [114].

Parallelization Strategy

Parallelization in ViennaLS is achieved through a domain decomposition approach that splits the entire simulation domain into subdomains. The domain is split along a coordinate axis in such a way that approximately the same amount of grid points are present in each subdomain [134].

5.2.1 Geometries and Mean Curvature Values

The applicability of the curvature calculation methods presented in Section 3.4 and the newly developed *Big Stencil* method for the feature detection algorithm presented in Section 5.1.2 is evaluated on several test geometries:

1. Sphere
2. Stacked nanosheet FET [12]
3. Selectively grown epitaxial crystal [87]

Sphere

In case of the sphere the mean curvature can be analytically calculated and compared to numerical approximations. A sphere with radius r has an analytical mean curvature of $1/r$. Furthermore, every time step in a level-set based simulation is a numerical approximation which introduces small numerical errors into the discretization of the surface. Thus, to compare the analytical mean curvature values of the sphere to values calculated during a simulation workflow, the sphere has been subjected to a velocity field. By way of example the here considered sphere is initialized with a radius of 15, and subsequently subjected to a constant velocity field with a velocity of -1 for 5 time units. This results in a sphere that should have an analytical radius of 10, and thus, a mean curvature H of 0.1. A first-order Engquist Osher scheme is used to solve the level-set equation (see Section 4.1.1). The distribution of the calculated mean curvature values of all points on the zero level-set, for the *Big Stencil* and the methods discussed in Section 3.4, is shown in Figure 5.6. Due to the numerical errors introduced by the discretization of the level-set function, the distance to the center of the sphere from all points on the zero level-set 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 discretization into the regular grid.

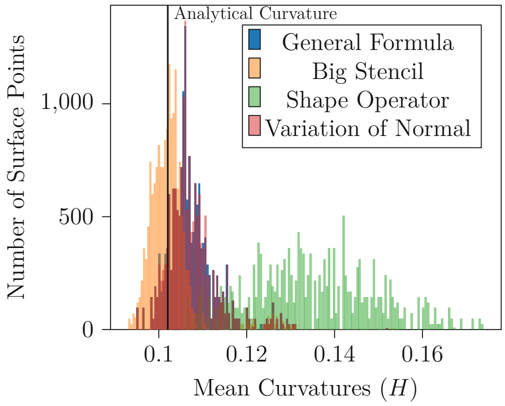


Figure 5.6: Mean curvature values for a sphere with radius 9.7 and grid resolution 0.27 after 5 time units. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Thus, the calculated mean curvature values are distributed around the analytical mean curvature of a sphere with radius 9.7, which is indicated by the black line in Figure 5.6.

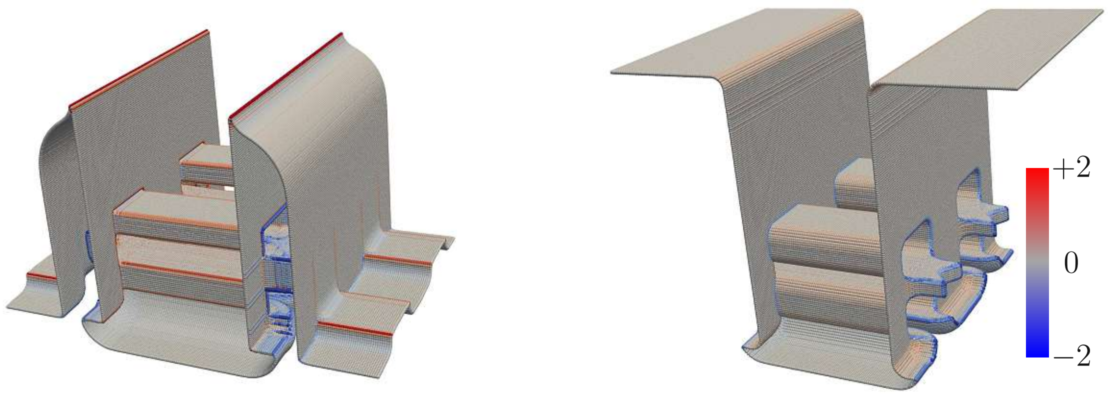
The mean curvature values computed with the *General Formula* and the *Variation of Normal* methods do not significantly differ from each other. The mean curvature values calculated with the *Shape Operator* method deviate the most from the analytical solution. However, this is expected since the star stencil uses fewer grid points, thus the numerical error of the discretization is more pronounced, as discussed in Section 5.1.3. The *Big Stencil* method produces the best match with the expected mean curvature values from the analytical solution. These investigations show that the mean curvature calculation methods, except for the *Big Stencil* method, tend to overestimate the mean curvature of the sphere. Furthermore, the *Shape Operator* method overestimates the mean curvature values by up to 75%. However, the empirical analysis presented in Section 5.2.3 shows that this error does not disqualify the *Shape Operator* method from being used for the purpose of feature detection.

Stacked Nanosheet FET

The first geometry originating form a topography simulation workflow is a stacked nanosheet FET. As discussed in Section 4.3, different materials are represented by their individual level-set functions, which are combined through layer wrapping to represent the topography of the 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 layer. Figure 5.7 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 [12]): the surfaces are the result of several processing steps, which have undergone considerable geometric changes due to the employed process models. For all the above-mentioned processing steps a first-order Engquist Osher scheme is used to solve the level-set equation.

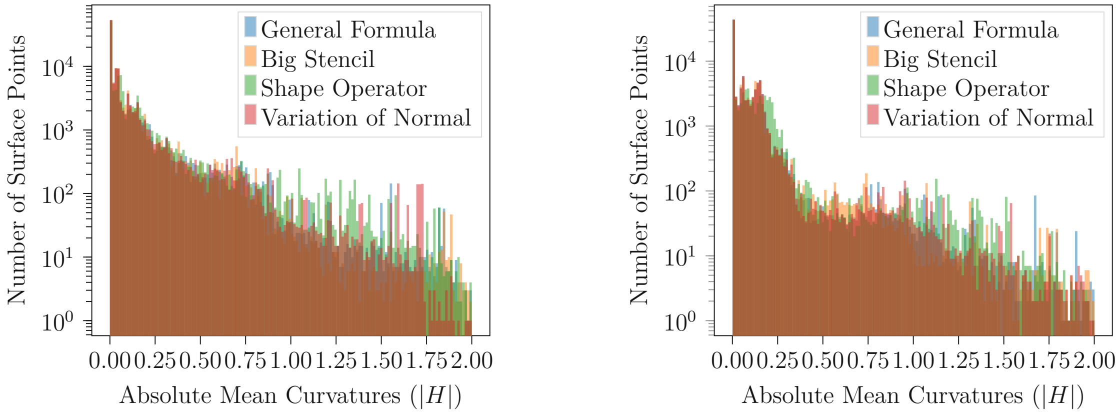
TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



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

Figure 5.7: Calculated mean curvature values of two material layers of a stacked nanosheet FET, calculated with *Shape Operator*. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.



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

Figure 5.8: Distribution of the calculated absolute mean curvature values of the two material layers of a stacked nanosheet FET shown in Figure 5.7. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

In Figure 5.8 the distribution of the absolute mean curvatures of layers 3 and 5 are shown. A similar behavior of the calculated mean curvature values as for the sphere can be identified. Specifically, the mean curvature values calculated with the *Shape Operator* method overestimate the mean curvature more than the other methods. About 96% of the calculated absolute mean curvature values fall in the interval between 0.0 and 0.5. This illustrates that most of the geometry is flat or only slightly bent, without many sharp features.

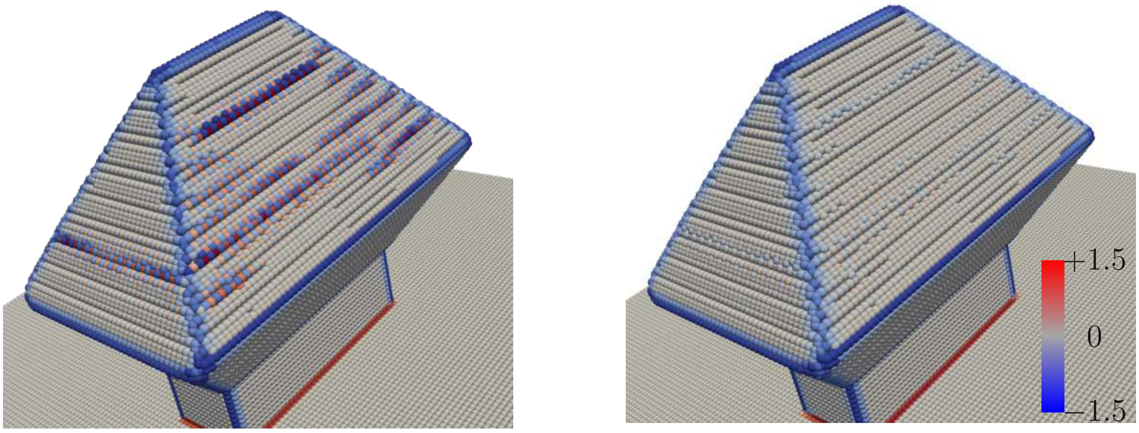
Selectively Grown Epitaxial Crystal

The second considered geometry of a topography simulation workflow is a heteroepitaxially grown SiGe crystal on a Si fin.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

These crystals are fabricated using strongly anisotropic processing techniques which are a common fabrication steps for non-planar semiconductor device geometries (e.g., FinFETs) [135]. The simulation of such processes result in non-convex Hamiltonians. In Section 4.1.1 the Lax-Friedrichs scheme is discussed. It is required to solve the level-set equation when simulating crystallographic orientation-dependent growth. Furthermore, the process model depends upon special interpolation schemes to calculate the velocities for all points on the zero level-set [136]. The selective epitaxial growth (SEG) of the SiGe crystal is characterized by crystal facets (i.e., the planes formed by the different growth characteristics depending on the crystallographic planes). The resulting device topographies contain areas with high-curvatures and essentially flat areas [137]. Simulating such processes results in level-set functions with sharp corners that have to be maintained during the simulation of the growth process.

Figure 5.9 depicts the SiGe material layer of the simulated crystal surface, and Figure 5.10 shows the distribution of the calculated absolute mean curvature values. Comparing the calculated mean curvature values of Figure 5.9a and Figure 5.9b shows that there is slight noise on crystalline facets when using the *General Formula* method. The noise is introduced by the numerical methods used to solve the process model (e.g., the Lax-Friedrichs scheme and the interpolation of the velocity values). However, the mean curvature values calculated with the *Big Stencil* method shown in Figure 5.9b are only marginally affected by the noise in the level-set function. The superior results achieved with the *Big Stencil* method concerning the noise in the level-set function is explained by the additional grid points used to approximate the second-order derivatives: As discussed in Section 3.4.2 the trace of the Hessian of the level-set function contains all information needed to calculate the mean curvature.



(a) *General Formula* method

(b) *Big Stencil* method

Figure 5.9: 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. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

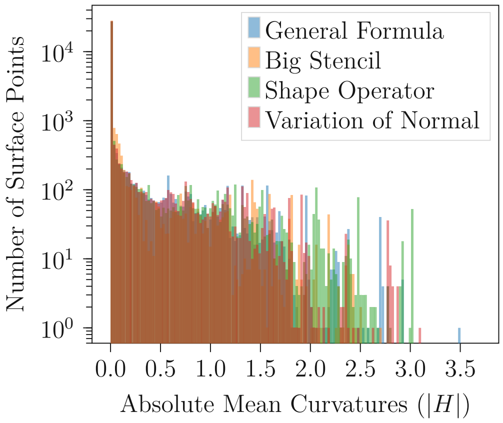


Figure 5.10: Distribution of the calculated absolute mean curvature values for the SiGe material layer shown in Figure 5.9. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Thus, increasing the numerical accuracy of the second order derivatives in the same direction (i.e., the elements of the trace of the Hessian) increases the accuracy of the calculated mean curvature. Figure 5.10 supports the above observations since most of the absolute mean curvature values are smaller than 2 and most of them are close to 0, compared to the other three methods.

5.2.2 Parameter Study

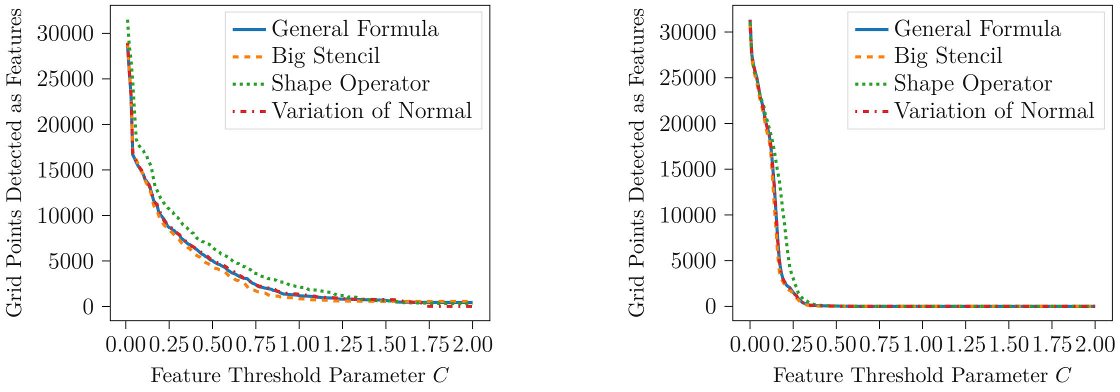
In the previous section the calculated mean curvature values of several geometries originating from process TCAD simulations have been investigated. The feature detection algorithm presented in Section 5.1.2 requires a feature threshold parameter C. Thus, to obtain such a parameter for the investigated topographies a parameter study is conducted. The parameter search starts with a (for this case appropriate) value of 0.01 for the feature detection parameter C up to $1/\Delta x$ with a step length of 0.01. $1/\Delta x$ is chosen as the maximum value, since it is the maximal absolute curvature a level-set function with a given grid resolution of Δx can describe (see Section 3.4). 0.01 is chosen as the minimum value as smaller values tend to flag nearly the entire surface as a feature. Figure 5.11 shows the different feature detection parameters C compared to the surface points detected as features for the two representative material layers of the stacked nanosheet FET.

The observations from the previous section still hold, thus, the *Shape Operator* identifies more surface points as features than the other methods. However, as long as the additional grid points which are detected as features are still adjacent to other features of the geometry they can be ignored. This suggests that the *Shape Operator* method is still able to reliably detect features of the zero level-set. The other three methods identify approximately the same number of surface points as features with only minor deviations.

The graphs depicted in Figure 5.11 indicate that the feature detection parameter C for the here considered representative topographies should be chosen between 0.1 and 1.0:

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Material layer 3.

(b) Material layer 5.

Figure 5.11: Surface points detected as features of the zero level-set compared to the chosen feature detection parameter C for the stacked nanosheet FET shown in Figure 5.7. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

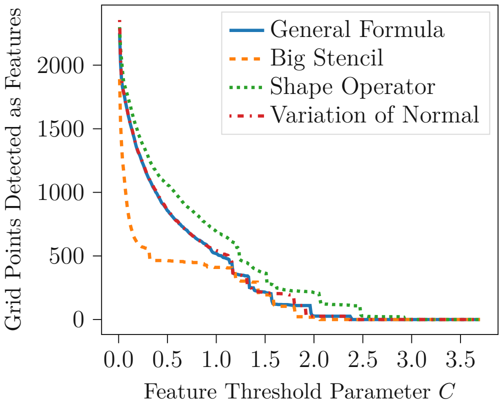


Figure 5.12: Surface points detected as features of the zero level-set compared to the chosen flagging parameter C for the heteroepitaxially grown SiGe crystal shown in Figure 5.9. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

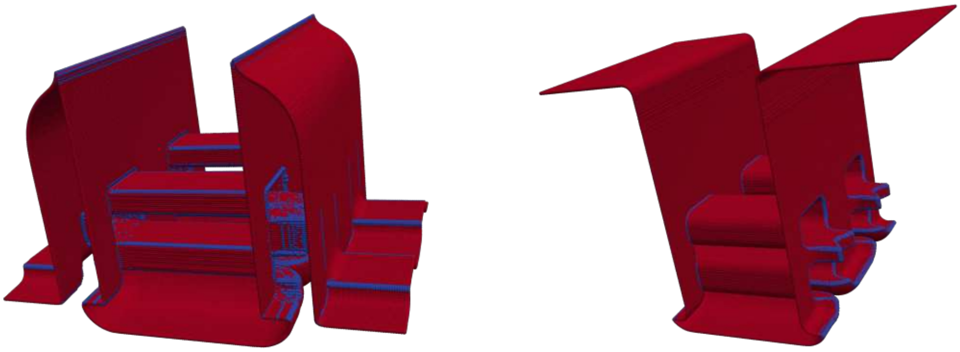
If the feature detection parameter is larger than 1.0 almost no grid points are detected as features leading to less and less features being identified. On the other hand if the feature detection parameter is chosen smaller than 0.1 it is possible that nearly all surface points are detected as features. These deliberations and Figure 5.8 suggest 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.

Figure 5.12 depicts the same parameter search for the heteroepitaxially grown SiGe crystal surface as for the stacked nanosheet FET. The *Big Stencil* method identifies far fewer grid points as features in the range of 0.2 and 1.1 than the other three methods, which contrasts the behavior of the feature detection parameter observed for the stacked nanosheet FET. As argued in Section 5.2.1 the lower amount of grid points detected as features stems from the higher accuracy of the *Big Stencil* method, which enables it to ignore the noise in the crystal facets.

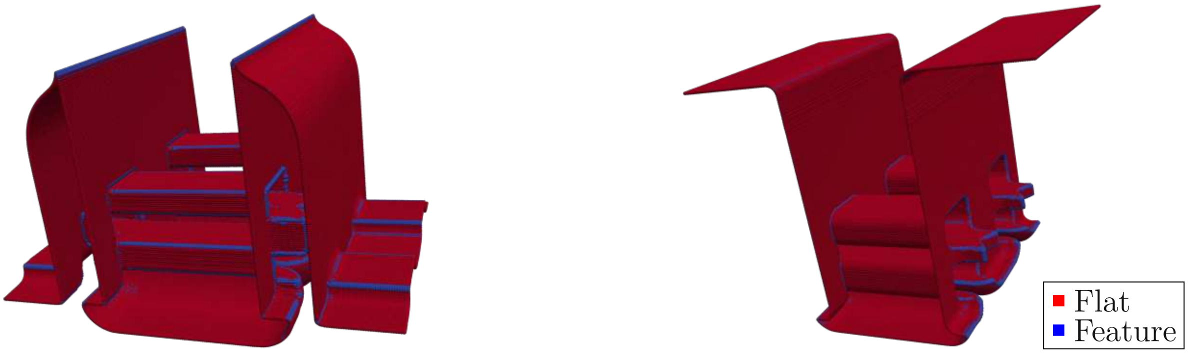
At a feature detection parameter of $C = 1.1$ (see Figure 5.12) the amount of grid points detected as features between the *General Formula*, *Variation of Normal*, and *Big Stencil* equalizes. However, as previously discussed this is a relatively high value (see the distribution of the mean curvature values in Figure 5.8 and Figure 5.10) for the feature detection parameter, which can lead the feature detection to miss important features of the investigated geometry. The amount of grid points detected as features is constant between the feature threshold parameters 0.4 and 1.1 when using the *Big Stencil* method. Thus, $C = 0.5$ is a sensible choice for the feature detection parameter when the *Big Stencil* method is used to calculate the curvatures of the SiGe crystal.

5.2.3 Empirical Evaluation

Here, the feature detection algorithm presented in Section 5.1.2 combined with the identified feature detection parameter C (see Section 5.2.2) are evaluated for their feasibility to detect features on geometries originating from process TCAD simulations. Figure 5.13 depicts the features detected by the feature detection algorithm using the *Shape Operator* and the *Big Stencil* methods on the topography of a stacked nanosheet FET.



(a) *Shape Operator* material layer 3 **(b)** *Shape Operator* material layer 5



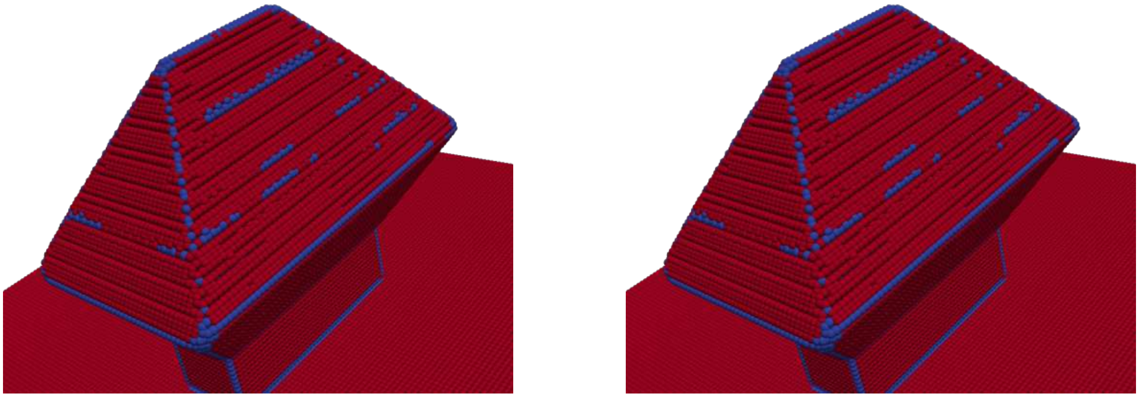
(c) *Big Stencil* material layer 3 **(d)** *Big Stencil* material layer 5

Figure 5.13: Detected features of the stacked nanosheet FET using a feature detection parameter C of 0.5. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

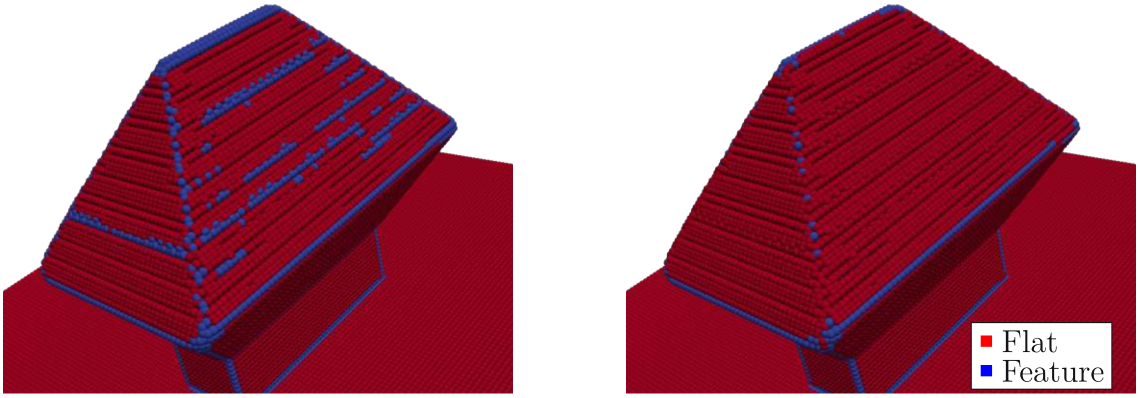
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

As already suggested by the results in Section 5.2.2, the *Variation of Normal* and *General Formula* method detect similar features to the other two methods. Furthermore, Figure. 5.13 confirms that a feature detection parameter of 0.5 is well suited to distinguish features from flat parts.

In Section 5.2.2 a feature detection parameter of 1.1 is suggested for the SiGe crystal, the results of the feature detection using this parameter are shown in Figure 5.14. Figure 5.12 shows that all curvature calculation methods detect a similar amount of grid points as features with a feature threshold parameter of $C = 1.1$. However, when consulting Figure 5.14, it is evident, that the features detected with this feature threshold parameter are inadequate, since on the one hand points on the crystal facets are detected as features and on the other hand points at the edges of the crystal are not. The falsely detected features at the crystal facets are a consequence of noise introduced into the level-set function by the process model. Reevaluating Figure 5.12 with the above deliberations in mind, it has to be concluded that the *General Formula*, *Shape Operator*, and *Variation of Normal* methods are not suited for feature detection in this scenario.



(a) Variation of Normal **(b)** General Formula



(c) Shape Operator **(d)** *Big Stencil*

Figure 5.14: Detected features of a SiGe crystal with a feature detection parameter C of 1.1. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/.`

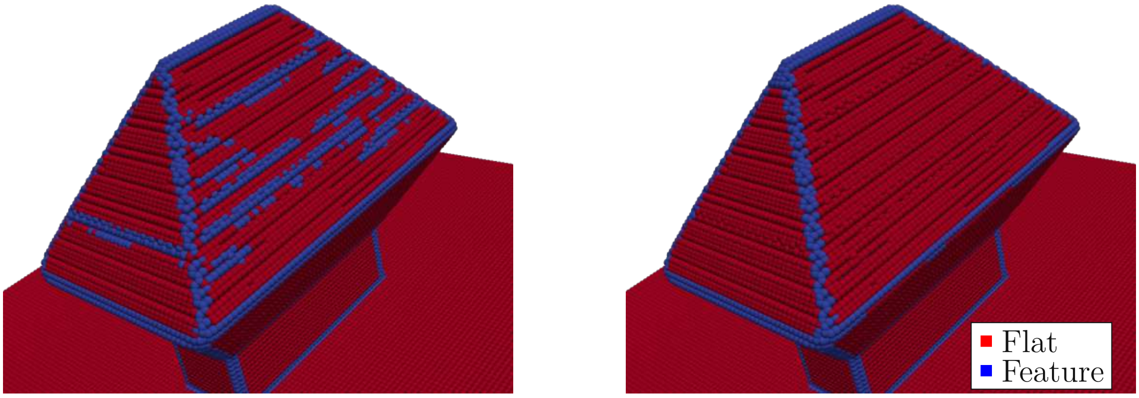
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Nonetheless, the results generated with the *Big Stencil* method still appear promising, since the higher quality of the curvature calculation allows this method to ignore the noise on the crystal facets. Therefore, the results reported in Figure 5.12 for the *Big Stencil* method have to be reassessed: These results show that the amount of grid points detected as features stays nearly constant between a feature detection parameters of 0.5 and 1.0. Thus, a feature detection parameter of 0.5 is considered for the SiGe crystal, the feature detection results for the *General Formula*, and *Big Stencil* method with this feature detection parameter are shown in Figure 5.15. The results obtained with the *Shape Operator*, and *Variation of Normal* method are omitted since they are similar to the results of the *General Formula* method. The detected features of the SiGe crystal shown in Figure 5.15b illustrates that the *Big Stencil* method achieves an adequate feature detection of the SiGe crystal using a feature detection parameter of 0.5.

5.2.4 Parallel Run-Time and Speedup

The average performance (run-time and parallel speedup) of the feature detection algorithm applied to the geometries introduced in Section 5.2.1 is tested in this section. All four discussed curvature calculation methods are evaluated. In all three run-time tests (see Figure 5.16, Figure 5.17, and Figure 5.18) the fastest curvature calculation method is the *Shape Operator* method, while the other three methods have a comparable performance.

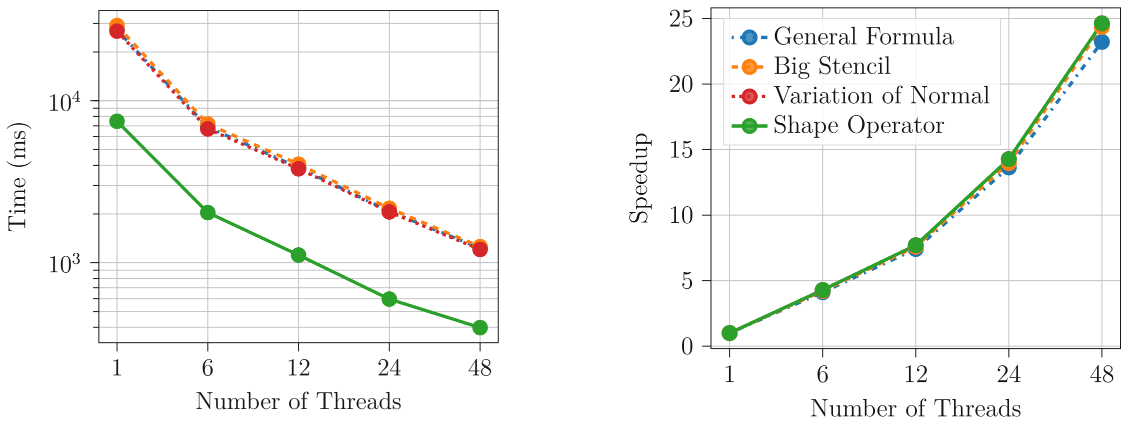
The speedup shown in Figure 5.16b illustrates the best case scenario. In light of the convexity of the sphere the domain decomposition algorithm is able to create subdomains with a very similar number of surface points. Thus, this speedup should not be expected from the more complex geometries originating from process TCAD simulations. Furthermore, the run-time and the speedup can differ due to load balancing issues, which can be observed by comparing Figure 5.17 and Figure 5.18 to Figure 5.16.



(a) General Formula **(b)** *Big Stencil*

Figure 5.15: Detected features of a SiGe crystal with a feature parameter C of 0.5. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/.`

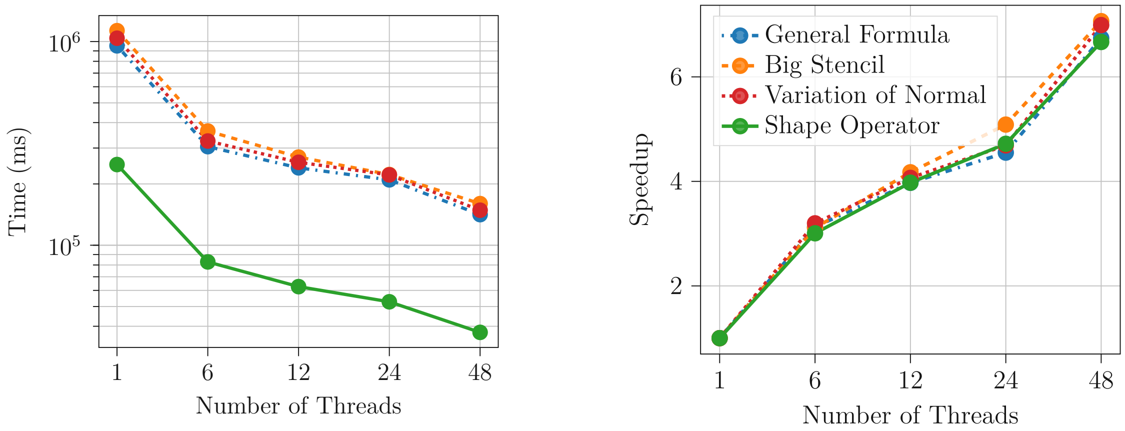
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) Run-Times

(b) Speedup

Figure 5.16: Run-time and speedup of the feature detection algorithm on a sphere with radius 9.7 and grid resolution 0.27. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.



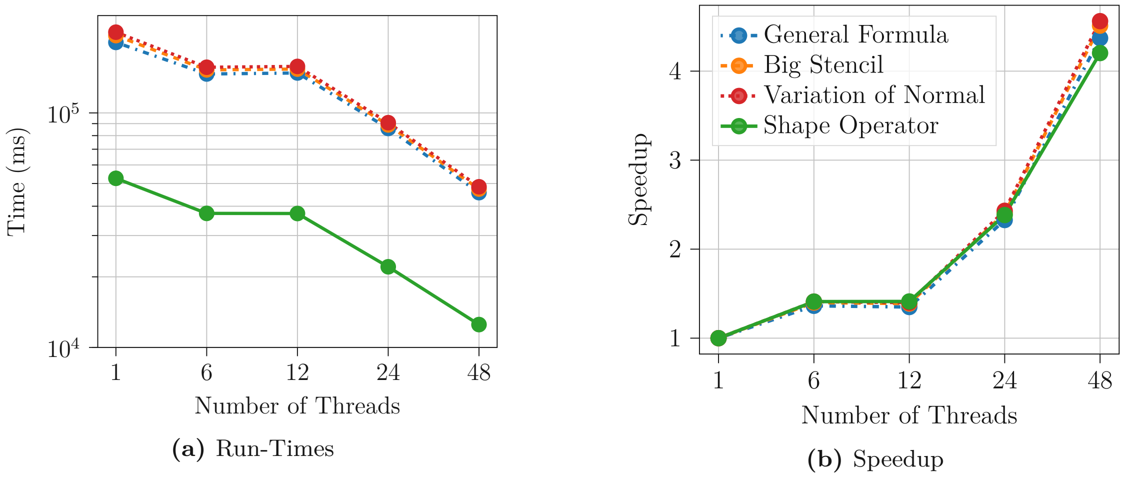
(a) Run-Times

(b) Speedup

Figure 5.17: Run-time and speedup of the feature detection algorithm for all material layers of the stacked nanosheet FET after 24 process steps. Adapted from Lenz et al., *J Sci Comput.* 71, (2023), p. 108258 [74], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) Run-Times

(b) Speedup

Figure 5.18: Run-time and speedup of the feature detection algorithm for the SiGe crystal.

In these examples the domain decomposition splits the narrow band into roughly equally sized subdomains, considering the entire grid, however, the number of surface points is not equal in each subdomain. Another issue leading to the worse speedup for the considered process TCAD geometries is the fact that feature detection (i.e., curvature calculation) is memory bound since the primary effort is the movement of the finite difference stencil over the grid.

The results presented in this section support the considerations in the previous section. Since the *Big Stencil* method does not result in a considerable increase in run-time compared to the *General Formula* and *Variation of Normal* methods it should be the primary choice for complex process models with noise. However, if the process model is simple and the primary concern is performance the *Shape Operator* method is the better choice.

5.3 Summary

In this chapter, the surface classification methods presented in Chapter 3 for the surface representations from Chapter 2 have been used to formulate a feature detection algorithm for topography simulation. Furthermore, a novel way to calculate the surface curvature using the general formula for implicit surfaces has been developed. The performance of the algorithm on 3D level-set functions with all discussed methods of calculating the curvatures has been investigated. In the course of these investigations two methods for curvature calculation of a level-set functions stood out.

The *Big Stencil* method displayed the highest numerical accuracy of the calculated curvature values. Furthermore, the *Big Stencil* method is less affected by noise in the level-set function introduced by certain process models, which has been demonstrated with the noise introduced by crystallographic orientation-dependent velocity fields. These improvements have been achieved without increasing the size of the finite difference stencil that is used for curvature calculation compared to the *General Formula* or *Variation of Normal* methods.

The *Shape Operator* method requires the smallest finite difference stencil to approximate the mean curvature of the level-set function. Moreover, when using the *Shape Operator* method the calculation of the Gaussian curvature can be avoided to check if a point on the zero level-set is part of a minimal surface, which makes this method the best performing method with respect to computational effort. However, the *Shape Operator* method has the lowest numerical accuracy of the investigated curvature calculation methods. Nonetheless, this does not preclude it from being utilized for feature detection as demonstrated in Section 5.2.3.

In summary, the insights obtained from the research in this chapter are: If run-time is the most important aspect for the feature detection the *Shape Operator* method is the preferred method to calculate the curvatures of the level-set function. On the other hand, if accuracy is the most important aspect or if there is slight noise in the level-set function, then the *Big Stencil* method should be used. Due to the *Big Stencil* methods higher numerical accuracy and similar performance to the *General Formula* and *Variation of Normal* methods. Furthermore, a parameter search for the feature detection parameter C for process TCAD simulations has been performed which resulted in a feature detection parameter of 0.5 for all presented methods.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Chapter 6

High Accuracy Hierarchical Grids for Topography Simulation

In Chapter 5, an algorithm is presented that detects areas of a discretized surface with significant geometric variations (i.e., features). As discussed in Chapter 5, typical device topographies originating from process TCAD simulations are composed of extensive areas which are essentially flat and small areas with features. These observations motivate using a simulation domain with varying resolutions (e.g., higher resolutions for features and lower resolutions for non-features) to improve simulation performance.

Locally increasing the resolution of a discretized simulation domain to improve the accuracy of the solution when numerically solving PDEs (e.g., the level-set equation) is a common strategy [19, 20, 21, 22, 23, 24]. Furthermore, these techniques are also employed in other numerical simulations like Monte Carlo simulations [138]. For process TCAD simulations, the foundations are presented in [139]. The technique of locally adapting the resolution of the simulation domain is typically referred to as *adaptive mesh refinement* (AMR). It should be mentioned here that the term mesh in AMR is a different mathematical object than a conforming mesh (see Definition 2.2.10). To avoid confusion, when the word *mesh* is used in this work, it refers to Definition 2.2.10, meshes in the context of AMR are called *grids*.

In level-set based topography simulations that utilize AMR, the entire simulation domain is covered by a so-called *base* grid (or *level 0* grid) and several rectangular refined grids (or *level 1* grids) with higher resolutions. Furthermore, it is possible that refined grids (i.e., level 2 grids) with an even finer resolution are nested inside a refined grid (i.e., level 1), which is further discussed in Section 6.1.1. A base grid with all its refined nested grids is called a *hierarchical grid*. The general term grid refers to all kinds of grids (i.e., the base grid and refined grids). Figure 6.1 illustrates three refinement levels with different resolutions.

A straightforward approach that utilizes hierarchical grids in a level-set based simulation is to cover the entire narrow band around the zero level-set with refined grids [140, 141, 142]. However, this approach still resolves non-features of the geometry with a high resolution, which leads to large areas that do not benefit from the higher resolution.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

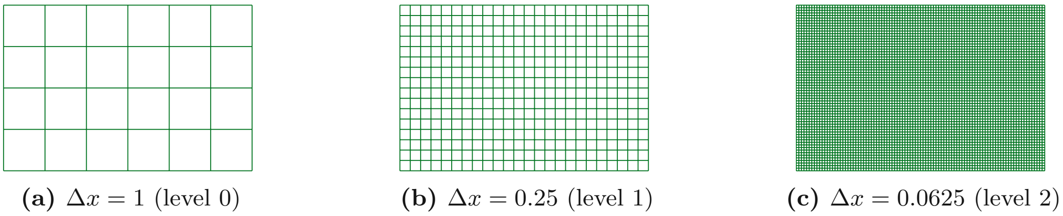

(a) $\Delta x = 1$ (level 0)
(b) $\Delta x = 0.25$ (level 1)
(c) $\Delta x = 0.0625$ (level 2)

Figure 6.1: Illustration of three grid levels of a hierarchical grid.

A computationally more efficient way of covering the simulation domain with refined grids is to detect areas of interest in the simulation domain and cover them with refined grids [143]. Therefore, a hierarchical grid placement algorithm is proposed that analyzes the device topography and only covers areas of interest (i.e., features) with refined grids [22, 144].

Figure 6.2 illustrates a level-set function with three features covered by refined grids with higher spatial accuracy (i.e., three times finer). In order to maximize the performance gains from this hierarchical approach, these refined grids need to be optimally placed, guided by an automatic feature detection method.

The automatic hierarchical grid placement algorithm consists of two parts

1. An automatic feature detection step where areas of interest (i.e., features) are marked (*flagged*) as presented in Chapter 5.
2. An automatic grid generation algorithm that clusters the flagged grid points and places refined grids with a finer resolution at these clusters (see Section 6.1.2).

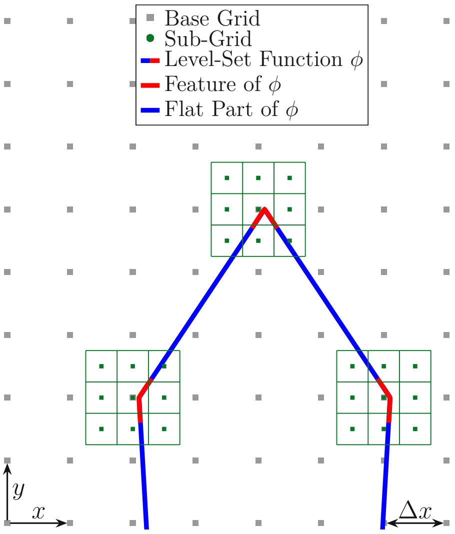


Figure 6.2: Illustration of a level-set function ϕ (blue/red line segments) with three features (i.e., red line segments) on a hierarchical grid. The base grid has a resolution of Δx and the features of ϕ are covered by refined grids with a three times higher resolution (green boxes). Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

In Section 6.1 sub-grids (i.e., refined grids with additional properties) are defined. Furthermore, nesting criteria, to avoid computational overhead, for these sub-grids and how they are placed in the simulation domain are discussed. Next, in Section 6.2 a hierarchical grid placement algorithm is introduced that places sub-grids based on the detected features of the level-set function (see Section 5.1). This hierarchical grid placement algorithm is then incorporated into the general workflow for topography simulations (see Section 4.5). Section 6.3 evaluates the performance of the hierarchical grid placement algorithm by simulating SEG of silicon-germanium (SiGe) fins in narrow oxide trenches.

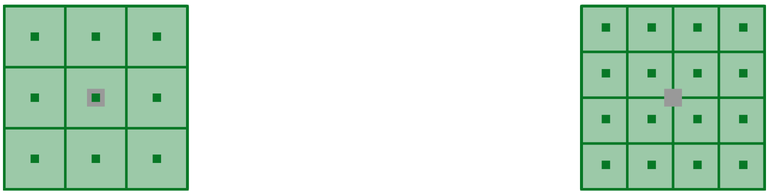
Own Contributions

This chapter introduces a combination of the previously presented feature detection algorithm (Chapter 5) with an algorithm for automatic grid generation to realize a *hierarchical grid placement* algorithm for topography simulations based on the level-set method. The work has been presented at the EUROSOI-ULIS 2021 conference [145] and was published as a journal article in Solid-State Electronics [61].

6.1 Hierarchical Grids

In the introduction to this chapter, the basic terminology for hierarchical grids is introduced (e.g., base grid, refined grids, and level n grids). For further deliberations in this work, additional terms have to be defined: A positive integer describes the so-called *refinement ratio* ($\mathrm{R}_{\mathrm{ratio}}$) which describes the ratio of the resolution between two grid levels of the simulation domain. When the refinement ratio is even, a grid point on a coarser grid level is covered by the intersection of the 4 (2D) or 8 (3D) closest grid cells on the next finer level. In the uneven case, the grid point on the coarser level is directly covered by the central grid point of the refined grid with a finer resolution. Figure 6.3 illustrates a grid cell covered by refined grids with different refinement ratios.

The refinement ratio between two grid levels must not necessarily be equal, as suggested in [146]. In the remainder of this thesis, the refinement ratio between grid levels is assumed to be constant with a refinement factor of $\mathrm{R}_{\mathrm{ratio}} = 4$, which is considered an industry standard [147].



(a) $\mathrm{R}_{\mathrm{ratio}} = 3$ **(b)** $\mathrm{R}_{\mathrm{ratio}} = 4$

Figure 6.3: Illustration of a grid point with two refined grids with different refinement ratios $\mathrm{R}_{\mathrm{ratio}}$.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

However, in Chapter 7, this convention is broken, and the last grid level refinement ratio is dynamically chosen to improve simulation performance.

To solve the level-set equation, at least a star stencil $\eta_S(\mathbf{x})$ is required (see Section 4.1.1). However, points on the border of a refined grid are missing at least one ϕ-value to calculate the finite differences. This, however, is tackled by expanding the definition of a refined grid to a *sub-grid* [147].

6.1.1 Definition (Sub-Grid) A *sub-grid* is a refined grid that stores:

- Its resolution Δx.
- Its position relative to the grid on the next coarser level.
- Its extent (i.e., the number of grid points) in all Cartesian directions.
- A set of ghost cells.

The ghost cells directly neighbor the grid cells at the border of the refined grid (see Figure 6.4). This construction allows for calculating finite differences even when the finite difference stencil extends beyond the boundary of a refined grid. Furthermore, the ghost cells are used to transfer data between neighboring sub-grids and set boundary conditions. Using ghost cells increases the required memory footprint since each sub-grid has to store its ghost cells. Additionally, if two sub-grids are adjacent, the ghost cells must be stored multiple times. If bigger finite difference stencils are required, the number of ghost cells may be increased. The concept of level n grids (for $n > 0$) can naturally be expanded to sub-grids, e.g., a level 2 sub-grid is a sub-grid with a $\mathrm{R}_{\mathrm{ratio}}^2$ finer resolution than the base grid.

6.1.1 Nesting

During a typical topography simulation workflow, numerous nested sub-grids are present in the simulation domain. Furthermore, these sub-grids have to exchange data amongst each other (e.g., between sub-grids on the same level or between grid-levels). If the sub-grids could be placed arbitrarily in the simulation domain, a substantial computational overhead would be necessary to handle the data exchange between the sub-grids since all possible relations between sub-grids have to be treated. Thus, the sub-grid placement is restricted to only allow for specific sub-grid configurations. These restrictions allow for efficient handling of the interactions between sub-grids.

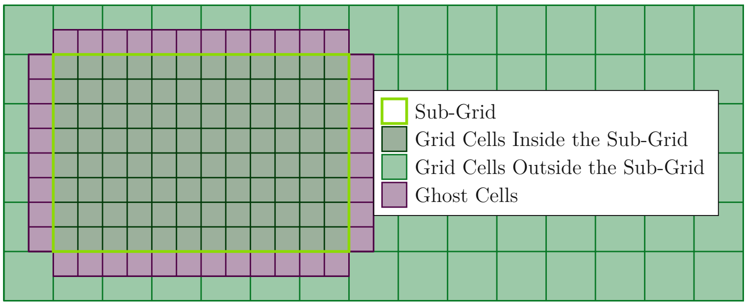

Sub-Grid
Grid Cells Inside the Sub-Grid
Grid Cells Outside the Sub-Grid
Ghost Cells

Figure 6.4: Illustration of a sub-grid: A refined grid with ghost cells on a regular grid.

When a grid G_1 on a grid level overlaps a sub-grid G_2 on a finer level, G_1 is referred to as the *parent* of G_2, inversely G_2 is referred to as the *child* of G_1.

The placement restrictions (nesting criteria) for sub-grids used in this work are as follows [147]:

1. Sub-grids may not overlap other sub-grids on the same level.
2. Each sub-grid has a unique parent grid on the next lower level (i.e., a sub-grid or the base grid).
3. Sub-grids have to be aligned to their parent's grid cells (i.e., each sub-grid is either fully covered by a grid cell of its parent or is not covered at all).
4. A sub-grid may not border an area not refined on a lower grid level.

The first two criteria guarantee that each sub-grid has a single unique parent grid, which improves computational efficiency when exchanging data between levels. The last two criteria ensure that there is a gradual refinement from a coarser to a finer resolution.

Figure 6.5 illustrates hierarchical grids that fulfill and do not fulfill the nesting criteria.

6.1.2 Grid Generation

This section discusses how sub-grids are obtained from the flags created by the feature detection algorithm. For the remainder of this section, it is assumed that a single 2D grid with several flagged grid points is given. The extension of the presented algorithm to hierarchical grids is presented in Section 6.2. Furthermore, the general algorithm is independent of the dimension of the grid [148].

The basic grid generation algorithm used in this thesis is a slight variation of the grid generation algorithm of Berger and Rigoutsos [148]. This algorithm considers the flagged and non-flagged grid points in a grid as a black and white image. In this context, the flagged grid points are considered to be black "1", and the non-flagged grid points are considered to be white "0".

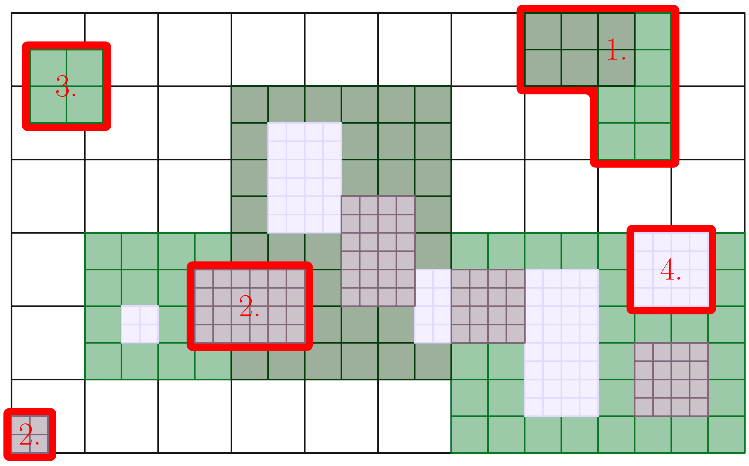


Figure 6.5: Illustration of a hierarchical grid with several nested sub-grids. The sub-grids that violate the nesting criteria are marked in red. The numbers inside the sub-grids correspond to the violated nesting criteria.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The thus defined image (i.e., the currently examined grid) is called a *patch* (P), which is put into a queue. This queue stores all patches until they are considered to be efficient, the *efficiency* of the patch P is determined by evaluating

$$\mathcal{E}(P) = \frac{\text{number of flagged grid points in } P}{\text{total number of grid points in } P}. \tag{6.1}$$

Furthermore, the minimum extent in all Cartesian directions of the patch is determined. If the calculated efficiency value of the patch is smaller than a threshold parameter E or the minimum extent of the patch has reached a minimum width M, the patch is considered efficient. The efficient patch is removed from the queue and placed into a separate *pre-grid* list.

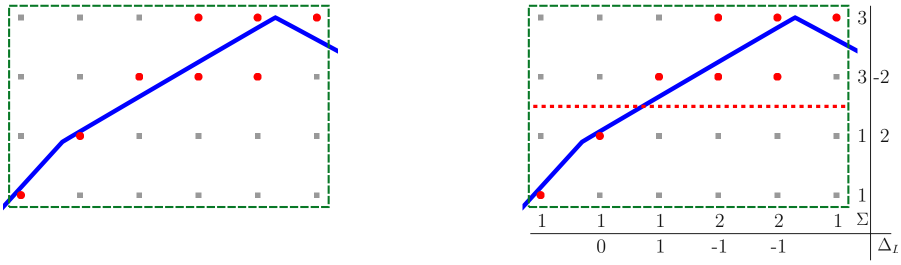
If one of the previously discussed criteria is not met, the patch is not efficient and has to be split. The position of the split is determined by calculating the *signature* (Σ) at the borders of the patch. Starting at each point at the borders of the patch, the number of flagged grid points that lie on a straight line (i.e., a grid line) passing through the patch is calculated. If an element in the signature has a value of 0, the patch can be split along this grid line. In case there are no 0s in the signature, an approach related to edge detection in images is used [148]. To that end, the Laplacian (Δ_L) of the signature has to be calculated by using the finite difference formula shown in Equation 3.19 with $\Delta x = 1$. The values of the Laplacian of the signature array are evaluated regarding so-called *zero crossings* (or inflection points), which are points where the sign of the second derivatives changes. In general, there are multiple zero crossings. Thus, the zero crossing with the biggest numerical change is chosen as the position to split the patch. Furthermore, there may be no clear point at which to split the patch, in this case, the patch is split in half. After the split, all grid lines bordering a new patch with a signature of 0 are removed. The newly created patches are added to the queue, and the next patch in the queue is considered.

Figure 6.6 shows an example of how the above discussed grid generation algorithm operates on a 2D grid. In this example, the efficiency parameter is set to $\mathcal{E}(P) = 0.3$ and the minimum patch side size to $M = 2$. The signature of the patch has no 0s, so the Laplacian of the signature array has to be calculated. There are two zero crossings (i.e., between -1 and 1 on the x-axis and between -2 and 2 on the y-axis). Thus, the zero crossing on the y-axis is chosen, and the patch is split along the indicated line in Figure 6.6b.

The grid generation algorithm terminates when the queue is empty. Afterwards, for each patch stored in the pre-grid list, a sub-grid, with a finer resolution (according to $\mathrm{R}_{\mathrm{ref}}$), is created that covers the patch. The ϕ-values of the newly created sub-grids are calculated in a two-step process. First, the ϕ-values on the newly created sub-grid are initialized using bilinear interpolation from the parent grid; This step initializes the position of the zero level-set on the new sub-grid. Second, a hierarchical re-distancing step is performed that calculates the final ϕ-values of the new sub-grid [149].

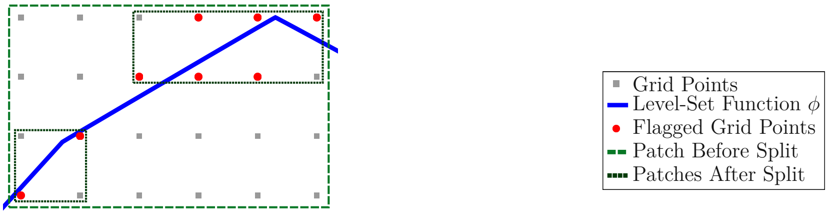
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) Illustration of a grid with a level-set function, flagged grid points, and a patch.

(b) The signature (Σ) of the patch (i.e., the numbers closest to the grid) and the Laplacian (Δ_L), the second derivatives, of the signature (i.e., the numbers next to the signature). The red dotted line indicates where the patch is split.



(c) The original patch and the two patches it is split into. Both new patches are considered efficient and the grid generation algorithm stops.

Figure 6.6: 2D example of the grid generation algorithm. With an efficiency parameter of $\mathcal{E}(P) = 0.3$ and a minimum patch side size of $M = 2$. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

6.2 Hierarchical Grid Placement

In this section, the algorithms presented in Section 5.1.2 and Section 6.1.2 are combined into a single algorithm for hierarchical grid placement for level-set functions (see Algorithm 2). Moreover, this algorithm is incorporated into a topography workflow. This algorithm is further referred to as the *regridding* algorithm. The regridding algorithm starts with a feature detection step that iterates over all grids in the simulation domain and flags the features of the level-set functions (see Section 5.1.2). It should be noted that during the flagging procedure, not only geometric features (i.e., features defined by the curvatures of the geometry) may be of interest. Examples of such non-geometric features are interfaces between multiple material layers (i.e., level-set functions). After the flagging procedure, the automatic grid placement algorithm starts by computing the new sub-grids on the base grid. The flagging and grid generation steps are executed iteratively until a pre-defined number of refinement levels – *the general refinement level* ($\mathrm{G}_{\mathrm{ref}}$) – is reached, which then terminates the grid placement algorithm. If during this process, a sub-grid is found that has not changed position and size from the previous simulation step, the previous data (e.g., the ϕ-values) is copied into the new sub-grid. Furthermore, it is checked if the sub-grids fulfill the nesting criteria (see Section 6.1.1).

Algorithm 2: Regridding algorithm for hierarchical grids.

```
input  : Hierarchical grid $G$; Feature detection parameter $C$; General
         refinement level $\mathrm{G_{ref}}$; Refinement ratio $\mathrm{R_{ratio}}$
output: Updated hierarchical grid $G$

recomputeSDF($G$); // Update signed-distance function (SDF), see
    Section 4.1.2
foreach Grid $g$ in $G$ with grid level $< \mathrm{G_{ref}} - 1$ do
    detectFeatures($g$, $C$, $\mathrm{G_{ref}}$); // see Section 5.1
preGridList ← ∅;
currLevel ← 0;
while currLevel $< \mathrm{G_{ref}} - 1$ do
    tmpGridList ← ∅;
    foreach Grid $g$ in $G$ with grid level currLevel do
        tmpGridList ← tmpGridList ∪ gridPlacemnt($g$); // see
            Section 6.1.2
    if checkNestingCriteria(tmpGridList,$G$) = False then
        /* nesting violation */
        addAdditionalFlags($G$);
        preGridList = preGridList / getLevelNGrids($G$, currLevel -1) ;
        // remove already generated sub-grids on this level
        currLevel ← currLevel -1;
    else
        preGridList ← preGridList ∪ tmpGridList;
        currLevel ← currLevel +1;
$G_{\mathrm{new}}$ ← getBaseGrid($G$);
foreach Patch $P$ in preGridList do
    if $P$ is not equal to an existing sub-grid $g$ then
        $G_{\mathrm{new}} \leftarrow G_{\mathrm{new}} \cup g$
    else
        $G_{\mathrm{new}} \leftarrow G_{\mathrm{new}} \cup$ generateNewGrid(P, $\mathrm{R_{ratio}}$);
$G \leftarrow G_{\mathrm{new}}$;
recomputeHierarchicalSDF(G); // Hierarchical SDF update[149]
```

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The first three criteria are automatically fulfilled due to the construction of the grid generation algorithm (see Section 6.1.2)

1. Sub-grids on the same level cannot overlap since each patch is split into smaller non overlapping patches.
2. Sub-grids always have a unique parent since the grid generation is executed on each grid on the same grid level individually.
3. All sub-grids are aligned to their parents grid cells since only grid cells of the parent grid are refined (see Section 6.1).

Only the fourth criteria has to checked manually. If a sub-grid is found that borders an unrefined area on the previous grid level, the points on the border are flagged as features. Afterwards, the grid generation algorithm is executed again on the previous grid level. The algorithm concludes with a hierarchical re-distancing step [149].

6.2.1 Extended Topography Simulation Workflow

The major difference introduced into the simulation workflow due to the utilization of hierarchical grids is after the advection step, where the regridding algorithm has to be executed. During the advection step, the velocities and the solution to the level-set equation must be calculated on each grid. These steps can be executed on all grids simultaneously. However, the CFL condition (see Equation 4.11) is bound by the grid resolution of the finest sub-grid in the simulation domain. This restriction is necessary since otherwise, the zero level-set on the coarser sub-grids would evolve faster than the zero level-set on the finer sub-grids. Thus, the zero level-set of the level-set function would have two different positions depending on the observed sub-grid. It should be mentioned here that it is possible to evolve the zero level-set on different grid levels with different CFL conditions. However, this has two primary drawbacks. First, the ϕ-values of the coarser grid level have to be interpolated in time to solve the level-set equation on the finer grid level. Second, several edge cases need to be handled, e.g., when another part of the zero level-set evolves into a finer sub-grid during a time step on the coarser level.

Figure 6.7 depicts an updated topography simulation workflow (see Figure 4.7). The reconstruction of the signed distance function has to be done before the feature detection to guarantee accurate ϕ-values of the level-set function after the advection step, which avoids errors in the feature detection. Furthermore, the regridding does not need to be invoked every simulation time step. Ideally, it should only be invoked when a feature moves out of a sub-grid. Thus, for this work it has been empirically determined, that the regridding procedure should be called every 4 time steps.

6.3 Benchmark Example Selective Epitaxial Growth

This section presents an evaluation of the extended topography simulation workflow introduced in Section 6.2.1. By comparing it to simulations of two SEG processes investigated by Jang *et al.* [137]. To that end, the run-time and accuracy of the simulated device topographies are analyzed.

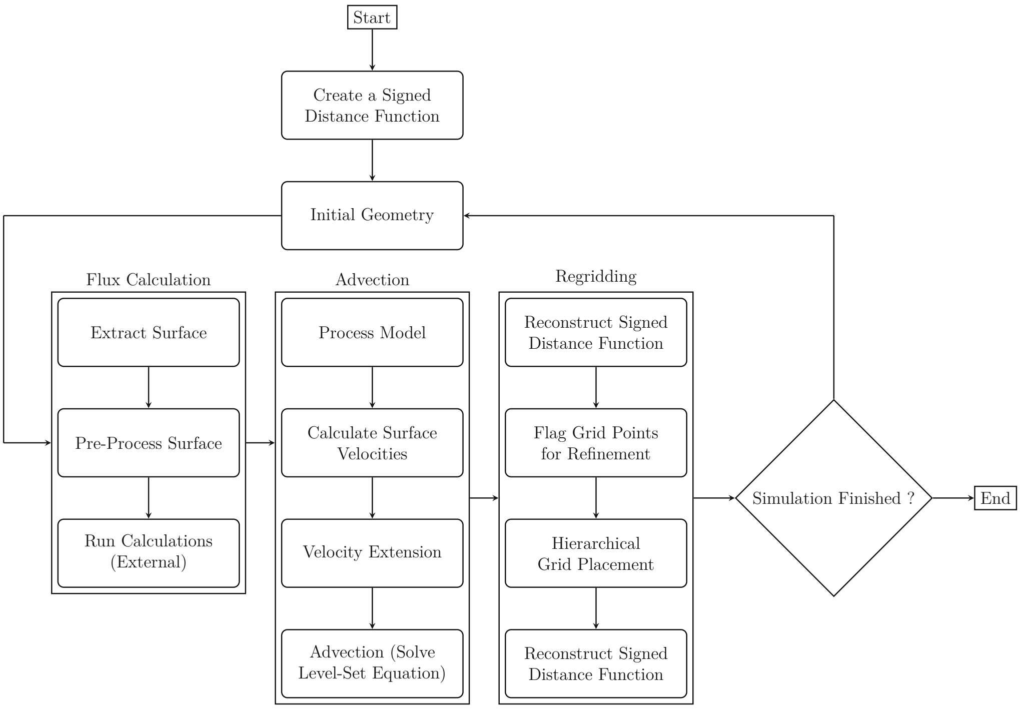

Start
Create a Signed Distance Function
Initial Geometry
Flux Calculation
Extract Surface
Pre-Process Surface
Run Calculations (External)
Advection
Process Model
Calculate Surface Velocities
Velocity Extension
Advection (Solve Level-Set Equation)
Regridding
Reconstruct Signed Distance Function
Flag Grid Points for Refinement
Hierarchical Grid Placement
Reconstruct Signed Distance Function
Simulation Finished ?
End

Figure 6.7: Updated flowchart describing a topography simulation workflow utilizing hierarchical grids.

As previously shown in Section 5.2, topographies originating from SEG processes result in geometries with pronounced features which benefit from a higher resolution. Furthermore, in this section, to ensure a well-resolved description of the level-set functions, the material interfaces with the SiGe material are considered as features. The simulations were performed with Silvaco's Victory Process and were executed on the ICS (see Section 4.6.3).

6.3.1 Simulation Setup

In the experiments presented in Jang *et al.* [137], the authors grow SiGe fins inside SiO_2 trenches. The trenches are formed in an initial dry etching step on top of a Si substrate. The SiGe is grown in a cyclic process inside the SiO_2 trench. Si_2H_6 and GeH_4 is deposited for 15 seconds from flowing source gases. Afterwards, the SiO_2 surface has to be cleaned from nucleated $Si_{1-x}Ge_x$, which is achieved by a 12s etching step by flowing Cl_2. These two steps compose a single SEG cycle. The composition of the produced $Si_{1-x}Ge_x$ alloy is influenced by the rate of the GeH_4. This cyclic process leads to the formation of high-quality $\{1\,0\,0\}$, $\{1\,1\,1\}$, and $\{3\,1\,1\}$ crystal facets.

The cyclic SEG process is modeled as a continuous epitaxy process that is perfectly selective (i.e., the deposition rate on the SiO_2 walls is 0). The process model (i.e., the velocity field) is constructed from experimentally characterized growth rates presented in Jang *et al.* [137]. Note that the process model does not directly simulate the cyclic process.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Nevertheless, the simulated surface of the SiGe crystal (that is created after one of these cycles) is referred to as a *SEG step* or a *SEG cycle*. One SEG cycle can consist of several level-set simulation time steps (i.e., advection steps).

In Jang *et al.* [137], the deposition rates on the crystallographic facets for two different alloys $Si_{0.64}Ge_{0.36}$ (SEG1) and $Si_{0.45}Ge_{0.55}$ (SEG2) were measured. For the $\{1\,0\,0\}$, $\{1\,1\,1\}$, and $\{3\,1\,1\}$ facets (the rates used for the crystal orientation-dependent process model are reported in Table 6.1). To apply these rates during a topography simulation, the process model utilizes a four-rate Hubbard interpolation to calculate velocity values for all grid points [136].

The epitaxial growth simulations of the SiGe crystals SEG1 and SEG2 are performed with three different *grid settings*, shown in Table 6.2: A singular grid without sub-grids is utilized when performing the SEG simulations with the *Fine* and *Coarse* grid settings. The simulation domains are comprised of $\approx 3 \cdot 10^5$ grid points when using *Fine* grid settings and $\approx 1.9 \cdot 10^4$ grid points when using the *Coarse* grid settings, which is why it was omitted in the here presented analysis. These two grid settings are intended to create a frame of reference against which the performance and accuracy of the hierarchical approach can be compared. For the sake of completeness, a grid resolution of 0.00125 µm is considered, which is the grid resolution between *Fine* and *Coarse*. However, a simulation using a single grid and a grid resolution of 0.00125 µm does only perform marginally better than a simulation utilizing the *Coarse* grid-setting.

The SEG simulations using the *Multi-Grid* grid settings utilize a base grid with the same spatial resolutions as *Coarse* grid settings and a general refinement level G_{ref} of 1, which leads to the generation of a plethora of sub-grids with the same spatial resolution as the *Fine* grid settings. On average, a simulation run with *Multi-Grid* grid settings consists of $\approx 2.7 \cdot 10^4$ grid points (i.e., the sum of the grid points of the base grid and sub-grids). The interfaces of SiO_2 and SiGe are considered as features (see Section 6.2). For the grid placement algorithm, the efficiency parameter is set to $\mathcal{E}(P) = 0.7$ and the minimal patch width is set to $M = 6$.

Table 6.1: Simulation parameters employed for the SEG cycles in trench arrays [137] (i.e., process model). The number of deposition cycles P_i refers to the number of SEG cycles needed to achieve the topographies in Figure 6.8 and Figure 6.12.

	Rates [nm/cycle]				Number of Deposition Cycles for Profile P		
Name	$R_{1\,0\,0}$	$R_{1\,1\,0}$	$R_{3\,1\,1}$	$R_{1\,1\,1}$	P_1	P_2	P_3
SEG1	13	5	3.1	1.6	5	24	47
SEG2	5	3	3.5	1	8	33	55

Table 6.2: Grid resolutions employed for the SEG in trench arrays (i.e., grid settings).

Simulation	Base Grid Resolution	Sub-Grid Resolution
Coarse	0.002 µm	-
Fine	0.0005 µm	-
Multi-Grid	0.002 µm	0.0005 µm

In what follows, the simulation results attained with the *Multi-Grid* grid settings are compared to the measurement data reported by Jang *et al.* [137]. Furthermore, the final SiGe crystal surfaces generated with *Fine*, *Coarse*, and *Multi-Grid* grid settings are analyzed and compared against each other. The SiGe crystal surface generated with the *Fine* grid settings is considered the comparison's reference surface since it has the highest numerical accuracy. Therefore, the error introduced by the *Coarse* and *Multi-Grid* grid settings is investigated by calculating the smallest Euclidean distance (i.e., d_E) between each surface point of the SiGe crystal simulated with *Coarse* and *Multi-Grid* grid settings and the closest point on the reference surface (d_E-error).

6.3.2 Example 1

The simulated SiGe surfaces after several SEG cycles using the SEG1 process model (see Table 6.1) and the *Multi-Grid* grid settings compared to the experiment are shown in Figure 6.8. The simulation results are in good agreement with the experiment after all three measured cycles. Figure 6.9 shows the final surfaces (i.e., after 47 SEG cycles) of the SEG1 simulation with all grid settings reported in Table 6.2. The *Fine* and *Multi-Grid* simulation results are in good agreement. On the other hand, the simulated SiGe crystal surface utilizing the *Coarse* grid settings does not match the other simulated surfaces. The error in the crystal surface is most pronounced in the mismatch of the peak of the SiGe crystal.

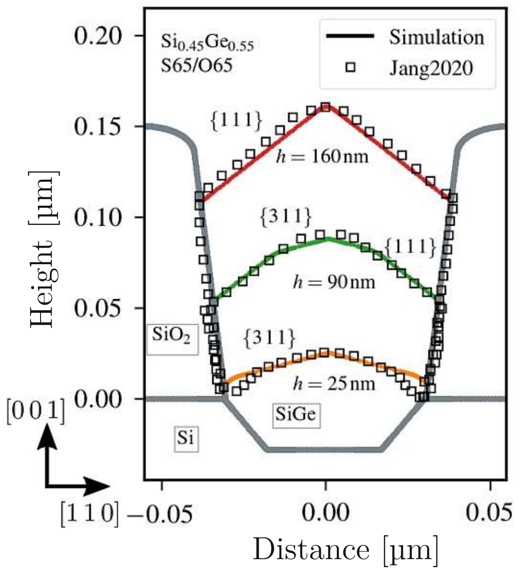


Figure 6.8: Simulated surface of the SiGe crystal using the SEG1 velocity-parameters and *Multi-Grid* grid settings compared with the experimental results from [137] after 5 (orange), 24 (green), and 47 (red) SEG cycles. The simulation results show good agreement with the experimental data. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

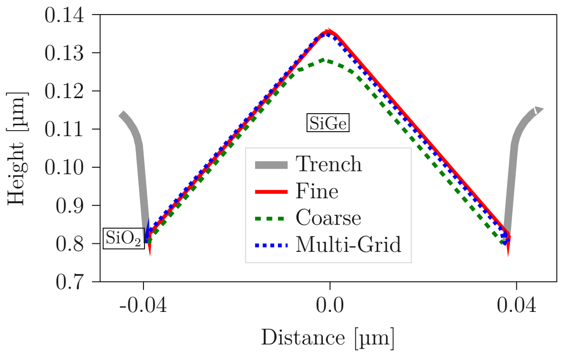


Figure 6.9: Surface for the final simulation result of the SEG1 process after 47 SEG cycles using *Coarse*, *Fine*, and *Multi-Grid* grid settings. The error in the peak of the SiGe crystal using *Coarse* resolution is the largest since the grid resolution is not high enough to simulate the SEG process at this feature properly. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

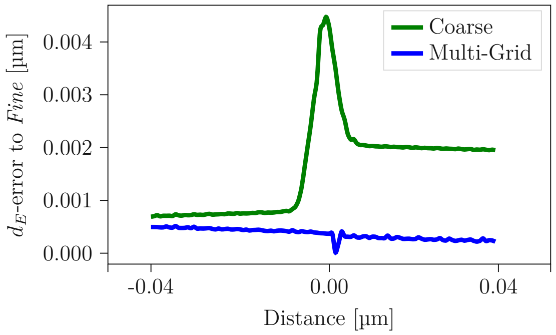


Figure 6.10: Smallest d_E-error measured from the surface points of *Multi-Grid* and *Coarse* to the nearest surface point of *Fine*, in the final simulation result of the SEG1 process (see Figure 6.9). The simulation error using the *Multi-Grid* grid settings is negligible compared to the error when using the *Coarse* grid settings. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

The d_E-error between the SiGe crystal surfaces is reported in Figure 6.10. As again indicated by the spike in the d_E-error, the simulation results obtained with the *Coarse* grid settings do not offer sufficient accuracy compared to the other grid settings. Furthermore, the d_E-error between the *Multi-Grid* and *Fine* grid settings is negligible (i.e., it is much smaller than the grid resolution), showing the anticipated increase in accuracy of the hierarchical approach. The d_E-error of the surface using the *Multi-Grid* grid settings is as big as the expected error from a simulation run using the *Fine* grid settings.

The simulation run-times of the SEG1 and SEG2 simulations run with the three studied grid settings are shown in Table 6.3: The results show that the *Fine* grid settings have the worst simulation run-times. This is expected since the high resolution used in the entire simulation domain unnecessarily resolves non-features that do not benefit from the high resolution. In contrast and as expected, the simulation using the *Coarse* grid-setting results in the fastest run-times.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 6.3: Run-times (ICS) for the entire simulation (47/55 SEG cycles) of the SEG in trench arrays with respective grid settings.

Simulation	Run-Time SEG1	Run-Time SEG2
Coarse	28 s	18 s
Fine	19 min54 s	9 min25 s
Multi-Grid	13 min38 s	3 min58 s

However, as discussed previously, the quality of the final simulated surface is insufficient. The *Multi-Grid* grid settings improves the simulation performance by 32% compared to the *Fine* grid settings.

The flagged grid points and thus generated sub-grids after 24 SEG cycles, using the SEG1 process model, are shown in Figure 6.11. During this simulation step of the SEG process, a new feature forms between the {1 1 1} and {3 1 1} crystal facets, which are detected by the feature detection algorithm. The hierarchical grid placement algorithm places the sub-grids accordingly, as indicated by (a) in Figure 6.11.

6.3.3 Example 2

The simulated SiGe surfaces utilizing the SEG2 process model and *Multi-Grid* grid settings compared to the experiment are shown in Figure 6.12. The final surfaces (i.e., after 55 SEG cycles) of the SiGe crystal using all grid settings given in Table 6.2 are shown in Figure 6.13.

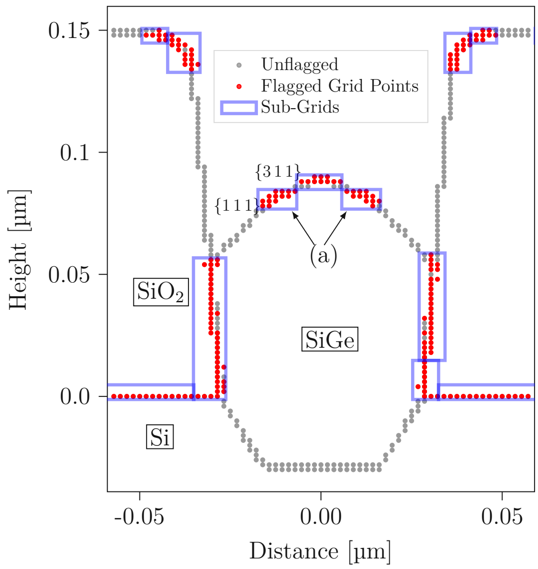


Figure 6.11: Grid points near the level-set function for the base grid of the simulation using *Multi-Grid* grid settings after 24 SEG cycles using the SEG1 process model. The flagged grid points (red) and generated sub-grids (blue boxes) for this time step are shown. (a) indicates sub-grids over fine features that develop during the SEG1 process. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

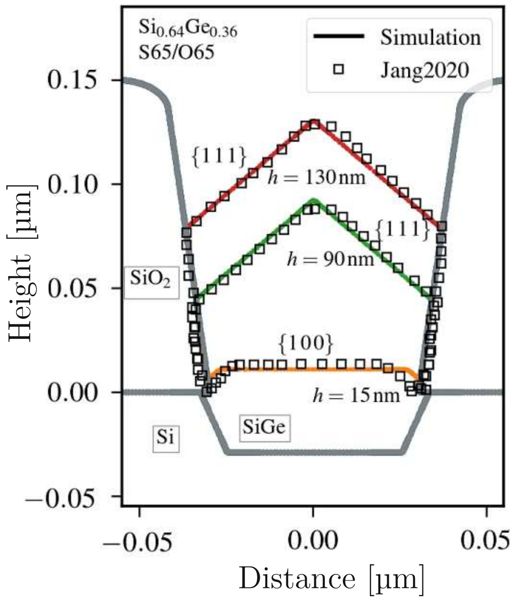


Figure 6.12: Simulated surface of the SiGe crystal using the SEG2 velocity-parameters compared with the experimental results from [137] after 8 (orange), 33 (green), and 55 (red) SEG cycles. The simulation results show good agreement with the experimental data. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

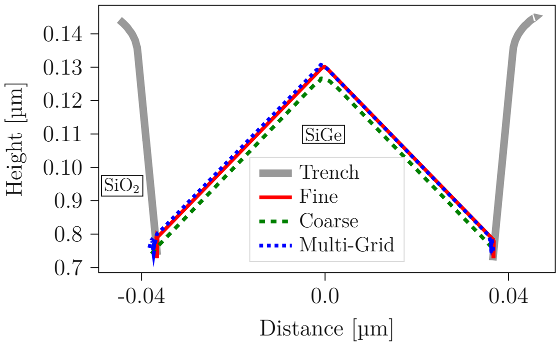


Figure 6.13: Surface for the final simulation result of the SEG2 process after 55 SEG cycles using *Coarse*, *Fine*, and *Multi-Grid* grid settings. The entire SiGe crystal surface using *Coarse* grid settings is below the other computed surfaces. Thus, in this example, the entire crystal surface has an error due to the too small resolution. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

The error between the *Coarse* and *Fine* grid settings is not as pronounced as in the first experiment (i.e., SEG1), although there is still a mismatch between the surfaces, especially at the peak of the crystal. The surfaces obtained from the *Multi-Grid* and *Fine* simulations are again in good agreement.

Figure 6.14 reports the d_E-error of the crystal surfaces simulated with the SEG2 process model. The d_E-error of the *Coarse* grid settings is again larger than the error of the *Multi-Grid* grid settings. Furthermore, the minimal d_E-error between the surfaces is bigger than in the SEG1 experiment. These results again confirm the anticipated increase in accuracy obtained by the hierarchical grid with a negligible d_E-error.

Considering the run-times reported in Table 6.3 for the SEG2 process model, the observations made in the previous experiment still hold. The *Coarse* grid-setting again has the fastest run-time, with the drawback of an inaccurate description of the SiGe crystal surface. In the SEG2 experiment the *Multi-Grid* grid settings improve the simulation run-time by 58% compared to the *Fine* grid settings.

Another point that requires discussion is the difference in total run-time between the SEG1 and SEG2 experiments (see Table 6.3). The SEG2 experiment simulates 55 SEG cycles, while the SEG1 experiment only simulates 47 cycles with more than double the simulation run-time compared to SEG2. This difference in run-time is explained by the complexity of the velocity field (i.e., process model), which influences how far the zero level-set can propagate in a single time step. A closer investigation of the actual time steps the level-set method has to calculate during the SEG simulation shows that for the *Multi-Grid* grid settings the SEG1 simulation has to perform 2259 time steps compared to 819 for SEG2. This explains the difference in simulation run-times between SEG1 and SEG2, which is due to the different number of simulation time steps that must be performed.

Figure 6.15 shows the flagged grid points and sub-grids generated by the hierarchical grid placement algorithm after 33 SEG cycles using the SEG2 process model. The process model for the SEG2 process does not generate a new feature during the SEG process.

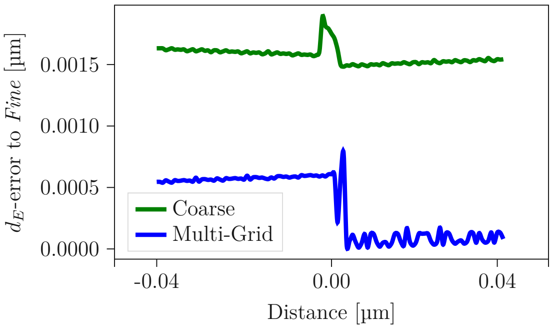


Figure 6.14: Smallest d_E-error measured from the surface points of *Multi-Grid* and *Coarse* to the nearest surface point of *Fine*, in the final simulation result of the SEG2 process (see Figure 6.13). The largest error is around the peak of the SiGe crystal of the simulation when using *Coarse* and *Multi-Grid* grid settings. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

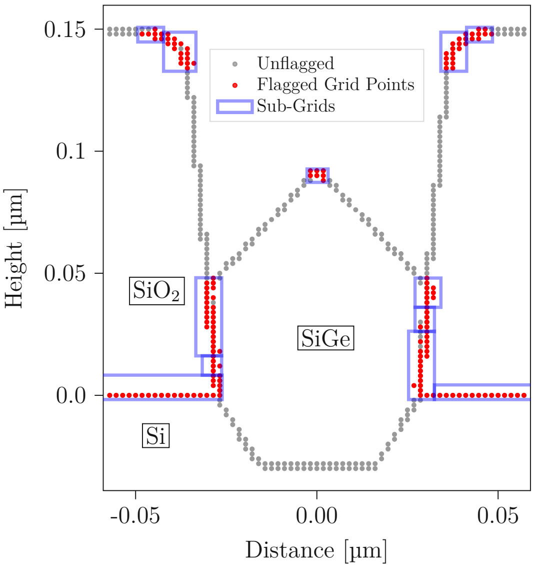


Figure 6.15: Grid points near the level-set function for the base grid of the simulation using *Multi-Grid* grid settings after 33 SEG cycles using the SEG2 process model. The flagged grid points (red) and generated sub-grids (blue boxes) for this time step are shown. Adapted from Lenz et al., *Solid State Electron.* 191, (2023), p. 108258 [61], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Thus no additional sub-grids are needed on the crystalline surface. Still, the peak of the SiGe crystal is detected as a feature, and a sub-grid is correctly placed accordingly.

6.4 Summary

The feature detection algorithm presented in Chapter 5 has been combined with an AMR algorithm to create a hierarchical grid placement algorithm. It has been discussed how the AMR algorithm determines the size and position of sub-grids. Furthermore, nesting criteria for nested sub-grids as well as a strategy to calculate finite differences at the borders of these sub-grids, have been discussed. The hierarchical grid placement algorithm has been integrated into the topography simulation workflow. To demonstrate the efficiency of the hierarchical grid placement algorithm for topography simulations, representative and practically relevant simulations of selectively grown epitaxial SiGe fins in oxide trenches have been performed. The SEG process with two different parameter sets has been successfully simulated. During the simulation of the SEG processes, the hierarchical grid placement algorithm allows for a low base grid resolution combined with sub-grids with a higher resolution which are optimally placed to cover the features of the SiGe crystal. The hierarchical grid placement algorithm presented in this chapter improves the simulation run-time by 32% and 58% with respect to the two process models, SEG1 and SEG2, respectively compared to a simulation run with a single high resolution grid while also upholding accurately.

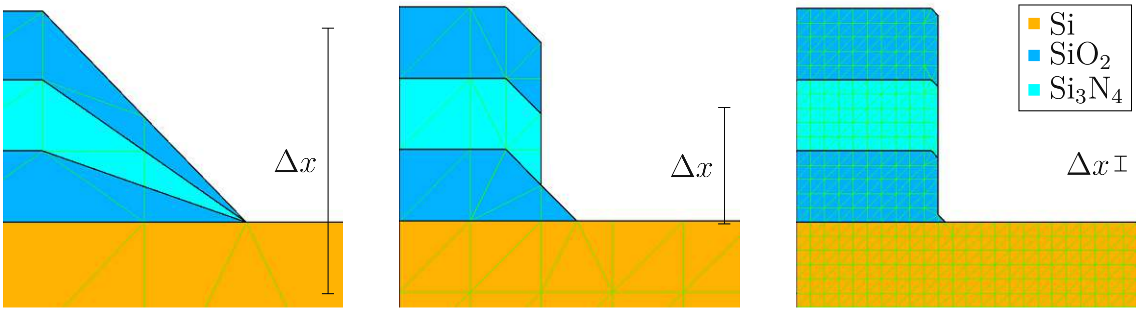
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Chapter 7

Thin Material Layer Refinement for Etching Simulations

Thin material layers are structures that commonly occur during the fabrication of semiconductor devices (e.g., LEDs or *staircase* patterns in 3D NAND flash memories) [31, 32]. During the fabrication of such devices, thin material films are deposited on top of the wafer, which are subsequently partially etched to create the desired device topography. Therefore, the appropriate handling of thin material layers is important to enable highly accurate process TCAD simulations of cutting edge electronic devices. The fabrication processes utilized during the fabrication of such devices (i.e., LEDs or NAND flash memories) contain multiple etching process steps. Etching processes can be simulated with Boolean operations of level-set functions, which will be discussed in Section 7.1. As long as a flat thin material layer (e.g., in the nm regime) continuously passes through the simulation domain, i.e., after the deposition of a thin material film, the level-set method is able to represent these material layers with a coarse (e.g., µm) resolution. However, when these thin material layers are etched in a subsequent process step (i.e., using Boolean operations), the low resolution of the simulation domain gets exposed and numerical artifacts manifest, such as those shown in Figure 7.1:



(a) $\Delta x = 0.75$µm **(b)** $\Delta x = 0.25$µm **(c)** $\Delta x = 0.005$µm

Figure 7.1: Thin material layers with a thickness of 0.25µm after a Boolean operation with different grid resolutions. Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

The relative thickness d of the thin material layers stays constant in all three depicted scenarios, whereas the grid resolution is varied. The resolution in Figure 7.1a is $4d$, in Figure 7.1b $2d$, and in Figure 7.1c, $d/6$. As is shown, the effects of the numerical artifacts are reduced when the resolution of the simulation domain is increased. However, as discussed in Chapter 6, merely increasing the resolution of the simulation domain is prohibitive due to its impact on the simulation performance.

To alleviate this issue, the hierarchical grid placement algorithm presented in Section 6.2 is used. This algorithm is based on the geometric features of the zero level-set (i.e., surface curvatures). It can thus detect where a thin material layer has been etched. However, the detection of the features happens after the Boolean operation has been performed. Therefore, the entire Boolean operation has to be performed again after the newly created sub-grids have been placed. Thus, the performance of the simulation can be improved by performing the refinement before the Boolean operation is applied. Furthermore, the geometric feature based feature detection strategy does not utilize additional information about the topography available when considering etching simulations of thin material layers (e.g., the thickness of the material layers). Nevertheless, the hierarchical grid placement algorithm based on geometric features serves as a benchmark for the algorithm proposed in this chapter.

Furthermore, it is important to note that the here discussed problem cannot be solved by the wrapping layer approach presented in Section 4.3 (see Figure 4.6). The problems in the discretization of the level-set functions discussed in this chapter stem from an insufficient resolution of the simulation domain, whereas the wrapping layer approach presented in Section 4.3 prevents the formation of undesirable voids when two level-set functions are stacked on top of each other.

In Section 7.1 it is discussed how etching processes can be simulated using Boolean operations. Section 7.2 describes the newly developed thin layer refinement algorithm. First the two primary aspects of the algorithm are discussed as well as how to determine the required resolution and how to detect the material layers affected by a Boolean operation, which are then combined into the thin layer refinement algorithm. Finally, the thin layer refinement algorithms performance is benchmarked by simulating the fabrication of a single LED pixel of an LED array (Section 7.3).

Own Contributions

The contributions in this chapter are the formulation of a flagging algorithm for Boolean operations on thin material layers. Furthermore, the algorithm is able to determine a desired target resolution to properly represent the thin material layers after the simulated etching process. This work was presented at the SISPAD 2022 conference [151] and was published in a journal article in Solid-State Electronics [150].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

7.1 Etching Simulations with Boolean Operations

As discussed in Section 2.3.3 a Boolean operation between two level-set functions can be interpreted as a Boolean operation between volumes. An etching simulation can be interpreted as the removal of materials from the wafer surface until a specific volume is removed. Thus, a process step that etches the wafer surface can be simulated in a level-set based simulation framework by Boolean operations between level-set functions [58].

First, a description of the volume (i.e., the material) that is removed (i.e., etched) from the wafer surface is required, which is represented by an additional level-set function χ. Figure 7.2 shows an illustration of this process. The etching process is simulated by calculating the relative complement of all level-set functions representing material layers and the level-set function χ representing the volume removed from the wafer.

7.2 Hierarchical Grid Placement for Thin Material Layers

The algorithm for Boolean operations presented in this section determines the distance between the two closest material layers affected by the Boolean operation. This information is then further used to calculate a *minimal required local resolution* (Δx_{tar}) to prevent the artifacts shown in Figure 7.1 from forming. Suppose the resolution of the final sub-grid is not fine enough to represent the thin material layers after the Boolean operation adequately. In that case, the grid cells are flagged for refinement, and a modified version of the hierarchical grid placement algorithm presented in Section 6.2 is executed.

First the procedure used to calculate Δx_{tar} from the ϕ-values of the level-set functions is discussed. Next, a procedure is described that is able to determine if and where a level-set function is affected by a Boolean operation with another level-set function. These two procedures are then combined to formulate the thin layer refinement algorithm.

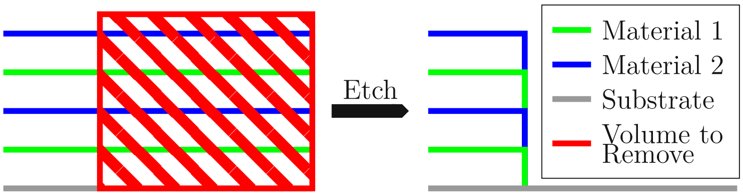


Figure 7.2: Illustration of four stacked material layers on a wafer and a simulated etching process using a Boolean operation. Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, https://creativecommons.org/licenses/by-nc-nd/4.0/.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

7.2.1 Calculating the Minimal Required Resolution

To calculate Δx_{tar} for accurately representing a thin material layer affected by a Boolean operation, the distance to the closest other material layer (i.e., the distance between the zero level-sets) affected by the Boolean operation ($\mathrm{d}_{\mathrm{closest}}$) has to be determined. In general the distance d between two level-set functions (i.e., ϕ and ψ) can be determined by utilizing the ϕ/ψ-values of the level-set function at a grid point (i, j) by calculating

$$\mathrm{d}_{(\phi_{i,j},\psi_{i,j})} = \phi_{i,j} - \psi_{i,j}. \tag{7.1}$$

An illustration of this calculation is shown in Figure 7.3.

In Section 2.3, the convention that negative ϕ-values describe the inside and positive ϕ-values the outside of the volume represented by the level-set function has been introduced. This convention is essential in the following considerations since the sign of two level-set functions following this convention gives information about the direction of the outward pointing normal vector of the zero level-set.

In the case that $\mathrm{d}_{\phi_{i,j},\psi_{i,j}} > 0$ the level-set function ϕ lies in outward normal direction of ψ, in the case $\mathrm{d}_{\phi_{i,j},\psi_{i,j}} < 0$ it does not. If $\mathrm{d}_{\phi_{i,j},\psi_{i,j}} = 0$, the two level-set functions overlap each other in that point and are treated as if ϕ does not lie in outward normal direction of ψ. Only checking the distances in outward normal direction guarantees that each material layer on top of the wafer gets a distance value assigned. Figure 7.4 shows an illustration of two level-set functions and three examples of distances in outward normal direction.

Therefore, the closest distance (in outward normal direction) between a fixed level-set function ψ and all other level-set functions $\phi^{k\in 1..n}$ with $\phi^k \neq \psi$ in the domain in the grid point (i, j) is calculated as follows

$$\mathrm{d}_{\mathrm{closest}} = \min_{\phi^{k\in 1..n}}(\{\mathrm{d}_{(\phi^k,\psi)} | \mathrm{d}_{(\phi^k,\psi)} > 0\}). \tag{7.2}$$

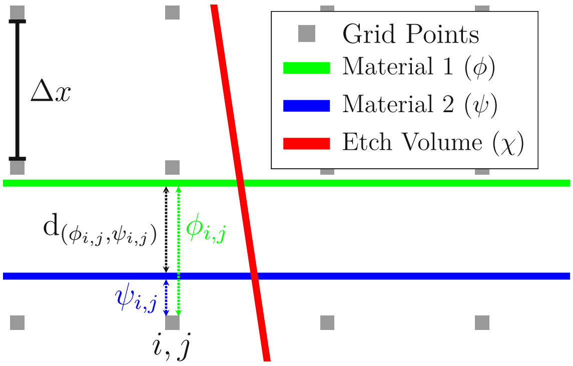


Figure 7.3: Level-set functions involved in a Boolean operation: ϕ and ψ represent material layers and χ represents the material to be removed. The distance between the level-set functions $\mathrm{d}_{(\phi_{i,j},\psi_{i,j})}$ is calculated by using the ϕ, ψ-values of the grid point (i, j). Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, https://creativecommons.org/licenses/by-nc-nd/4.0/.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

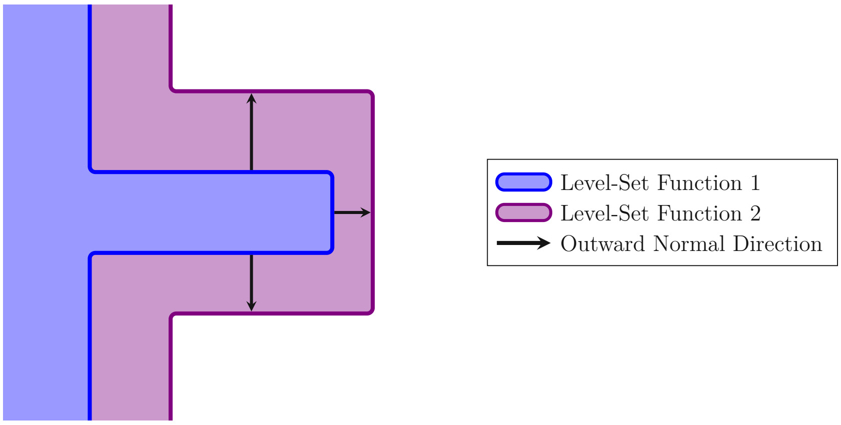


Figure 7.4: Illustration of two level-set functions representing material layers and examples of the distance in outward normal direction with respect to the level-set function 1 at different points of the level-set function.

To associate the calculated thickness of a material layer (i.e., $\mathrm{d}_{\mathrm{closest}}$) with a grid resolution, an additional parameter is required. The parameter describes the minimal number of grid points that are required to represent a single material layer ($\mathrm{N}_{\mathrm{min}}$).

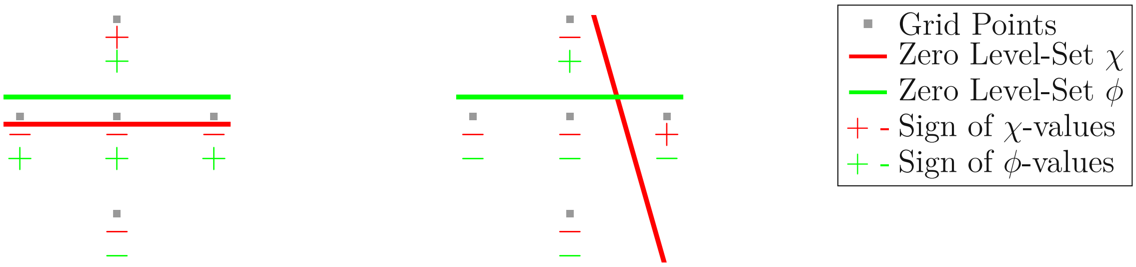
By combining the thickness of a material layer with the minimal number of grid points, the *minimal required resolution* Δx_{tar} can be expressed as

$$\frac{\mathrm{d}_{\mathrm{closest}}}{\mathrm{N}_{\mathrm{min}}} = \Delta x_{\mathrm{tar}}. \tag{7.3}$$

7.2.2 Detection of Affected Material Layers

To detect if the zero level-sets of two level-set functions ϕ and χ intersect each other near a grid point (i, j), the ϕ and χ-values in a star stencil η_S around the grid point have to be analyzed. First, the absolute ϕ and χ-values of the central grid point of the star stencil (i.e., the grid point (i, j)) are examined. If both are smaller than the *resolution of the currently examined sub-grid* (Δx_{curr}), then the two level-set functions may intersect each other close to the examined grid point. However, only considering the central grid point also detects zero level-sets that run parallel to each other or are part of the wrapping layer. Thus, the other points in the star stencil have to be examined. When the signs of at least two of the ϕ or χ-values change in two coordinate directions, the two zero level-sets intersect each other near the grid point (i, j). Otherwise, the two zero level-sets do not intersect each other near the grid point (i, j). An illustration of two intersecting and two non-intersecting zero level-sets are shown in Figure 7.5.

TU Bibliothek Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar. The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) Non-intersecting zero level-sets

(b) Intersecting zero level-sets

Figure 7.5: Two level-set functions and the signs of the ϕ-values for non-intersecting (a) and intersecting (b) zero level-sets. Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

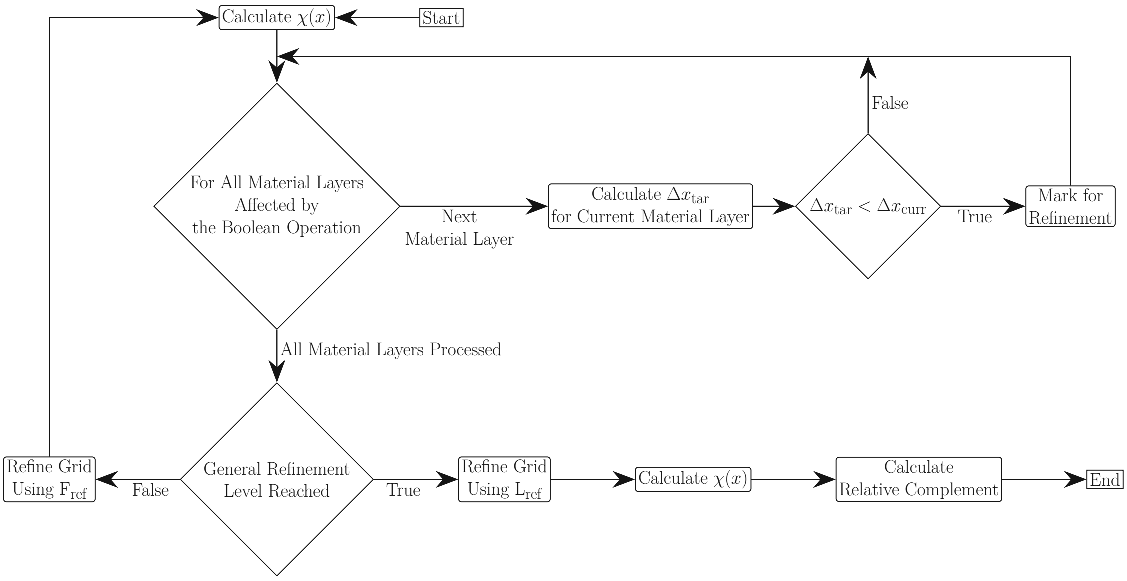

Calculate $\chi(x)$
Start
False
For All Material Layers Affected by the Boolean Operation
Next Material Layer
Calculate Δx_{tar} for Current Material Layer
$\Delta x_{\text{tar}} < \Delta x_{\text{curr}}$
True
Mark for Refinement
All Material Layers Processed
Refine Grid Using F_{ref}
False
General Refinement Level Reached
True
Refine Grid Using L_{ref}
Calculate $\chi(x)$
Calculate Relative Complement
End

Figure 7.6: Flowchart of the thin layer refinement algorithm operating on hierarchical grids. Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

7.2.3 Thin Layer Refinement Algorithm

The procedures described in Section 7.2.1 and Section 7.2.2 are now combined into the thin layer refinement algorithm, Figure 7.6 depicts a flowchart of the entire algorithm. The algorithm starts by determining the volume that has to be removed on the base grid and calculates the respective level-set function χ. Then all level-set functions in the simulation domain intersecting the level-set function χ are determined (see Section 7.2.2). The level-set functions with their respective grid points that identify the intersections are stored in a list.

Next, the minimal distance in outward normal direction from all zero level-sets at the intersecting grid points to all other zero level-sets is calculated. This is achieved by calculating the distances with Equation 7.1 and checking the signs of the calculated distances if they are positive (see Section 7.2.1).

When the calculated distance is negative, it is disregarded, and the distance to the next level-set function is calculated. If the resolution of the currently examined grid Δx_{curr} is bigger than Δx_{tar} (see Equation 7.3), the grid points are flagged for refinement.

After all level-set functions stored in the intersection list have been processed, the hierarchical grid placement algorithm is executed (see Section 6.2). The level-set function χ is updated accordingly to the newly placed hierarchical grids. This refinement process is repeated on the newly generated sub-grids until the maximum refinement level (i.e., $\mathrm{G}_{\mathrm{ref}}$) is reached.

In a final refinement step, the resolution required to represent the thinnest material layer properly is calculated. The refinement ratio for the final sub-grids ($\mathrm{F}_{\mathrm{ref}}$) is calculated by combining the target resolution (i.e., Δx_{tar}), the resolution of the current sub-grid (i.e., Δx_{tar}), and the refinement ratio (i.e., $\mathrm{R}_{\mathrm{ratio}}$)

$$\left\lceil \frac{\log\left(\frac{\Delta x_{\mathrm{curr}}}{\Delta x_{\mathrm{tar}}}\right)}{\log(\mathrm{R}_{\mathrm{ratio}})} \right\rceil = \mathrm{F}_{\mathrm{ref}}. \tag{7.4}$$

Thus, the calculated resolution is used in a concluding hierarchical grid placement step.

The advantage of the refinement algorithm presented in this chapter over an algorithm based on geometrical features of the surface (see Chapter 5) is its ability to dynamically adapt the refinement based on the thickness of the material layers involved in a Boolean operation. Additionally, the knowledge about the thickness of the material layers enables the algorithm to deviate from a fixed refinement ratio.

7.3 Benchmark Example LED Pixel Fabrication

The in Section 7.2.3 presented algorithm is evaluated by simulating the fabrication of an individual LED pixel of a LED array reported in the literature [32, 152]. The simulations presented in this section have been executed with Silvaco's Victory Process and where executed on the ICS (see Section 4.6.3).

7.3.1 Simulation Setup

The fabrication process of an LED pixel starts by growing a 1.9µm thick GaN layer on a (0001) sapphire substrate. In the next ten process steps alternating layers of InGaN and GaN with different thicknesses are deposited up to a total height of 117.5nm. Afterwards, a p-GaN cap layer with a thickness of 210nm is grown on top of the structure [32, 152]. The thinnest material layer in the entire structure is a 3nm thick InGaN layer. The excess material is etched to fabricate an individual LED pixel with a diameter of 75µm.

The base grid resolution of the simulation is set to 0.125µm. The minimal number of grid points to represent the thinnest material layer (i.e., $\mathrm{N}_{\mathrm{min}}$) is set to 6. The parameters for the automatic Grid placement are $\mathcal{E}(P) = 0.7$ and $M = 6$.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Additionally, a minimum of two grid refinement levels is presupposed (i.e., $G_{ref} = 2$) to guarantee an accurate description of corners in the simulation domain. The here presented simulation study focuses on the final etching step of the fabrication simulation. Evaluating Equation 7.4 with the discussed simulation parameters shows that the final sub-grid needs at least a 256 times finer resolution than the base-grid for an accurate representation of the thinnest material layer.

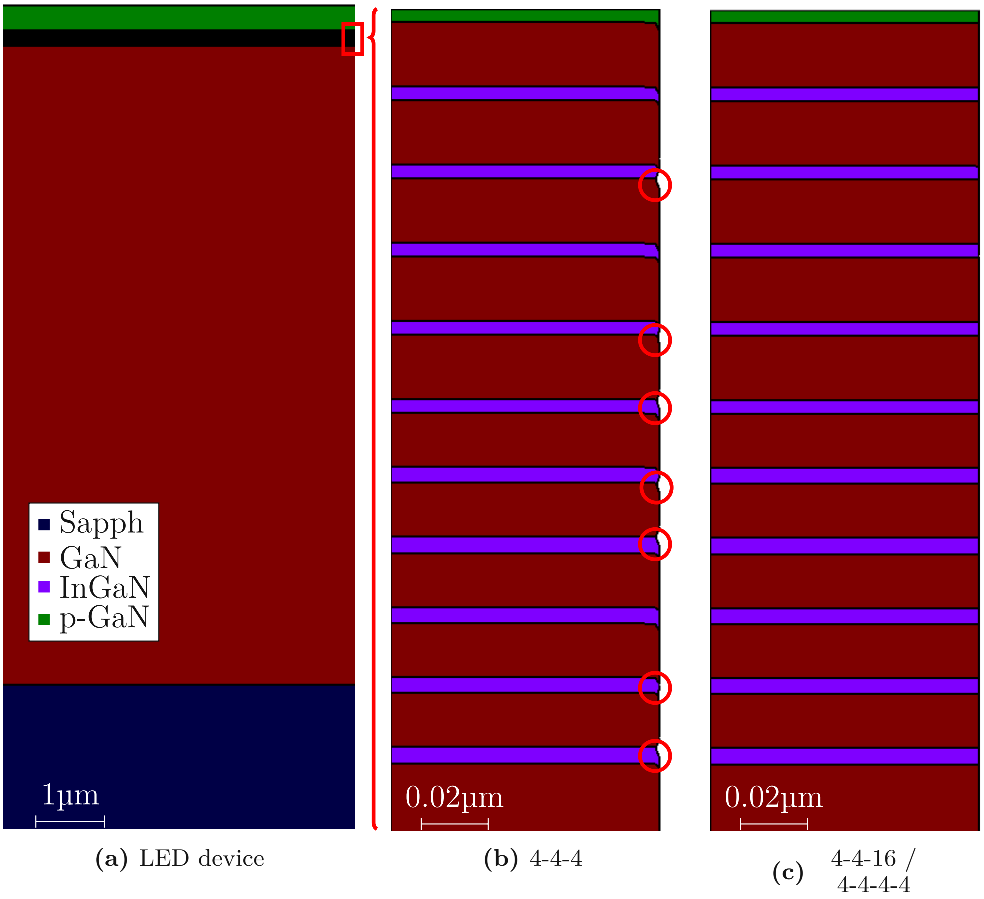
Four different configurations of the entire simulation flow are assessed in the following. To demonstrate the necessity of the previously calculated minimal required resolution, the first configuration utilizes a fixed refinement ratio R_{ratio} and a general refinement level of G_{ref} of 3 (4-4-4) (e.g., the finest sub-grid resolution is only 64 times finer than the base grid resolution). In the second and third simulation configurations, the previously calculated minimal required resolution is reached by refining the simulation domain with a general refinement level G_{ref} of 4 and a constant refinement ratio (4-4-4-4). These two configurations differ from each other in the chosen feature detection approach. The second configuration uses the benchmark hierarchical grid placement algorithm presented in Section 6.2 with geometric feature detection. For the third configuration the algorithm presented in Section 7.2.3 is utilized, excluding the dynamic adaptation of the grid resolution of the final sub-grid. The final configuration uses the entire algorithm presented in Section 7.2.3 utilizing a general refinement level of 2 with a constant refinement ratio and a concluding sub-grid with a 16 times finer resolution (4-4-16).

7.3.2 Discussion

Figure 7.7 depicts the entire LED device after the etching process step and zoomed-in versions of the thin material layers (active region). It can clearly be seen (see the visible kinks in Figure 7.7b) that using a 4-4-4 refinement is not sufficient, due to its too coarse final resolution, to properly resolve the thin InGaN material layers after the etching simulation. Thus, the following discussion focuses on refinement configurations that reach the minimum required resolution. Using the 4-4-16 and the 4-4-4-4 refinement produces the same final topography after the etching step (see Figure 7.7c).

The run-times for the etching process step are reported in Table 7.1. The simulation run with the hierarchical grid placement algorithm based on geometric features is the slowest. This is explained by the fact that it has to calculate the entire Boolean operation for each refinement level used. These observations are confirmed by the faster simulation time of the experiment using the 4-4-4-4 refinement and the in this section presented algorithm. Furthermore, when the hierarchical grid placement algorithm for Boolean operations is allowed to dynamically set the final sub-grid refinement level, it achieves a three times faster run-time than the hierarchical grid placement algorithm based on geometric features. On the one hand, this speedup is achieved by only calculating the Boolean operation once. On the other hand, this approach reduces the number of placed sub-grids since one entire grid level is skipped due to the dynamic refinement parameter.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



(a) LED device **(b)** 4-4-4 **(c)** 4-4-16 / 4-4-4-4

Figure 7.7: LED device: (a) entire device, (b) active region with 3 grid levels (fixed refinement ratio) – red circles highlight kinks resulting from the inadequate, low resolution, (c) active region with 4 grid levels. Adapted from Lenz et al., *Solid State Electron.* 200, (2023), p. 108534 [150], © The Authors, licensed under the CC BY-NC-ND 4.0 License, `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

Table 7.1: Etching simulation run-times based on the ICS for different refinement level configurations.

Feature Detection Method	Refinement Ratios	Run-Time
Our method	4-4-16	4 min 22 s
Our method	4-4-4-4	8 min 45 s
Benchmark, geometrical	4-4-4-4	11 min 45 s

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.4 Summary

A hierarchical grid placement algorithm for thin material layers affected by Boolean operations (i.e., etching simulations) has been presented. The algorithm automatically determines the thickness of the material layers affected by the Boolean operation and calculates the required refinement level based on the minimal amount of grid points that should be used to represent a material layer. The ability to consider the material layer thickness allows the algorithm to prevent the formation of numerical artifacts that occur due to an insufficient grid resolution before the Boolean operation is executed. This algorithm complements the general hierarchical grid placement algorithm presented in Section 6.2.

Moreover, the run-time of the Boolean operation is improved as a result of the ability of the algorithm to determine the minimum necessary refinement dynamically. Thus, the algorithm is able to avoid the formation of additional sub-grids. Therefore, the algorithm has a two times faster run-time when using a dynamic refinement level and a three times faster run-time than the benchmarking algorithm that utilizes geometric features of the topography for feature detection.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

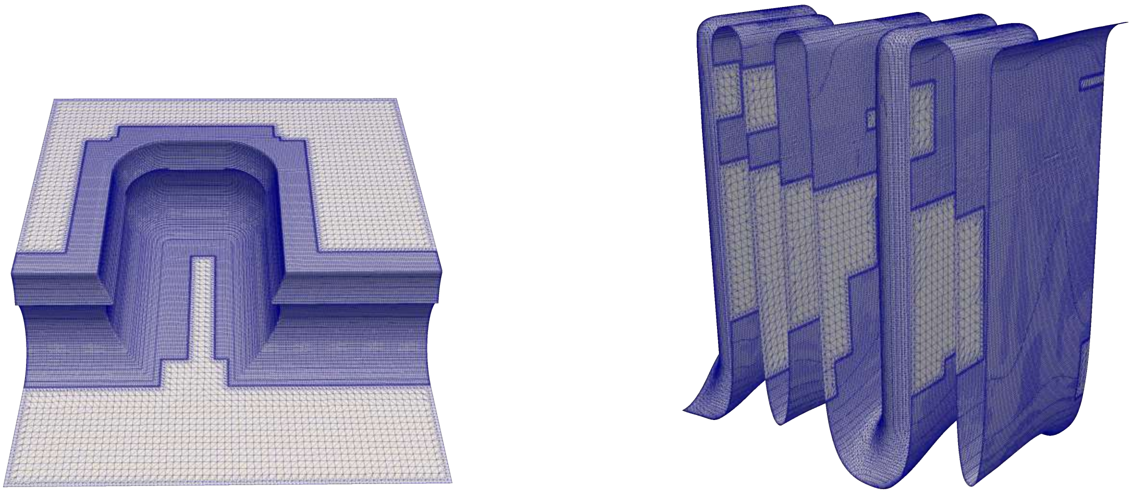
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Chapter 8

Surface Mesh Simplification for Efficient Top Down Flux Calculation

A crucial step during a process TCAD simulation workflow is the estimation of the surface flux (see Section 4.2): Several process models require top-down ray tracing to efficiently calculate the surface flux used to calculate the velocity field. The flux calculation takes up a significant portion of the overall run-time of a process TCAD simulation [103]. Therefore, methods that reduce the run-time of the flux calculation are required to improve overall simulation performance. One attractive approach is to optimize the surface meshes needed for ray tracing. However, the challenge is that the surface meshes extracted from implicit level set representations tend to be vastly inefficient due to unnecessarily dense meshes (large number of mesh elements, i.e., triangles, in geometrically irrelevant areas of the mesh) if left untreated.

Figure 8.1 shows two example geometries extracted from a process TCAD simulation utilizing hierarchical grids and a marching cubes algorithm.



(a) Surface 1: 70831 vertices.

(b) Surface 2: 175550 vertices

Figure 8.1: Examples of surface meshes extracted during a process TCAD simulation workflow.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

These figures illustrate that the marching cubes algorithm generates a considerable amount of triangles in flat regions of the geometry, which stem from the discretization of the level-set function on the grid. Obviously, improving the performance of any algorithm operating on such dense surface meshes, e.g., ray tracing, the number of mesh elements must be drastically reduced whilst preserving the features of the geometry.

Reducing the number of elements of a surface mesh is widely studied in computer graphics and is often referred to as *surface mesh simplification*. Several algorithms exist that simplify surface meshes with respect to a given metric [153, 154, 155, 156]. On the one hand, these algorithms operate in such a way that all triangles have approximately the same size [153, 154], thus, homogeneously simplifying the surface mesh. On the other hand, these algorithms use computationally expensive metrics during the simplification process [155, 156]. Both of these strategies have drawbacks when applied during a process TCAD simulation. A homogenous simplification of the entire surface may lead to the degradation of topographical features when a significant amount of surface elements is removed. Although a simplified surface mesh using an algorithm with an expensive metric may result in a suitable representation of the topography, such an approach is prohibitive since the surface flux calculation has to be performed in every time step. These constraints lead to the following consideration: To efficiently simplify a surface mesh extracted with the marching cubes algorithm, a computationally performant simplification strategy is required that is aware of the previously indicated feature of the geometry. Therefore, this chapter introduces a combination of the feature detection algorithm introduced in Section 5.1.2 with a surface mesh simplification algorithm using a computationally efficient metric (e.g., the Lindstrom-Turk algorithm [153]) to realize a so-called region simplification algorithm. The developed algorithm is evaluated using problem cases originating from process TCAD simulations, underlining practical application.

Section 8.1 introduces the *edge collapse* and Lindstrom-Turk algorithm, which are the fundamental algorithms used for mesh simplification in this work. Next, the developed region simplification algorithm is introduced (Section 8.2), which simplifies features of the geometry to a lower degree than non-features, allowing for a significantly improved balance between number of mesh elements and feature preservation. In the last section of this chapter (Section 8.3) the performance of the region simplification algorithm is evaluated. The geometric error introduced by the simplification is analyzed by calculating the Hausdorff-distance between the original and simplified surface meshes. Furthermore, the region simplification and the ray tracing run-times are evaluated.

Own Contributions

The key contribution presented in this chapter is the feature aware region simplification algorithm. This work was presented at the SCEE 2020 conference [157] and published as a book chapter in Scientific Computing in Electrical Engineering [158].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

8.1 Surface Mesh Simplification

This section introduces the underlying, key algorithms used for the developed region simplification algorithm presented in Section 8.2. First, the edge collapse algorithm is presented, which provides the basic mechanism to remove elements from the mesh. Next, the Lindstrom-Turk algorithm is discussed, which provides the basic metrics to decide which mesh elements should be removed.

8.1.1 Edge Collapse Algorithm

As previously mentioned, the utilized algorithm for surface mesh simplification is the edge collapse algorithm [159]. This algorithm requires three functions: First, a weight function that determines which edge should be removed. Second, a placement function that determines a point in space where a new vertex is placed after an edge is removed from the surface mesh. Third, a termination function which stops the simplification process.

The algorithm starts by calculating the weight of all edges (see Section 8.1.2) in the surface mesh and storing them in a minimum heap $\hbar$. Afterwards, the edge with the lowest weight is chosen; this edge is removed from the surface mesh and replaced by a vertex. For the new vertex, a position is calculated (see Section 8.1.2), and all edges that were previously connected to the vertices of the removed edge are connected to the newly created vertex. Figure 8.2 shows an illustration of this process. Next, the weights of the affected edges are recalculated and put into the heap $\hbar$. This process is repeated until the termination function ends the simplification process. Each collapsed edge removes one vertex, two triangles, and three edges from the surface mesh. In this work, the following two termination criteria are used:

- If the edge length of the to be removed edge is larger than a user supplied edge length ($l_{\max}$).
- If no edges have been removed in the previous simplification step.

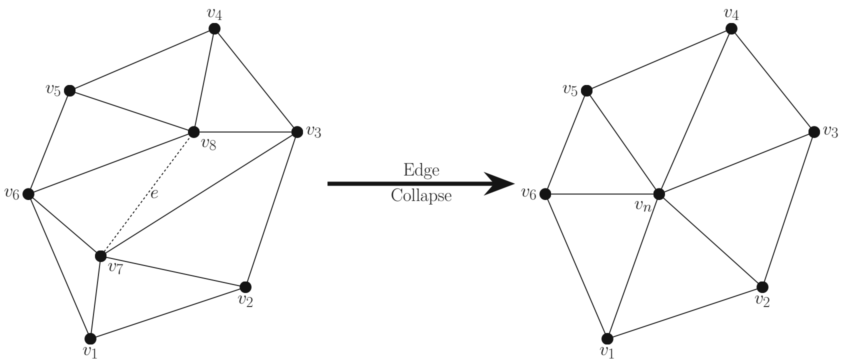


Figure 8.2: Illustration of an edge collapse on a surface mesh. The edge e is removed from the surface mesh and replaced by the vertex v_n. All edges connected to the removed edge are connected to the new vertex.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

8.1.2 Lindstrom-Turk Algorithm

The method used to calculate the weights of the edges and the position of the newly placed vertices is provided by the Lindstrom-Turk algorithm [153]. This algorithm is based on four constraints that aim to minimize the change in the volume characterized by the surface mesh. These four constraints are *boundary preservation*, *volume preservation*, *volume optimization*, and *triangle shape optimization*. Depending on the examined edge, three of the four constraints are chosen. Each constraint can then be interpreted as a plane in $\mathbb{R}^3$, which is linearly independent of the planes of the other constraints. The intersection point of these three planes describes the optimal position (i.e., the minimal change in volume after the edge is removed) of the to be placed vertex, and the weight of the edge is determined by the weighted (a user supplied parameter) sum of all optimization parameters.

The Lindstrom-Turk algorithm allows for an efficient calculation of the weights of the edges as well as the position of the new vertex. Furthermore, the Lindstrom-Turk algorithm takes the quality of the newly created triangle into consideration (i.e., aiming to create equilateral triangles).

8.2 Region Simplification Algorithm

The region simplification algorithms start by analyzing the features of the surface mesh, which are used to split the surface mesh into two disjunct so-called feature and transition regions, as will be discussed in detail below. These regions can then be simplified using different strategies, allowing for a higher resolution at features of the geometry. In this work, the Lindstrom-Turk algorithm with different parameter sets is used to guide the simplification process in both regions of the surface mesh, however, the region simplification algorithm is not limited to the Lindstrom-Turk algorithm. Thus, other simplification methods can be used to simplify the transition and feature region respectively.

This section starts by discussing the feature detection strategy. Next, it is described how the detected features are used to split the mesh into regions. Finally, the simplification of the different regions and a strategy to create a smooth transition from smaller to bigger mesh elements is discussed.

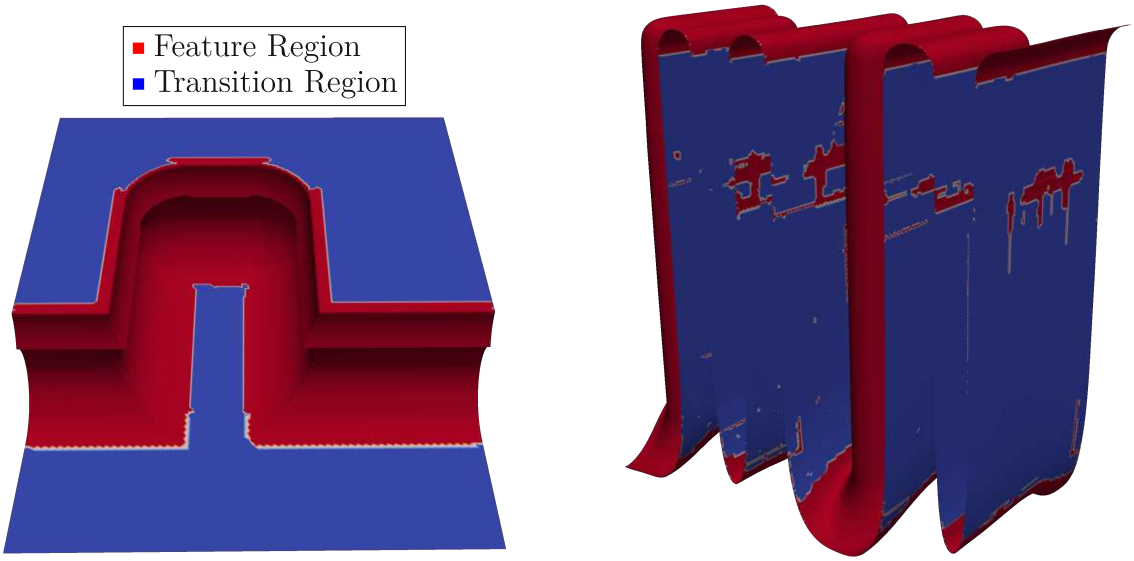
8.2.1 Feature Detection

The simplification algorithm starts by performing a feature detection on the entire surface mesh. This is achieved by using the feature detection algorithm from Section 5.1.2 and applying it to the vertices of the surface mesh. Furthermore, the mean (H) and Gaussian curvature (K) for the vertices are calculated with the curvature calculation methods presented in Section 3.3. The feature detection algorithm, with a feature detection parameter of $C = 0.5$, is used to classify each vertex in the surface mesh as either a *flat* or a *feature* vertex. A vertex is considered to be a feature vertex if it is detected as a feature and flat otherwise.

An illustration of the detected features on the two surface meshes introduced in Figure 8.1 is shown in Figure 8.3.

8.2.2 Mesh Partitioning and Extension of Regions

As previously mentioned, the detected features are used to partition the surface mesh into regions, the *feature regions* (i.e., feature vertices) and the *transition regions* (i.e., flat vertices). The partitioning of the mesh allows the surface mesh simplification algorithm to simplify the transition region to a greater extent. Thus, removing more mesh elements from flat parts of the geometry while maintaining a higher resolution at features of the surface mesh. However, the straightforward approach of only simplifying triangles in the transition region up to a fixed edge length produces a significant number of low quality triangles (see Section 2.2), as can be seen in Figure 8.4. Therefore, the region simplification utilizes a mesh grading that linearly increases the edge length of the triangles the further they reach into the transition region, to minimize the number of low quality triangles. This is achieved in the following way: After the feature region and the transition region are identified, the iterative simplification algorithm starts by simplifying the entire surface mesh, including the feature region, with a minimum edge length of l_0. In the case that the feature region should not be simplified, l_0 is set to 0. The subsequent simplification step starts at the border between the transition and the feature region. The transition region is simplified until a minimal edge length of $l_1 = l_0 + sl$, where sl denotes a user specified step length, is reached. Next, the feature region is extended into the transition region (i.e., moved) by adding all vertices that are adjacent (e.g., connected by an edge) to the feature region. The contracted transition region is again simplified until a minimal edge length of $l_{i+1} = l_i + sl$ with $i \in \{0, 1, \ldots, n \in \mathbb{N}\}$ is reached.



(a) Surface 1 **(b)** Surface 2

Figure 8.3: Detected features of the surface meshes from Figure 8.1.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

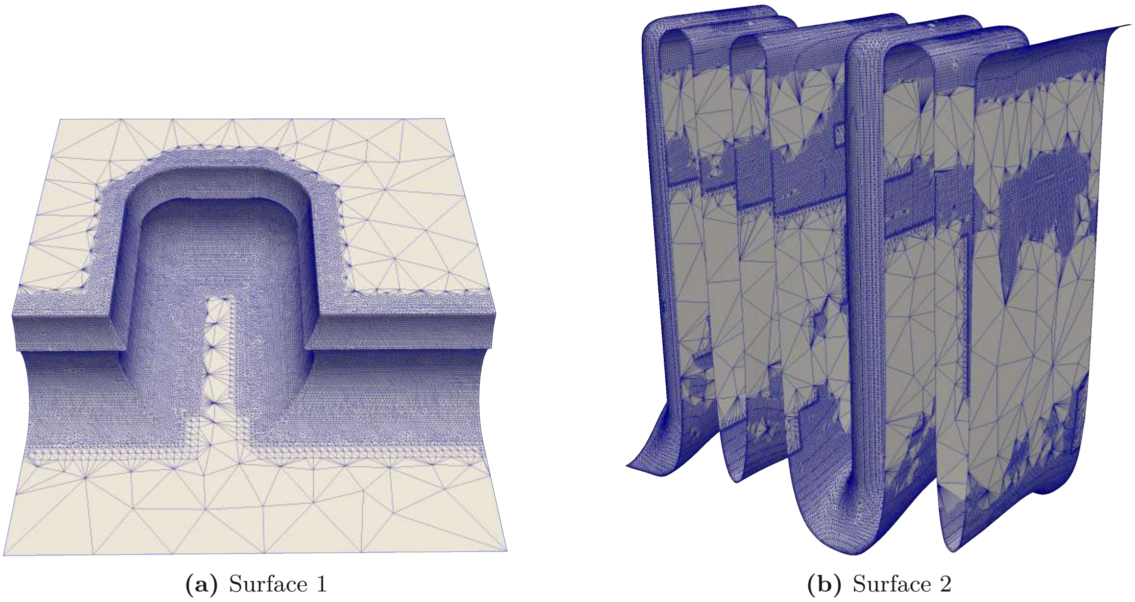

(a) Surface 1 (b) Surface 2

Figure 8.4: Surface meshes where only the transition region has been simplified. The not simplified elements in the upper third of Surface 2 are a consequence of the feature detection (see Figure 8.3) and the Lindstrom-Turk algorithm assigning large weights to the edges between the features.

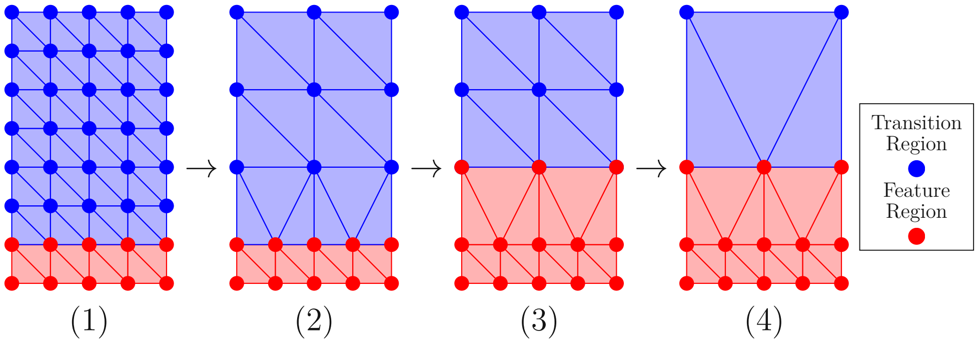


Figure 8.5: Example of the simplification process: (1) shows the mesh after it has been divided into regions. (2) shows the simplification of the transition region. (3) shows the extension of the feature region. (4) shows again the simplification of the transition region with an increased edge length considered for the simplification of the transition region. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.

These last two steps continue until one of the two termination conditions introduced in Section 8.1.1 is fulfilled, thus, terminating the simplification process. Figure 8.5 illustrates the above discussed extension and, subsequent, simplification of the regions. Furthermore, Figure 8.6 depicts a flow chart of the simplification process.

To avoid the generation of too large triangles in this iterative scheme, the previously mentioned parameter $l_{\max}$ can be chosen accordingly. This terminates the simplification process when the minimal edge length in the transition region l_i has reached an edge length equal or larger than $l_{\max}$.

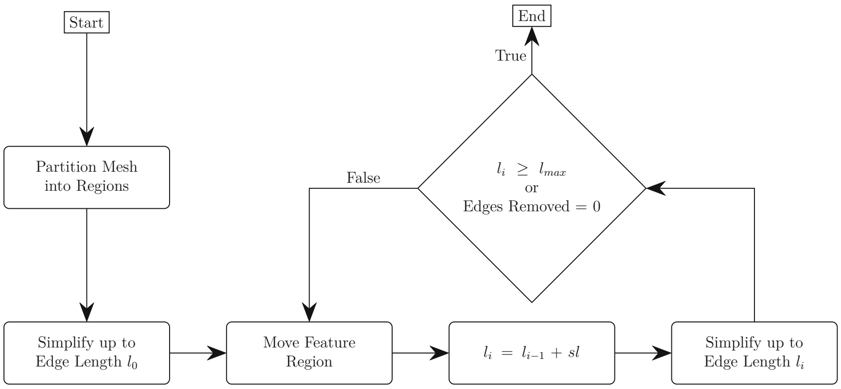

Start
Partition Mesh into Regions
Simplify up to Edge Length l_0
Move Feature Region
$l_i = l_{i-1} + sl$
Simplify up to Edge Length l_i
$l_i \geq l_{max}$ or Edges Removed = 0
False
True
End

Figure 8.6: Flow chart of the region simplification algorithm.

In a surface mesh extracted from a level-set function, the parameter l_0 can be chosen in concordance with the smallest sub-grid resolution Δx. For other surface meshes, the parameter l_0 can be chosen by averaging the length of all edges in the feature region. Furthermore, empirically it has been observed that the step length sl should not be chosen bigger than l_0: Although the amount of mesh elements would be reduced by choosing a larger parameter, the triangle quality of the surface mesh suffers.

8.3 Comparison and Evaluation

The region simplification algorithm is evaluated with respect to three metrics:

- The distance between the original and simplified surface mesh (i.e., the error introduced by the simplification).
- The run-time of the simplification process.
- The run-time of surface flux calculations using Monte Carlo ray tracing.

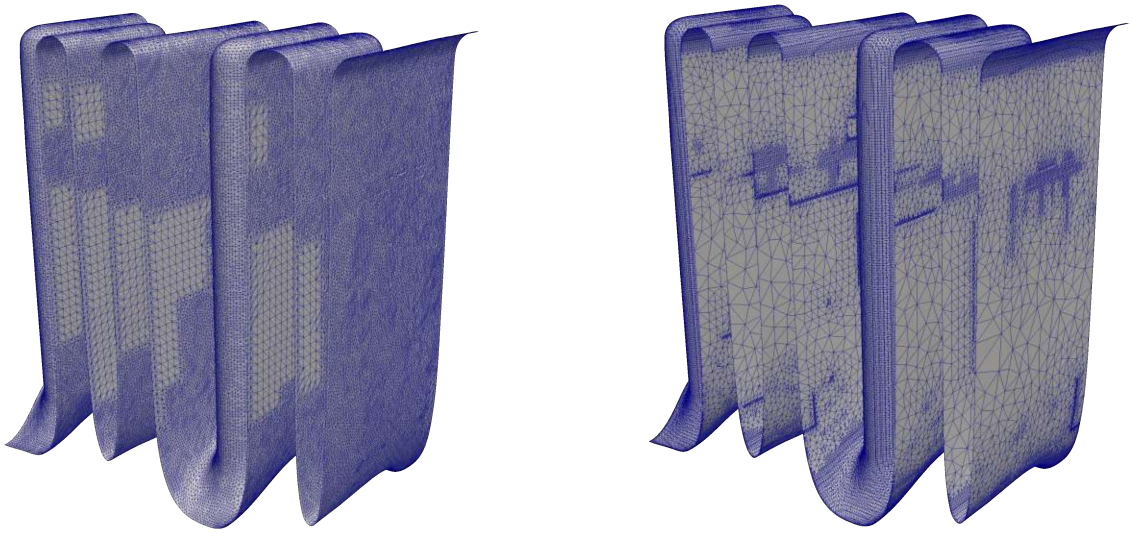
To that end, the two example meshes (see Figure 8.1) are simplified using the region simplification algorithm and the Lindstrom-Turk algorithm. Each of the example meshes is simplified using eight different parameter sets, which reduced the number of vertices in the mesh by 20 to 90%. An exemplary, comparative result is shown in Figure 8.7. This comparison illustrates that the region simplification algorithm primarily removes triangles in the flat region and keeps a higher resolution at the features of the geometry. The surface meshes have been simplified using ViennaMesh and CGAL and were executed on the benchmark platform Workstation 1 (see Section 4.6.3).

8.3.1 Distance to Original Geometry

The simplification process introduces minor errors in the discretization of the surface. Although the Lindstrom-Turk algorithm tries to minimize this error, it is unavoidable since each edge collapse removes information about the surface. Thus, a metric is required that is able to capture the introduced error and allows to compare the two algorithms.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Simplification algorithm: Lindstrom-Turk.

(b) Simplification algorithm: Region simplification.

Figure 8.7: Example of a simplified mesh with the two discussed simplification strategies. The same number of vertices has been removed from both surface meshes. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.

A commonly used metric that calculates the distance between two sets (e.g., surface meshes) is the *Hausdorff distance* [160]. The Hausdorff distance is defined as follows [161]:

8.3.1 Definition (Hausdorff Distance) Let $X, Y \subset \mathbb{R}^3$ be two sets. Then the *Hausdorff distance* is defined as:

$$d_H(X,Y) = \max(d_{H'}(X,Y), d_{H'}(Y,X)).$$

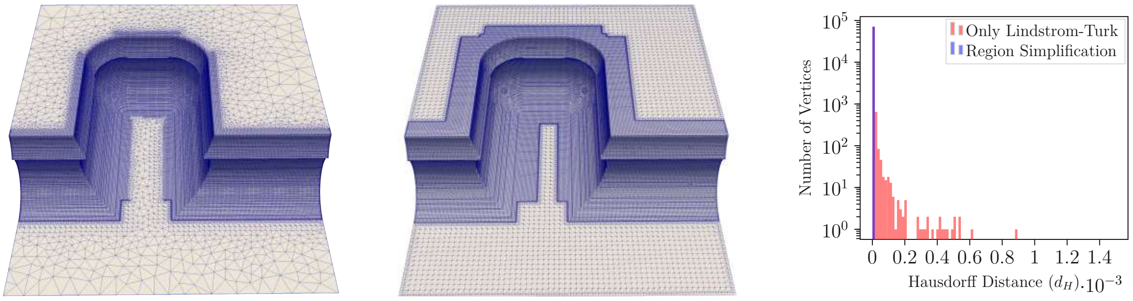
$d_{H'}(X,Y)$ stands for the *one-sided Hausdorff distance*:

8.3.2 Definition (One-Sided Hausdorff Distance) Let $X, Y \subset \mathbb{R}^3$ be two sets. Then the *one-sided Hausdorff distance* is defined as follows:

$$d_{H'}(X,Y) := \max_{\mathbf{p}\in X}(\min_{\mathbf{q}\in Y}(\|\mathbf{p}-\mathbf{q}\|)).$$

In the case of surface meshes, the Hausdorff distance is measured from the vertices of one surface mesh to the closest triangle intersection of the other surface mesh [161]. The error introduced by the simplification process is determined by calculating the Hausdorff distance from the original geometry (e.g., the not simplified surface mesh) to the simplified surface mesh.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

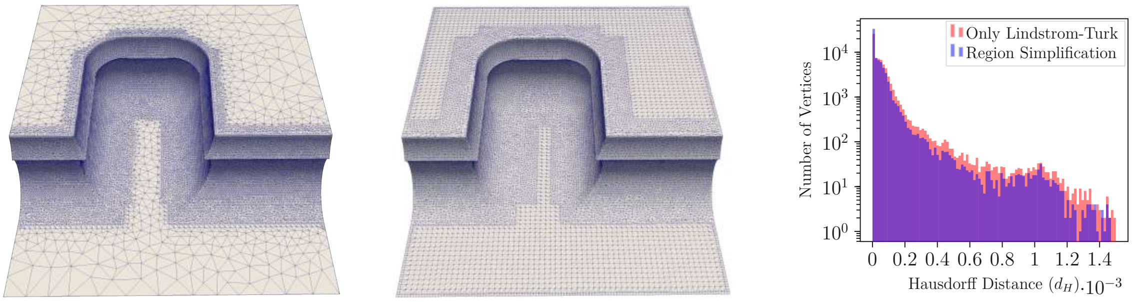


(a) Simplified surface mesh (region simplification).

(b) Simplified surface mesh (Lindstrom-Turk).

(c) Hausdorff distances from the original geometry.

Figure 8.8: Simplified Surface 1 with the parameters $l_0 = 0.0$ and $s_l = 0.01$. The meshes have been simplified to have the same number of vertices.



(a) Simplified surface mesh (region simplification).

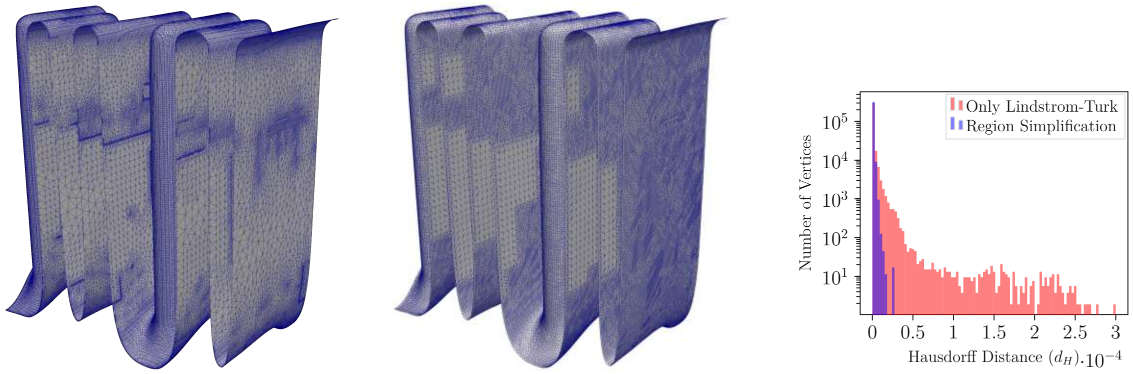
(b) Simplified surface mesh (Lindstrom-Turk).

(c) Hausdorff distances from the original geometry.

Figure 8.9: Simplified Surface 1 with the parameters $l_0 = 0.019$ and $s_l = 0.018$. The meshes have been simplified to have the same number of vertices. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.

Figure 8.8 and Figure 8.9 show the simplified Surface 1 and the Hausdorff distances to the original geometry. The Hausdorff distances reported in Figure 8.8c show that there is no geometric error introduced into the simplified geometry, since the region simplification algorithm does not simplify the feature region (i.e., $l_0 = 0$). Furthermore, this indicates that the feature detection is able to detect all features of the surface mesh correctly and thus only simplifies triangles in flat parts of the surface mesh, which does not alter the geometry represented by the surface mesh. On the other hand, when the feature region is simplified, to achieve an overall higher degree of simplification, geometric errors are introduced into the simplified surface mesh, as can be seen in Figure 8.9c. The smaller geometric error is a consequence of the feature region being simplified to a lesser degree and more elements are removed from the flat region when using the region simplification algorithm compared to the Lindstrom-Turk algorithm. Nevertheless, the surface mesh simplified with the region simplification algorithm has a smaller error than the one simplified with the Lindstrom-Turk algorithm.

Figure 8.10 and Figure 8.11 again show the Hausdorff distances to the original geometry and the simplified surface mesh of Surface 2. First, it has to be mentioned that the average Hausdorff distance of Surface 2 is one order of magnitude smaller than that of Surface 1.

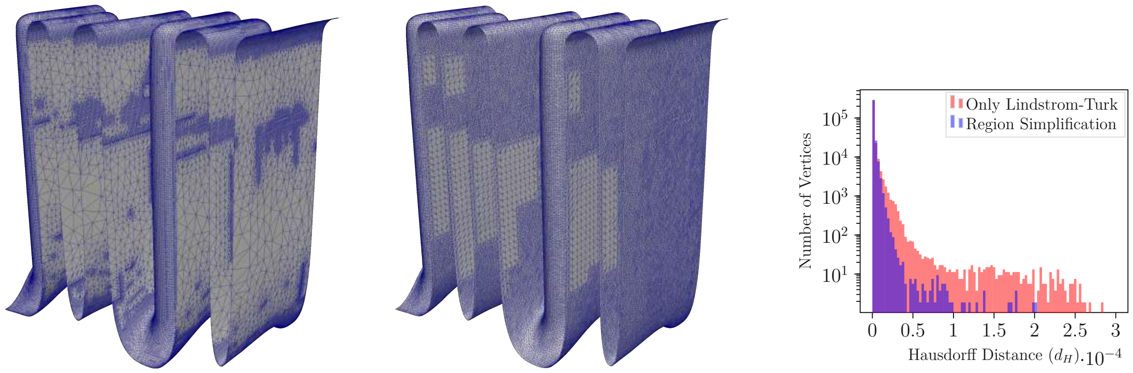


(a) Simplified surface mesh (region simplification).

(b) Simplified surface mesh (Lindstrom-Turk).

(c) Hausdorff distances from the original geometry.

Figure 8.10: Simplified Surface 2 with the parameters $l_0 = 0.0$ and $s_l = 0.0006$. The meshes have been simplified to have the same number of vertices.



(a) Simplified surface mesh (region simplification).

(b) Simplified surface mesh (Lindstrom-Turk).

(c) Hausdorff distances from the original geometry.

Figure 8.11: Simplified Surface 2 with the parameters $l_0 = 0.0014$ and $s_l = 0.0008$. The meshes have been simplified to have the same number of vertices. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.

The reason for that difference is the base grid resolution of the underlining simulation from which the two meshes where extracted. In Figure 8.10c, a small error in the Hausdorff distance can be seen, even though the feature region is not simplified. This error occurs since Surface 2 has finer features than Surface 1, which are not captured by the feature detection algorithm. However, in general, a small error in the feature detection is to be expected since a feature detection parameter has to be used (see Section 5.1), which is going to miss features smaller than the feature detection parameter. Additionally, the surface mesh simplified with region simplification has a far smaller error than the mesh simplified with the Lindstrom-Turk algorithm, since it removes more mesh elements from the transition region than from the feature region, which is further confirmed in Figure 8.11c for the case of simplifying the feature region.

On average region simplification has a 20 - 40 % lower Hausdorff distance to the original geometry than using the Lindstrom-Turk algorithm.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

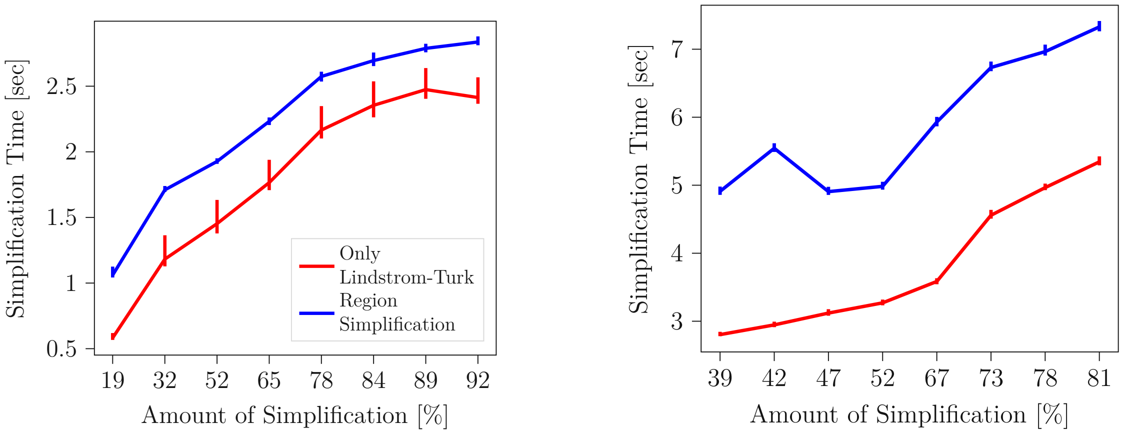
Next, the simplification run-times of the two simplification algorithms are investigated to study the computational effort of both simplification algorithms.

8.3.2 Simplification Run-Time

The simplification run-times of the example meshes using the different parameters are reported in Figure 8.12. As expected, it is shown that region simplification introduces an overhead into the simplification process compared to the Lindstrom-Turk algorithm. The main reason is the feature detection used to split the surface mesh into regions. Compared to the Lindstrom-Turk algorithm, region simplification has, on average, a 17 % longer simplification time. However, when considering entire process TCAD workflows (see Section 4.5), the additional time spent on improved surface mesh simplification easily pays of with costly subsequent process simulation steps, such as ray tracing for flux calculations, as significantly less triangles have to be processed.

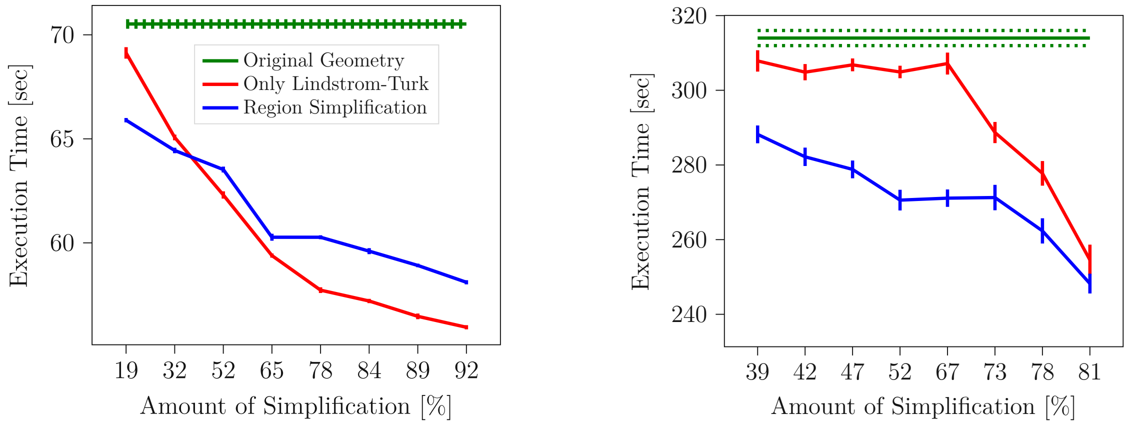
8.3.3 Flux Calculation and Monte Carlo Ray Tracing Run-Time

The run-times of flux calculations using Monte Carlo ray tracing based on the simplified example meshes (see Figure 8.1) are shown in Figure 8.13. It is shown that the speedup of the ray tracing also depends on the underlying geometry. For geometries that represent deep trenches (i.e., Surface 2), the speedup achieved by the surface mesh simplification is more dependent on the size of the triangles than for geometries with smaller horizontal variation(i.e., Surface 1). These differences are explained by the difference in the bounding volume hierarchy (i.e., the data structure used for ray tracing), which can be traversed faster in a deep trench with bigger elements.



(a) Surface 1 **(b)** Surface 2

Figure 8.12: Surface mesh simplification run-times with different degrees of simplification. The degree of simplification denotes the number of vertices that have been removed. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.



(a) Surface 1

(b) Surface 2

Figure 8.13: Execution time of Monte Carlo ray tracing using 10^8 rays. The amount of simplification denotes the number of vertices that have been removed from the original mesh. Reproduced with permission from Springer Nature from Lenz et al., *Math Indust.* (2021), pp. 73-81 [158], © 2020, under exclusive license to Springer Nature Switzerland AG.

When tracing Surface 1, the simplified surface meshes improve the run-times of the flux calculation. Furthermore, both simplification methods perform similarly to each other. The run-times of ray tracing performed on Surface 2 are also improved by the surface mesh simplification. Additionally, the surface meshes simplified with the region simplification algorithm outperform the meshes simplified with the Lindstrom-Turk algorithm. The biggest ray tracing performance differences between the two algorithms is about 12 % at the simplification levels between 52 and 67 %.

Comparing the run-times for the surface mesh simplification (see Figure 8.12) and the ray tracing (see Figure 8.13) shows that the speedup achieved for the ray tracing on the simplified surface meshes is bigger than the cost of simplifying the surface mesh, confirming the previously stated justification in Section 8.3.2.

8.4 Summary

A new surface mesh simplification algorithm has been presented. The algorithm uses the features of a surface mesh to divide it into disjunct regions that are simplified with different parameters. The algorithm is well suited for surface meshes originating from process TCAD simulations used for surface flux calculations. The surface mesh simplification algorithm has been evaluated by investigating the distance to the original geometry, the run-times of the surface mesh simplification, and a subsequent flux calculation step using Monte Carlo ray tracing. Surface meshes simplified with the proposed surface mesh simplification algorithm have an improved geometric distance to the original surface mesh compared to a surface mesh simplified with the reference algorithm. The average Hausdorff distance of the investigated geometries is improved by 20 - 40 %. On average, the ray tracing performance on the simplified surface meshes is improved by 15 %. Additionally, geometries from process TCAD simulations with deep trenches, simplified with the region simplification algorithm, perform much better in ray tracing.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The time spent on simplification is, on average, 17 % slower when using region simplification compared to the reference algorithm. Therefore, when considering the entire process of flux calculation, the acceleration of the ray tracing due to the simplified surface exceeds the time spent on surface mesh simplification, underlying the importance of surface mesh simplification in relevant process TCAD workflows.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 9

Summary and Outlook

This thesis introduces a set of newly developed geometry-aware algorithms for hierarchical grids which are centered around identifying and utilizing the discrete surface curvatures of topographies arising in semiconductor process TCAD simulations to optimize computational processing. These geometry-aware algorithms significantly increase the performance of topography simulations by selectively refining or simplifying the discrete representation of the device topography during a simulation.

The three most prominently used discrete surface representations during topography simulations were introduced; level-set functions, surface meshes, and point clouds. Furthermore, the most common ways of switching between these surface representations and their role during topography simulations were discussed. The primary numerical methods used during a topography simulation were considered: The representation of materials on the wafer surface, the evolution of these materials in time (i.e., the level-set method), and three strategies for estimating the surface flux. These discussed methods were then combined into a general workflow for topography simulations. Finally, the concept of the surface curvature on continuous surfaces was discussed, as well as several strategies of how to use this concept on the previously discussed discrete surface representations.

The surface curvatures of the discretized surfaces were used to formulate an automatic feature detection algorithm which detects parts of a discrete surface with significant geometric variation. For 3D level-set functions three methods from the literature and a novel extension of the standard calculation method of the surface curvatures have been investigated for their applicability in topography simulations. Two methods stood out, depending on the quality requirements of the feature detection. For performance oriented applications the *Shape Operator* method is superior to all other methods. This method uses the smallest finite difference stencil to calculate the mean curvature of the level-set function, while avoiding the calculation of the Gaussian curvature for a robust feature detection. The second method is the novel *Big Stencil* method, which has a similar computational performance to the other tested methods, yet it has a higher numerical accuracy and is less susceptible to numerical noise. Additionally, a feature detection parameter for topography simulations has been obtained through a parameter study performed on typical device topographies.

TU Bibliothek
WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

The feature detection algorithm and feature detection parameter were used to guide a hierarchical grid placement algorithm to refine the simulation domains of topography simulations. Due to the detected features of the device topography, the hierarchical grid placement algorithm was able to precisely place sub-grids at parts of the simulation domain that improve the discrete description of the topography, while minimizing impacts on simulation performance. This hierarchical approach has been used to simulate selective epitaxial growth of SiGe crystals, which leads to an improvement in computation time, while maintaining an accurate description of the crystal surface.

Furthermore, the feature detection algorithm has been used to improve Monte Carlo ray tracing based surface flux calculations on surface meshes. The detected features have been used to split the surface mesh into two separate regions which are used to guide a surface mesh simplification algorithm. Depending on the previously calculated regions, the surface mesh simplification algorithm is able to remove more or less triangles from the original surface mesh. Additionally, the quality of the triangles between the regions is taken into consideration to create a steady increase in the size of the triangles to prevent the formation of bad mesh elements. This approach maximizes the amount of triangles that are removed from the surface mesh, while maintaining a detailed description of its features.

A specially designed feature detection algorithm for etching simulations of thin material layers utilizing Boolean operations has been developed. This algorithm analyzes the thickness of the material layers that are affected by an etching simulation and determines a minimal required refinement level. Thus, it prevents the formation of numerical artifacts as a consequence of a too coarse resolution of the simulation domain. The computational performance of the algorithm is further improved by dynamically increasing the resolution of the final sub-grid to reach the previously determined minimal required refinement level.

Some possible, future extensions of the geometry-aware algorithms introduced in this work for topography simulations are discussed in the following paragraphs. Monte Carlo ray tracing based surface flux calculations introduce numerical noise into the discrete surface description. This noise prevents more straightforward implementations of feature detection strategies from accurately detecting the features of the surface. The *Big Stencil* method is able to ignore surface noise introduced by the process model and the finite difference scheme used to solve the level-set equation. Thus, the *Big Stencil* method could be able to only detect features of the topography and ignore the noise from Monte Carlo based simulations. Furthermore, finite difference schemes with even bigger finite difference stencils could be investigated, which may lead to a more reliable feature detection on surfaces with noise.

The introduced feature detection algorithm can be used to speed up Monte Carlo ray tracing based surface flux calculations on point clouds. In this case, the features of the device topography can be detected with the help of the implicitly defined level-set function, thus redistributing the expensive curvature calculations on point clouds to their computationally cheaper calculations on level-set functions.

The detected features could then be further used to simplify the point cloud during its extraction from the level-set function.

The initial motivation of the feature detection algorithm and hierarchical grid placement algorithm described in this work was to improve simulation performance of topography simulations by selectively refining the simulation domain at features of the topography. Clearly, the feature detection and hierarchical grid placement steps introduce an overhead into a topography simulation, which is evidently small enough to improve simulation performance. However, it is possible that for particularly complex topographies the amount of required sub-grids is so high that the overhead of the hierarchical approach exceeds the performance gains. Thus, it can be of interest to develop a heuristic that determines if a simulation should use a certain amount of grid levels and sub-grids or use a higher base grid resolution with fewer grid-levels.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] S. Hofstein and F. Heiman. "The Silicon Insulated-Gate Field-Effect Transistor". In: *Proceedings of the IEEE* 51.9 (1963), pp. 1190–1202. DOI: 10.1109/PROC.1963.2488.

[2] Y. Yasuda-Masuoka, J. Jeong, K. Son, S. Lee, S. Park, Y. Lee, J. Youn Kim, J. Lee, M. Cho, S. Lee, S. Hong, H. Hong, Y. Jung, C. Yoon, Y. Ko, K. Jung, T. Myung, J. M. Youn, and G. Jeong. "High Performance 4nm FinFET Platform (4LPE) with Novel Advanced Transistor Level DTCO for Dual-CPP/HP-HD Standard Cells". In: *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*. 2021, pp. 13.3.1–13.3.4. DOI: 10.1109/IEDM19574.2021.9720656.

[3] G. E. Moore. "Cramming More Components Onto Integrated Circuits, Reprinted From Electronics, Volume 38, Number 8, April 19, 1965, pp.114 ff." In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 33–35. DOI: 10.1109/N-SSC.2006.4785860.

[4] M. G. S. and S. Costas J. *Fundamentals of Semiconductor Manufacturing and Process Control*. John Wiley & Sons, 2006. DOI: 10.1002/0471790281.

[5] M. C. K. *Introducing Technology Computer-Aided Design (TCAD)*. 1st. John Wiley & Sons, 2017. DOI: 10.1201/9781315364506.

[6] A. Yanguas-Gil. *Growth and Transport in Nanostructured Materials: Reactive Transport in PVD, CVD, and ALD*. Springer, 2016. DOI: 10.1007/978-3-319-24672-7.

[7] L. M. A. and L. A. J. *Principles of Plasma Discharges and Materials Processing*. Jenny Stanford Publishing, 2005. DOI: 10.1002/0471724254.

[8] S. J. A. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[9] E. Chason, S. T. Picraux, J. M. Poate, J. O. Borland, M. I. Current, T. Diaz de la Rubia, D. J. Eaglesham, O. W. Holland, M. E. Law, C. W. Magee, J. W. Mayer, J. Melngailis, and A. F. Tasch. "Ion Beams in Silicon Processing and Characterization". In: *Journal of Applied Physics* 81.10 (1997), pp. 6513–6561. DOI: 10.1063/1.365193.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[10] K. K. Bhuwalka, H. Wu, W. Zhao, G. Rzepa, O. Baumgartner, F. Benistant, Y. Chen, and C. Liu. “Optimization and Benchmarking FinFETs and GAA Nanosheet Architectures at 3-nm Technology Node: Impact of Unique Boosters”. In: *IEEE Transactions on Electron Devices* 69.8 (2022), pp. 4088–4094. DOI: `10.1109/TED.2022.3178665`.

[11] H. Kwon, H. Huh, H. Seo, S. Han, I. Won, J. Sue, D. Oh, F. Iza, S. Lee, S. K. Park, and S. Cha. “TCAD Augmented Generative Adversarial Network for Hot-Spot Detection and Mask-Layout Optimization in a Large Area HARC Etching Process”. In: *Physics of Plasmas* 29.7 (2022), p. 073504. DOI: `10.1063/5.0093076`.

[12] X. Klemenschits, P. Manstetten, L. Filipovic, and S. Selberherr. “Process Simulation in the Browser: Porting ViennaTS using WebAssembly”. In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2019, pp. 339–342. DOI: `10.1109/SISPAD.2019.8870374`.

[13] X. Klemenschits. “Emulation and Simulation of Microelectronic Fabrication Processes”. Doctoral Dissertation. TU Wien, 2022.

[14] N. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. 1st. Springer, 2002. DOI: `10.1007/978-3-642-04074-0`.

[15] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Vol. 153. Springer, 2003. DOI: `10.1007/b98879`.

[16] P. Frey and L. George Paul. *Mesh Generation: Application to Finite Elements*. 2nd. Wiley-ISTE, 2013.

[17] X. Klemenschits, S. Selberherr, and L. Filipovic. “Modeling of Gate Stack Patterning for Advanced Technology Nodes: A Review”. In: *Micromachines* 9.12 (2018), p. 631. DOI: `10.3390/mi9120631`.

[18] P. Manstetten, J. Weinbub, A. Hössinger, and S. Selberherr. “Using Temporary Explicit Meshes for Direct Flux Calculation on Implicit Surfaces”. In: *Procedia Computer Science* 108 (2017), pp. 245–254. DOI: `10.1016/j.procs.2017.05.067`.

[19] M. J. Berger and J. Oliger. “Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations”. In: *Journal of Computational Physics* 53.3 (1984), pp. 484–512. DOI: `10.1016/0021-9991(84)90073-1`.

[20] B. Zönnchen and G. Köster. “A Parallel Generator for Sparse Unstructured Meshes to Solve the Eikonal Equation”. In: *Journal of Computational Science* 32 (2019), pp. 141–147. DOI: `10.1016/j.jocs.2018.09.009`.

[21] Å. Ervik, K. Y. Lervåg, and S. T. Munkejord. “A Robust Method For Calculating Interface Curvature and Normal Vectors Using an Extracted Local Level Set”. In: *Journal of Computational Physics* 257 (2014), pp. 259–277. DOI: `10.1016/j.jcp.2013.09.053`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[22] R. A. Trompert and J. G. Verwer. “A Static-Regridding Method for Two-Dimensional Parabolic Partial Differential Equations”. In: *Applied Numerical Mathematics* 8.1 (1991), pp. 65–90. DOI: 10.1016/0168-9274(91)90098-K.

[23] P. Lu and X. Xu. “A Robust Multilevel Preconditioner Based on a Domain Decomposition Method for the Helmholtz Equation”. In: *Journal of Scientific Computing* 81 (2019), pp. 291–311. DOI: 10.1007/s10915-019-01015-z.

[24] C. Wang, W. Wang, S. Pan, and F. Zhao. “A Local Curvature Based Adaptive Particle Level Set Method”. In: *Journal of Scientific Computing* 91 (2022). DOI: 10.1007/s10915-022-01772-4.

[25] M. P. d. Carmo. *Differential Geometry of Curves & Surfaces*. 2nd. Dover Publications, Inc., 2016.

[26] H. T. Ho and D. Gibbins. “Multi-Scale Feature Extraction for 3D Models using Local Surface Curvature”. In: *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2008, pp. 16–23. DOI: 10.1109/DICTA.2008.64.

[27] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas. “Voronoi-Based Curvature and Feature Estimation From Point Clouds”. In: *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), pp. 743–756. DOI: 10.1109/TVCG.2010.261.

[28] H. S. Kim, H. K. Choi, and K. H. Lee. “Feature Detection of Triangular Meshes Based on Tensor Voting Theory”. In: *CAD Computer Aided Design* 41 (2009), pp. 47–58. DOI: 10.1016/j.cad.2008.12.003.

[29] U. Clarenz, M. Rumpf, and A. Telea. “Robust Feature Detection and Local Classification for Surfaces Based on Moment Analysis”. In: *IEEE Transactions on Visualization and Computer Graphics* 10 (2004), pp. 516–524. DOI: 10.1109/TVCG.2004.34.

[30] L. F. Aguinsky. “Phenomenological Modeling of Reactive Single-Particle Transport in Semiconductor Processing”. Doctoral Dissertation. TU Wien, 2022. DOI: 10.34726/hss.2023.107502.

[31] P. Hong, Z. Zhao, J. Luo, Z. Xia, X. Su, L. Zhang, C. Li, and Z. Huo. “An Improved Dimensional Measurement Method of Staircase Patterns with Higher Precision in 3D NAND”. In: *IEEE Access* 8 (2020), pp. 140054–140061. DOI: 10.1109/ACCESS.2020.3012012.

[32] X. Zhou, P. Tian, C. W. Sher, J. Wu, H. Liu, R. Liu, and H. C. Kuo. “Growth, Transfer Printing and Colour Conversion Techniques Towards Full-Colour Micro-LED Display”. In: *Progress in Quantum Electronics* 71 (2020), p. 100263. DOI: 10.1016/j.pquantelec.2020.100263.

[33] O. Ertl and S. Selberherr. “Three-Dimensional Level Set Based Bosch Process Simulations using Ray Tracing for Flux Calculation”. In: *Microelectronic Engineering* 87.1 (2010), pp. 20–29. DOI: 10.1016/j.mee.2009.05.011.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[34] L. F. Aguinsky, F. Rodrigues, G. Wachter, M. Trupke, U. Schmid, A. Hössinger, and J. Weinbub. “Phenomenological Modeling of Low-Bias Sulfur Hexafluoride Plasma Etching of Silicon”. In: *Solid-State Electronics* 191 (2022), p. 108262. DOI: 10.1016/j.sse.2022.108262.

[35] T. Reiter, X. Klemenschits, and L. Filipovic. “Impact of Plasma Induced Damage on the Fabrication of 3D NAND Flash Memory”. In: *Solid-State Electronics* 192 (2022). invited, p. 108261. DOI: 10.1016/j.sse.2022.108261.

[36] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. “A Variational Level Set Approach to Multiphase Motion”. In: *Journal of Computational Physics* 127.1 (1996), pp. 179–195. DOI: 10.1006/jcph.1996.0167.

[37] S. C. Endres, M. Avila, and L. Mädler. “A Discrete Differential Geometric Formulation of Multiphase Surface Interfaces for Scalable Multiphysics Equilibrium Simulations”. In: *Chemical Engineering Science* 257 (2022), p. 117681. DOI: 10.1016/j.ces.2022.117681.

[38] L. Ma, Y. Li, J. Li, C. Wang, R. Wang, and M. A. Chapman. “Mobile Laser Scanned Point-Clouds for Road Object Detection and Extraction: A Review”. In: *Remote Sensing* 10.10 (2018). DOI: 10.3390/rs10101531.

[39] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. “Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1355–1361. DOI: 10.1109/ICRA.2017.7989161.

[40] S. B. Walsh, D. J. Borello, B. Guldur, and J. F. Hajjar. “Data Processing of Point Clouds for Object Detection for Structural Engineering Applications”. In: *Computer-Aided Civil and Infrastructure Engineering* 28.7 (2013), pp. 495–508. DOI: 10.1111/mice.12016.

[41] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst. “Embree: A Kernel Framework for Efficient CPU Ray Tracing”. In: *ACM Transactions on Graphics* 33.4 (2014), pp. 1–8. DOI: 10.1145/2601097.2601199.

[42] J. Otepka, S. Ghuffar, C. Waldhauser, R. Hochreiter, and N. Pfeifer. “Georeferenced Point Clouds: A Survey of Features and Point Cloud Management”. In: *ISPRS International Journal of Geo-Information* 2.4 (2013), pp. 1038–1065. DOI: 10.3390/ijgi2041038.

[43] T. Mølhave, P. K. Agarwal, L. Arge, and M. Revsbæk. “Scalable Algorithms for Large High-Resolution Terrain Data”. In: *Proceedings of the International Conference and Exhibition on Computing for Geospatial Research & Application (COM-GEO)*. 2010. DOI: 10.1145/1823854.1823878.

[44] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. 4th. Morgan Kaufmann, 2006.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

TU Bibliothek
WIEN Your knowledge hub

[45] T. Caelli and J. Berkmann. "Computation of Surface Geometry and Segmentation Using Covariance Techniques". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 16.11 (1994), pp. 1114–1116. DOI: `10.1109/34.334391`.

[46] K. Klasing, D. Althoff, D. Wollherr, and M. Buss. "Comparison of Surface Normal Estimation Methods for Range Sensing Applications". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 3206–3211. DOI: `10.1109/ROBOT.2009.5152493`.

[47] C. Siu-Wing, D. Tamal K., and S. Jonathan. *Delaunay Mesh Generation*. 1st. Chapman and Hall/CRC, 2013.

[48] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000. DOI: `10.1007/978-3-662-03427-9`.

[49] P. P. Pébay and T. J. Baker. "Analysis of Triangle Quality Measures". In: *Mathematics of Computation* 72 (2003), pp. 1817–1839. DOI: `10.1090/S0025-5718-03-01485-6`.

[50] I. Babuška and A. K. Aziz. "On the Angle Condition in the Finite Element Method". In: *SIAM Journal on Numerical Analysis* 13.2 (1976), pp. 214–226. DOI: `10.1137/0713021`.

[51] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. 2nd. Addison-Wesley, 1997.

[52] M. Chen, X. Chen, K. Tang, and M. Yuen. "Efficient Boolean Operation on Manifold Mesh Surfaces". In: *Computer-Aided Design and Applications* 7 (2013), pp. 405–415. DOI: `10.3722/cadaps.2010.405-415`.

[53] A. Requicha and H. Voelcker. "Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms". In: *Proceedings of the IEEE* 73.1 (1985), pp. 30–44. DOI: `10.1109/PROC.1985.13108`.

[54] S. Osher and J. A. Sethian. "Fronts Propagating with Curvature Dependent Speed". In: *Journal of Computational Physics* 79.1 (1988), pp. 12–49. DOI: `10.1016/0021-9991(88)90002-2`.

[55] R. T. Whitaker. "A Level-Set Approach to 3D Reconstruction from Range Data". In: *International Journal of Computer Vision* 29 (1998), pp. 203–231. DOI: `10.1023/A:1008036829907`.

[56] D. Adalsteinsson and J. A. Sethian. "A Fast Level Set Method for Propagating Interfaces". In: *Journal of Computational Physics* 118.2 (1995), pp. 269–277. DOI: `10.1006/jcph.1995.1098`.

[57] E. L. C. *Partial Differential Equations*. 2nd. Berkeley: Graduate Studies in Mathematics, 1998. DOI: `10.1090/gsm/019`.

[58] O. Ertl. "Numerical Methods for Topography Simulation". Doctoral Dissertation. TU Wien, 2010. DOI: `10.34726/hss.2010.001`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[59] B. Wyvill, A. Guy, and E. Galin. "Extending the CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System". In: *Computer Graphics Forum* 18.2 (1999), pp. 149–158. DOI: 10.1111/1467-8659.00365.

[60] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. "Function Representation in Geometric Modeling: Concepts, Implementation and Applications". In: *The Visual Computer* 11 (1995), pp. 429–446. DOI: 10.1007/BF02464333.

[61] C. Lenz, A. Toifl, M. Quell, F. Rodrigues, A. Hössinger, and J. Weinbub. "Curvature Based Feature Detection for Hierarchical Grid Refinement in TCAD Topography Simulations". In: *Solid-State Electronics* 191 (2022), p. 108258. DOI: 10.1016/j.sse.2022.108258.

[62] W. E. Lorensen and H. E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH)* 21.4 (1987), pp. 163–169. DOI: 10.1145/37401.37422.

[63] C. Maple. "Geometric Design and Space Planning Using the Marching Squares and Marching Cube Algorithms". In: *Proceedings of the International Conference on Geometric Modeling and Graphics (GMAG)*. 2003, pp. 90–95. DOI: 10.1109/GMAG.2003.1219671.

[64] M. W. Jones. *3D Distance from a Point to a Triangle*. Tech. rep. Department of Computer Science, University of Wales Swansea Technical Report CSR-5, 1995.

[65] U. Pinkall and K. Polthier. "Computing Discrete Minimal Surfaces and Their Conjugates". In: *Experimental Mathematics* 2.1 (1993), pp. 15–36. DOI: 10.1080/10586458.1993.10504266.

[66] F. Cazals and M. Pouget. "Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets". In: *Computer Aided Geometric Design* 22.2 (2005), pp. 121–146. DOI: 10.1016/j.cagd.2004.09.004.

[67] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds". In: *Proceedings of Visualization and Mathematics III (MATHVISUAL)*. Ed. by H.-C. Hege and K. Polthier. Springer Berlin Heidelberg, 2003, pp. 35–57. DOI: 10.1007/978-3-662-05105-4_2.

[68] Q. Du, V. Faber, and M. Gunzburger. "Centroidal Voronoi Tessellations: Applications and Algorithms". In: *SIAM Review* 41.4 (1999), pp. 637–676. DOI: 10.1137/S0036144599352836.

[69] D. Ulrich, H. Stefan, K. Albrecht, and W. Ortwin. *Minimal Surfaces II*. 1st ed. Springer, 1992.

[70] D. Ulrich, H. Stefan, K. Albrecht, and W. Ortwin. *Minimal Surfaces I*. 1st ed. Springer, 1992.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[71] E. Abbena, S. Salamon, and A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. 3rd. Chapman and Hall/CRC, 2006. DOI: 10.1201/9781315276038.

[72] K. Polthier and M. Schmies. "Straightest Geodesics on Polyhedral Surfaces". In: *Mathematical Visualization: Algorithms, Applications and Numerics*. Springer Berlin Heidelberg, 1998, pp. 135–150. DOI: 10.1007/978-3-662-03567-2_11.

[73] R. Goldman. "Curvature Formulas for Implicit Curves and Surfaces". In: *Computer Aided Geometric Design* 22.7 (2005), pp. 632–658. DOI: 10.1016/j.cagd.2005.06.005.

[74] C. Lenz, L. F. Aguinsky, A. Hössinger, and J. Weinbub. "A Complementary Topographic Feature Detection Algorithm Based on Surface Curvature for Three-Dimensional Level-Set Functions". In: *Journal of Scientific Computing* 94 (2023), p. 21. DOI: 10.1007/s10915-023-02133-5.

[75] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. "Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications". In: *Proceedings of the IEEE Visualization Conference (VIS)*. 2003, pp. 513–520. DOI: 10.1109/VISUAL.2003.1250414.

[76] R. T. Whitaker and X. Xue. "Variable-Conductance, Level-Set Curvature for Image Denoising". In: *Proceedings of the International Conference on Image Processing (ICIP)*. 2001, pp. 142–145. DOI: 10.1109/icip.2001.958071.

[77] A. Lefohn and R. T. Whitaker. *A GPU-Based, Three-Dimensional Level Set Solver with Curvature Flow*. Tech. rep. UC Davis: Institute for Data Analysis and Visualization, 2002.

[78] J. A. Sethian and D. Adalsteinsson. "An Overview of Level Set Methods for Etching, Deposition, and Lithography Development". In: *IEEE Transactions on Semiconductor Manufacturing* 10 (1997), pp. 167–184. DOI: 10.1109/66.554505.

[79] C.-W. Shu and S. Osher. "Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes". In: *Journal of Computational Physics* 77.2 (1988), pp. 439–471. DOI: 10.1016/0021-9991(88)90177-5.

[80] R. J. Spiteri and S. J. Ruuth. "A New Class of Optimal High-Order Strong-Stability-Preserving Time Discretization Methods". In: *SIAM Journal on Numerical Analysis* 40.2 (2002), pp. 469–491. DOI: 10.1137/S0036142901389025.

[81] B. Engquist and S. Osher. "Stable and Entropy Satisfying Approximations for Transonic Flow Calculations". In: *Mathematics of Computation* 34.149 (1980), pp. 45–75. DOI: 10.2307/2006220.

[82] M. G. Crandall and P.-L. Lions. "Two Approximations of Solutions of Hamilton-Jacobi Equations". In: *Mathematics of Computation* 43 (1984), pp. 1–19. DOI: 10.1090/S0025-5718-1984-0744921-8.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[83] S. K. Godunov and I. Bohachevsky. “Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics”. In: *Matematičeskij sbornik* 47(89).3 (1959), pp. 271–306.

[84] W. H. Press. *Numerical Recipes: The Art of Scientific Computing.* 3rd ed. Cambridge University Press, 1986.

[85] B. Radjenović, J. K. Lee, and M. Radmilović-Radjenović. “Sparse Field Level Set Method for Non-Convex Hamiltonians in 3D Plasma Etching Profile Simulations”. In: *Computer Physics Communications* 174.2 (2006), pp. 127–132. DOI: `10.1016/j.cpc.2005.09.010`.

[86] A. Harten and S. Osher. “Uniformly High-Order Accurate Nonoscillatory Schemes. I”. In: *SIAM Journal on Numerical Analysis* 24.2 (1987), pp. 279–309. DOI: `10.1137/0724022`.

[87] A. Toifl, M. Quell, X. Klemenschits, P. Manstetten, A. Hössinger, S. Selberherr, and J. Weinbub. “The Level-Set Method for Multi-Material Wet Etching and Non-Planar Selective Epitaxy”. In: *IEEE Access* 8 (2020), pp. 115406–115422. DOI: `10.1109/ACCESS.2020.3004136`.

[88] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations, Second Edition.* Society for Industrial and Applied Mathematics, 2004. DOI: `10.1137/1.9780898717938`.

[89] R. Courant, K. Friedrichs, and H. Lewy. “Über die Partiellen Differenzengleichungen der Mathematischen Physik”. In: *Mathematische Annalen* 100 (1928), pp. 32–74.

[90] D. L. Chopp. “Computing Minimal Surfaces via Level Set Curvature Flow”. In: *Journal of Computational Physics* 106.1 (1993), pp. 77–91. DOI: `10.1006/jcph.1993.1092`.

[91] J. A. Sethian. “A Fast Marching Level Set Method for Monotonically Advancing Fronts”. In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595. DOI: `10.1073/PNAS.93.4.1591`.

[92] E. Rouy and A. Tourin. “A Viscosity Solutions Approach to Shape-From-Shading”. In: *SIAM Journal on Numerical Analysis* 29.3 (1992), pp. 867–884.

[93] J. V. Gomez, D. Alvarez, S. Garrido, and L. Moreno. “Fast Methods for Eikonal Equations: An Experimental Survey”. In: *IEEE Access* 7 (2019), pp. 39005–39029. DOI: `10.1109/ACCESS.2019.2906782`.

[94] J. Weinbub and A. Hössinger. “Comparison of the Parallel Fast Marching Method, the Fast Iterative Method, and the Parallel Semi-Ordered Fast Iterative Method”. In: *Procedia Computer Science* 80 (2016), pp. 2271–2275. DOI: `10.1016/j.procs.2016.05.408`.

[95] D. Adalsteinsson and J. A. Sethian. “The Fast Construction of Extension Velocities in Level Set Methods”. In: *Journal of Computational Physics* 148.1 (1999), pp. 2–22. DOI: `10.1006/jcph.1998.6090`.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[96] D. Adalsteinsson and J. A. Sethian. "A Level Set Approach to a Unified Model for Etching, Deposition, and Lithography I: Algorithms and Two-Dimensional Simulations". In: *Journal of Computational Physics* 120 (1995), pp. 128–144. DOI: 10.1006/JCPH.1995.1153.

[97] D. Adalsteinsson and J. A. Sethian. "A Level Set Approach to a Unified Model for Etching, Deposition, and Lithography II: Three-Dimensional Simulations". In: *Journal of Computational Physics* 122 (1995), pp. 348–366. DOI: 10.1006/JCPH.1995.1221.

[98] M. Quell, A. Toifl, A. Hössinger, S. Selberherr, and J. Weinbub. "Parallelized Level-Set Velocity Extension Algorithm for Nanopatterning Applications". In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2019, pp. 335–338. DOI: 10.1109/SISPAD.2019.8870482.

[99] A. L. Magna and G. Garozzo. "Factors Affecting Profile Evolution in Plasma Etching of SiO2 : Modeling and Experimental Verification". In: *Journal of The Electrochemical Society* 150 (2003), F178–F185. DOI: 10.1149/1.1602084.

[100] F. Rodrigues, L. F. Aguinsky, A. Toifl, A. Scharinger, A. Hössinger, and J. Weinbub. "Surface Reaction and Topography Modeling of Fluorocarbon Plasma Etching". In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2021, pp. 229–232. DOI: 10.1109/SISPAD54002.2021.9592583.

[101] L. F. Aguinsky, F. Rodrigues, G. Wachter, M. Trupke, U. Schmid, A. Hössinger, and J. Weinbub. "Phenomenological Modeling of Low-Bias Sulfur Hexafluoride Plasma Etching of Silicon". In: *Solid-State Electronics* 191 (2022), p. 108262. DOI: 10.1016/j.sse.2022.108262.

[102] M. J. Kushner. "Hybrid Modelling of Low Temperature Plasmas for Fundamental Investigations and Equipment Design". In: *Journal of Physics D: Applied Physics* 42.19 (2009), p. 194013. DOI: 10.1088/0022-3727/42/19/194013.

[103] P. Manstetten. "Efficient Flux Calculations for Topography Simulation". Doctoral Dissertation. TU Wien, 2018. DOI: 10.34726/hss.2018.57263.

[104] O. Ertl and S. Selberherr. "A Fast Void Detection Algorithm for Three-Dimensional Deposition Simulation". In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2009, pp. 174–177. DOI: 10.1109/SISPAD.2009.5290221.

[105] K. Bean. "Anisotropic Etching of Silicon". In: *IEEE Transactions on Electron Devices* 25.10 (1978), pp. 1185–1193. DOI: 10.1109/T-ED.1978.19250.

[106] I. Zubel. "Anisotropic Etching of Si". In: *Journal of Micromechanics and Microengineering* 29.9 (2019), p. 093002. DOI: 10.1088/1361-6439/ab2b8d.

[107] P. Manstetten, A. Hössinger, J. Weinbub, and S. Selberherr. “Accelerated Direct Flux Calculations Using an Adaptively Refined Icosahedron”. In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2017, pp. 73–76. DOI: `10.23919/SISPAD.2017.8085267`.

[108] T. S. Cale, G. B. Raupp, and T. H. Gandy. “Free Molecular Transport and Deposition in Long Rectangular Trenches”. In: *Journal of Applied Physics* 68.7 (1990), pp. 3645–3652. DOI: `10.1063/1.346328`.

[109] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. 3rd. Wiley, 2016.

[110] P. Manstetten, J. Weinbub, A. Hössinger, and S. Selberherr. “Using Temporary Explicit Meshes for Direct Flux Calculation on Implicit Surfaces”. In: *Procedia Computer Science* 108 (2017), pp. 245–254. DOI: `10.1016/j.procs.2017.05.067`.

[111] S. J. Ruuth. “A Diffusion-Generated Approach to Multiphase Motion”. In: *Journal of Computational Physics* 145.1 (1998), pp. 166–192. DOI: `10.1006/jcph.1998.6028`.

[112] K. Smith, F. Solis, and D. Chopp. “A Projection Method for Motion of Triple Junctions by Level Sets”. In: *Interfaces and Free Boundaries* 4.3 (2002), pp. 263–276. DOI: `10.4171/IFB/61`.

[113] H. Li, Y. Yap, J. Lou, and Z. Shang. “Numerical Modelling of Three-Fluid Flow Using the Level-Set Method”. In: *Chemical Engineering Science* 126 (2015), pp. 224–236. DOI: `10.1016/j.ces.2014.11.062`.

[114] O. Ertl and S. Selberherr. “A Fast Level Set Framework for Large Three-Dimensional Topography Simulations”. In: *Computer Physics Communications* 180.8 (2009), pp. 1242–1250. DOI: `10.1016/j.cpc.2009.02.002`.

[115] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. 1st. Chapman and Hall/CRC, 2010. DOI: `10.1201/EBK1439811924`.

[116] Mahapatra, N. R. and Venkatrao, Balakrishna. “The Processor-Memory Bottleneck: Problems and Solutions”. In: *XRDS* 5.3 (1999), 2–es. DOI: `10.1145/357783.331677`.

[117] Null, L. and Lobur, J. *The Essentials of Computer Organization and Architecture*. 5th. Jones and Bartlett Publishers, 2006.

[118] G. M. Amdahl. “Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities”. In: *Proceedings of the Spring Joint Computer Conference (AFIPS)*. 1967, pp. 483–485. DOI: `10.1145/1465482.1465560`.

[119] J. L. Gustafson. “Reevaluating Amdahl’s Law”. In: *Communications of the ACM* 31.5 (1988), pp. 532–533. DOI: `10.1145/42411.42415`.

[120] Vienna Scientific Cluster. `https://vsc.ac.at/`; Accessed: 2023-5-18. 2022.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[121] S. V. Process. https://www.silvaco.com/tcad/victory-process-3d/; Accessed: 2023-5-18. 2022.

[122] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit.* Kitware, 2006.

[123] The CGAL Project. *CGAL User and Reference Manual.* 4.12.1. CGAL Editorial Board, 2018.

[124] F. Rudolf, J. Weinbub, K. Rupp, and S. Selberherr. "The Meshing Framework ViennaMesh for Finite Element Applications". In: *Journal of Computational and Applied Mathematics* 270 (2014), pp. 166–177. DOI: 10.1016/j.cam.2014.02.005.

[125] *Intel Embree.* https://www.embree.org/; Accessed:2023-5-18. 2022.

[126] Y. Liu, F. Kong, and F. Yan. "Level Set Based Shape Model for Automatic Linear Feature Extraction from Satellite Imagery". In: *Sensors and Transducers* 159.11 (2013), pp. 39–45.

[127] B. Beddad and K. Hachemi. "Brain Tumor Detection by Using a Modified FCM and Level Set Algorithms". In: *Proceedings of the International Conference on Control Engineering Information Technology (CEIT).* 2016, pp. 1–5. DOI: 10.1109/CEIT.2016.7929114.

[128] N. Christoff, A. Manolova, L. Jorda, S. Viseur, S. Bouley, and J.-L. Mari. "Level-Set Based Algorithm for Automatic Feature Extraction on 3D Meshes: Application to Crater Detection on Mars". In: *Proceedings of the Computer Vision and Graphics Conference (ICCVG).* 2018, pp. 103–114. DOI: 10.1007/978-3-030-00692-1_10.

[129] S. Popinet. "Numerical Models of Surface Tension". In: *Annual Review of Fluid Mechanics* 50 (2018), pp. 49–75. DOI: 10.1146/annurev-fluid-122316-045034.

[130] C. Lenz, A. Scharinger, M. Quell, P. Manstetten, A. Hössinger, and J. Weinbub. "Evaluating Parallel Feature Detection Methods for Implicit Surfaces". In: *Proceedings of the Austrian-Slovenian HPC Meeting (ASHPC).* 2021, p. 31. DOI: 10.3359/2021hpc.

[131] C. Dorai and A. Jain. "COSMOS-A Representation Scheme for 3D Free-Form Objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.10 (1997), pp. 1115–1130. DOI: 10.1109/34.625113.

[132] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1974.

[133] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. "Hierarchical RLE Level Set". In: *ACM Transactions on Graphics* 25 (2006), pp. 151–175. DOI: 10.1145/1122501.1122508.

[134] L. Filipović, O. Ertl, and S. Selberherr. "Parallelization Strategy for Herarchical Run Length Encoded Data Structures". In: *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN).* 2011, pp. 131–138. DOI: 10.2316/P.2011.719-045.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[135] J. Peng, Y. Qi, H.-C. Lo, P. Zhao, C. Yong, J. Yan, X. Dou, H. Zhan, Y. Shen, S. Regonda, O. Hu, H. Yu, M. Joshi, C. Adams, R. Carter, and S. Samavedam. "Source/Drain eSiGe Engineering for FinFET Technology". In: *Semiconductor Science and Technology* 32.9 (2017), p. 094004. DOI: `10.1088/1361-6641/aa7d3f`.

[136] Ted J. Hubbard. "MEMS Design: The Geometry of Silicon Micromachining". PhD Thesis. California Institute of Technology, 1994. DOI: `10.7907/TK4C-M144`.

[137] H. Jang, S. Koo, D.-S. Byeon, Y. Choi, and D.-H. Ko. "Facet Evolution of Selectively Grown Epitaxial $\mathrm{Si_{1-x}Ge_x}$ Fin Layers in sub-100 nm Trench Arrays". In: *Journal of Crystal Growth* 532 (2020), p. 125429. DOI: `10.1016/j.jcrysgro.2019.125429`.

[138] Z. Yang, J. Ming, C. Qiu, M. Li, and X. He. "A Multigrid Multilevel Monte Carlo Method for Stokes–Darcy Model with Random Hydraulic Conductivity and Beavers–Joseph Condition". In: *Journal of Scientific Computing* 90 (2022). DOI: `10.1007/s10915-021-01742-2`.

[139] W. Joppich and S. Mijalković. *Multigrid Methods for Process Simulation*. 1st. Springer, 1993. DOI: `10.1007/978-3-7091-9253-5`.

[140] M. E. Hubbard. "Adaptive Mesh Refinement for Three-Dimensional Off-Line Tracer Advection over the Sphere". In: *International Journal for Numerical Methods in Fluids* 40.3-4 (2002), pp. 369–377. DOI: `10.1002/fld.320`.

[141] S. L. Cornford, D. F. Martin, V. Lee, A. J. Payne, and E. G. Ng. "Adaptive Mesh Refinement Versus Subgrid Friction Interpolation in Simulations of Antarctic Ice Dynamics". In: *Annals of Glaciology* 57.73 (2016), pp. 1–9. DOI: `10.1017/aog.2016.13`.

[142] F. Löffler, Z. Cao, S. R. Brandt, and Z. Du. "A new Parallelization Scheme for Adaptive Mesh Refinement". In: *Journal of Computational Science* 16 (2016), pp. 79–88. DOI: `10.1016/j.jocs.2016.05.003`.

[143] A. Talpaert. "Direct Numerical Simulation of Bubbles with Adaptive Mesh Refinement with Distributed Algorithms". PhD Thesis. Université Paris Sacla, 2017.

[144] R. A. Trompert, J. G. Verwer, and J. G. Blom. "Computing Brine Transport in Porous Media with an Adaptive-Grid Method". In: *International Journal for Numerical Methods in Fluids* 16.1 (1993), pp. 43–63. DOI: `10.1002/fld.1650160104`.

[145] C. Lenz, A. Toifl, A. Hössinger, and J. Weinbub. "Curvature-Based Feature Detection for Hierarchical Grid Refinement in Epitaxial Growth Simulations". In: *Proceedings of the Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS)*. 2021, pp. 109–110.

[146] K. Museth. "VDB: High-Resolution Sparse Volumes with Dynamic Topology". In: *ACM Transactions on Graphics* 32.3 (2013). DOI: `10.1145/2487228.2487235`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[147] M. Quell. "Parallel Velocity Extension and Load-Balanced Re-Distancing on Hierarchical Grids for High Performance Process TCAD". Doctoral Dissertation. TU Wien, 2022. DOI: 10.34726/hss.2022.97084.

[148] M. Berger and I. Rigoutsos. "An Algorithm for Point Clustering and Grid Generation". In: *IEEE Transactions on Systems, Man and Cybernetics* 21.5 (1991), pp. 1278–1286. DOI: 10.1109/21.120081.

[149] M. Quell, G. Diamantopoulos, A. Hössinger, and J. Weinbub. "Shared-Memory Block-Based Fast Marching Method for Hierarchical Meshes". In: *Journal of Computational and Applied Mathematics* 392 (2021), p. 113488. DOI: 10.1016/j.cam.2021.113488.

[150] C. Lenz, P. Manstetten, L. F. Aguinsky, F. Rodrigues, A. Hössinger, and J. Weinbub. "Automatic Grid Refinement for Thin Material Layer Etching in Process TCAD Simulations". In: *Solid-State Electronics* 200 (2023), p. 108534. DOI: 10.1016/j.sse.2022.108534.

[151] C. Lenz, P. Manstetten, A. Hössinger, and J. Weinbub. "Automatic Grid Refinement for Thin Material Layer Etching in Process TCAD Simulations". In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2022, pp. 11–12.

[152] S. Zhang, Z. Gong, J. J. McKendry, S. Watson, A. Cogman, E. Xie, P. Tian, E. Gu, Z. Chen, G. Zhang, A. E. Kelly, R. K. Henderson, and M. D. Dawson. "CMOS-Controlled Color-Tunable Smart Display". In: *IEEE Photonics Journal* 4 (2012), pp. 1639–1646. DOI: 10.1109/JPHOT.2012.2212181.

[153] P. Lindstrom and G. Turk. "Fast and Memory Efficient Polygonal Simplification". In: *Proceedings of the Conference IEEE Visualization (VIS)*. IEEE Computer Society Press, 1998, pp. 279–286. DOI: 10.1109/VISUAL.1998.745314.

[154] M. Garland and P. S. Heckbert. "Surface Simplification Using Quadric Error Metrics". In: *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH)*. 1997, pp. 209–216. DOI: 10.1145/258734.258849.

[155] H. Borouchaki and P. Frey. "Simplification of Surface Mesh using Hausdorff Envelope". In: *Computer Methods in Applied Mechanics and Engineering* 194.48 (2005), pp. 4864–4884. DOI: 10.1016/j.cma.2004.11.016.

[156] S. J. Kim, C. H. Kim, and D. Levin. "Surface Simplification Using a Discrete Curvature Norm". In: *Computers & Graphics* 26.5 (2002), pp. 657–663. DOI: 10.1016/S0097-8493(02)00121-8.

[157] C. Lenz, A. Scharinger, A. Hössinger, and J. Weinbub. "A Novel Surface Mesh Coarsening Method for Flux-Dependent Topography Simulations of Semiconductor Fabrication Processes". In: *Proceedings of the International Conferences on Scientific Computing in Electrical Engineering (SCEE)*. 2020, pp. 99–100.

[158] C. Lenz, A. Scharinger, P. Manstetten, A. Hössinger, and J. Weinbub. "A Novel Surface Mesh Simplification Method for Flux-Dependent Topography Simulations of Semiconductor Fabrication Processes". In: *Scientific Computing in Electrical Engineering*. Ed. by M. van Beurden, N. Budko, and W. Schilders. Springer, 2021, pp. 73–81. DOI: `10.1007/978-3-030-84238-3_8`.

[159] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Mesh Optimization". In: *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH)*. 1993, pp. 19–26. DOI: `10.1145/166117.166119`.

[160] T. Birsan and D. Tiba. "One Hundred Years Since the Introduction of the Set Distance by Dimitrie Pompeiu". In: *Proceedings of the Conference on System Modeling and Optimization (CSMO)*. 2006, pp. 35–39.

[161] R. Straub. "Exact Computation of the Hausdorff Distance Between Triangular Meshes". In: *EG Short Papers*. Proceedings of the Conference of The Eurographics Association (EG), 2007. DOI: `10.2312/egs.20071023`.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Own Publications

Journal Articles

[1] **Lenz, C.**, Aguinsky, L. F., Hössinger, A., Weinbub, J., "A Complementary Topographic Feature Detection Algorithm Based on Surface Curvature for Three-Dimensional Level-Set Functions". In: *Journal of Scientific Computing* 94 (2023), p. 21. DOI: 10.1007/s10915-023-02133-5.

[2] **Lenz, C.**, Manstetten, P., Aguinsky, L. F., Rodrigues, F., Hössinger, A., Weinbub, J., "Automatic Grid Refinement for Thin Material Layer Etching in Process TCAD Simulations". In: *Solid-State Electronics* 200 (2023), p. 108534. DOI: 10.1016/j.sse.2022.108534.

[3] **Lenz, C.**, Toifl, A., Quell, M., Rodrigues, F., Hössinger, A., Weinbub, J., "Curvature Based Feature Detection for Hierarchical Grid Refinement in TCAD Topography Simulations". In: *Solid-State Electronics* 191 (2022), p. 108258. DOI: 10.1016/j.sse.2022.108258.

Book Contributions

[4] **Lenz, C.**, Scharinger, A., Manstetten, P., Hössinger, A., Weinbub, J., "A Novel Surface Mesh Simplification Method for Flux-Dependent Topography Simulations of Semiconductor Fabrication Processes". In: *Scientific Computing in Electrical Engineering*. Ed. by M. van Beurden, N. Budko, and W. Schilders. Springer, 2021, pp. 73–81. DOI: 10.1007/978-3-030-84238-3_8.

[5] **Lenz, C.**, Toifl, A., Hössinger, A., Weinbub, J., "Curvature Based Feature Detection for Hierarchical Grid Refinement in TCAD Topography Simulations". In: *Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS)*. Ed. by B. Cretu. IEEE, 2021, pp. 1–4. DOI: 10.1109/EuroSOI-ULIS53016.2021.9560690.

Conference Contributions

[6] **Lenz, C.**, Manstetten, P., Hössinger, A., Weinbub, J., "Automatic Grid Refinement for Thin Material Layer Etching in Process TCAD Simulations". In: *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2022, pp. 11–12.

[7] **Lenz, C.**, Toifl, A., Hössinger, A., Weinbub, J., “Curvature-Based Feature Detection for Hierarchical Grid Refinement in Epitaxial Growth Simulations”. In: *Proceedings of the Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS)*. 2021, pp. 109–110.

[8] **Lenz, C.**, Scharinger, A., Quell, M., Manstetten, P., Hössinger, A., Weinbub, J., “Evaluating Parallel Feature Detection Methods for Implicit Surfaces”. In: *Proceedings of the Austrian-Slovenian HPC Meeting (ASHPC)*. 2021, p. 31. DOI: `10.3359/2021hpc`.

[9] **Lenz, C.**, Scharinger, A., Hössinger, A., Weinbub, J., “A Novel Surface Mesh Coarsening Method for Flux-Dependent Topography Simulations of Semiconductor Fabrication Processes”. In: *Proceedings of the International Conferences on Scientific Computing in Electrical Engineering (SCEE)*. 2020, pp. 99–100.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Curriculum Vitae

Personal Information

Name	Christoph Lenz
Date of Birth	September 16, 1988, Wien
Nationality	Austrian
Place of Birth	Vienna, Austria

Education

06/2019	-	present	Doctoral Program, *Electrical Engineering,* *Institute for Microelectronics,* *TU Wien*
04/2016	-	04/2019	Graduate Studies (MSc), *Technical Mathematics,* *Discrete Mathematics,* *TU Wien,*
10/2009	-	11/2017	Graduate Studies (BSc), *Technical Mathematics,* *TU Wien,*
09/2003	-	06/2008	Matura, *Majors: Accounting and Data Processing* *HTL Donaustadt, Wien*

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Employment

06/2019	-	04/2023	Project Assistant, *Christian Doppler Laboratory for High Performance TCAD*, *Institute for Microelectronics, TU Wien*
09/2016	-	05/2019	Software Tester, *Usoft GmbH, Wien*
05/2013	-	06/2015	Salesman, *McSHARK, Wien*
02/2012	-	05/2013	Freelance Programmer, *Tchibo Coffee Service (Austria) GmbH, Wien*
03/2010	-	05/2013	Salesman, *MediaMarkt Österreich, Wien*

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.