



**Universidad de Jaén**

Escuela de Doctorado

**TESIS DOCTORAL**



**MULTILAYER REPRESENTATION FOR  
GEOLOGICAL INFORMATION SYSTEMS**

**PRESENTADA POR:  
ALEJANDRO GRACIANO SEGURA**

**DIRIGIDA POR:  
DR. D. ANTONIO JESÚS RUEDA RUIZ  
DR. D. FRANCISCO RAMÓN FEITO HIGUERUELA**

**JAÉN, 8 DE JULIO DE 2019**

**ISBN**

ALEJANDRO GRACIANO

MULTILAYER REPRESENTATION FOR GEOLOGICAL  
INFORMATION SYSTEMS



# MULTILAYER REPRESENTATION FOR GEOLOGICAL INFORMATION SYSTEMS

ALEJANDRO GRACIANO

A dissertation submitted in partial fulfillment of the  
requirements for the degree of Doctor of Philosophy

SUPERVISORS:

Dr. Antonio J. Rueda

Dr. Francisco R. Feito



Universidad de Jaén

Departamento de Informática  
Escuela Politécnica Superior de Jaén  
Universidad de Jaén

July 2019

Alejandro Graciano: *Multilayer representation for geological information systems*, © July 2019

El **Dr. Antonio J. Rueda Ruiz**, Profesor Titular de Universidad y el **Dr. Francisco R. Feito Higuera** Catedrático de Universidad del Departamento de Informática de la Universidad de Jaén, España,

CERTIFICAN:

Que la presente memoria, titulada **Multilayer representation for geological information systems**, ha sido realizada bajo su dirección. Y considerando que representa trabajo de Tesis, autorizan su presentación y defensa para optar al grado de Doctor por la Universidad de Jaén con mención de Doctor Internacional.

**Dr. Antonio J. Rueda Ruiz**      **Dr. Francisco R. Feito Higuera**

Dpto. de Informática

Dpto. de Informática

Universidad de Jaén

Universidad de Jaén.

**Alejandro Graciano Segura**

Ingeniero en Informática

Jaén, Julio de 2019



## ABSTRACT

---

Many geological applications require detailed and accurate models that represent the complexity of the geological reality. Computer representations for these models have to be efficient in terms of storage requirements, but at the same time flexible to allow the implementation of analysis methods, simulations of natural processes and fast advanced visualization. In this thesis, we propose the use of the Stack-Based Representation of Terrains (SBRT) for volumetric geological data. This data structure encodes geological structures represented as stacks using a compact data representation. In contrast with previous work where SBRT was used only for simulation of natural processes or as intermediate representation for rendering complex terrains, we suggest its use as main data structure for both processing and visualization of surface and subsurface information. The SBRT is further formalized with a framework based on the geo-atom theory to provide a precise definition and determine its properties. In addition, we have defined a set of common spatial primitives on this representation using the tools provided by map algebra.

Stacks can be organized hierarchically, introducing QuadStack, a novel data structure that improves the compression results provided by the SBRT. QuadStack exploits in its data arrangement the redundancy often found in layered dataset that are common, not only in geological models, but also in science and engineering fields such as geology, biology, mechanical engineering, medicine, etc. The associated data (color, material, etc.) and shape of these layer structures are decoupled and encoded independently, leading to high compression rates.

Visualization of geological models is one of the most useful applications for geoscientists. This is of great importance for the inspection of the data and interpretation of its structure and complexity, as well as in decision-making processes. This thesis also provides direct visualization methods for the SBR and QuadStack based on the well-known raycasting algorithm. By keeping the whole datasets in the GPU memory in a compact way, the proposed methods are fast enough to provide real-time frame rates. Furthermore, the implementation of some visual operations common in geoscientific applications such as borehole visualization, attenuation of material layers or cross sections has been carried out for the SBRT.



## RESUMEN

---

Muchas aplicaciones geológicas requieren modelos detallados y precisos que representen la complejidad del mundo geológico. Las representaciones de estos modelos usadas en sistemas informáticos tienen que ser eficientes en términos de requisitos de memoria, pero al mismo tiempo flexibles para permitir la implementación de métodos de análisis, simulaciones de procesos naturales y una rápida y avanzada visualización. En esta tesis proponemos el uso de la Representación de Terrenos Basada en Stacks (SBRT, por sus siglas en inglés) para datos geológicos volumétricos. Esta estructura de datos codifica estructuras geológicas representadas como stacks utilizando una compacta representación de datos. A diferencia de trabajos anteriores en los que la SBRT se ha utilizado sólo para la simulación de procesos naturales o como representación intermedia para la visualización de terrenos complejos, sugerimos su uso como estructura de datos principal tanto para el procesamiento como para la visualización de la información de la superficie y del subsuelo. A continuación, hemos formalizado la SBRT con un esquema basado en la teoría de geo-átomos para proporcionar una definición precisa y determinar sus propiedades. Además, hemos definido un conjunto de primitivas espaciales sobre esta representación utilizando las herramientas proporcionadas por el álgebra de mapas.

Los stacks pueden organizarse jerárquicamente en una nueva estructura de datos llamada QuadStack, mejorando los resultados de compresión proporcionados por la SBRT. La organización de datos dentro un QuadStack se realiza en base al aprovechamiento de la redundancia de información que a menudo se encuentra en los datos distribuidos por capas. Esta redundancia es común no solo en modelos geológicos, sino también en otros campos de la ciencia y la ingeniería como la biología, la ingeniería mecánica, la medicina, etc. En un QuadStack los atributos tales como el color o el material son desacoplados de su información geométrica y comprimidos de forma separada, dando lugar a altos índices de compresión.

La visualización de modelos geológicos es una de las aplicaciones más útiles para los geocientíficos. Esta aplicación tiene una gran importancia para interpretar la estructura y complejidad de los datos geológicos, así como en la toma de decisiones. En esta tesis hemos proporcionado mé-

todos de visualización directa para la representación basada en stacks y para QuadStacks basándonos en el conocido algoritmo de visualización raycasting. Al mantener los datos en todo momento y de forma compacta en la memoria de la GPU, los métodos propuestos son lo suficientemente rápidos como para proporcionar una visualización con tiempos de respuesta interactivos. Además, se han implementado para la SBRT una serie de operaciones visuales comunes en aplicaciones geocientíficas como la visualización de catas geológicas o secciones transversales, o el ocultamiento de capas de material.



## PUBLICATIONS

---

A large part of the work presented in this thesis has already been published or is being peer-reviewed to be published in scientific conferences and journals:

### ■ JCR-indexed journals

- **Graciano, A.**, Rueda, A.J. & Feito, F.R., 2018. A formal framework for the representation of stack-based terrains. *International Journal of Geographical Information Science*, 32(10), pp.1999–2022.

Journal impact factor: 3.545. Q1 (34/155) in *Computer Science, Information Systems* category.

- **Graciano, A.**, Rueda, A.J. & Feito, F.R., 2018. Real-time visualization of 3D terrains and subsurface geological structures. *Advances in Engineering Software*, 115, pp.314–326.

Journal impact factor: 4.194. Q1 (17/106) in *Computer Science, Interdisciplinary Applications* and Q1 (9/107) in *Computer Science, Software Engineering* categories.

### ■ Peer-reviewed conferences

- **Graciano, A.**, Rueda, A.J., Bittner, J., Pospíšil, A. & Benes, B., 2019. QuadStack: An efficient representation and direct rendering of layered datasets. SIGGRAPH Asia. (*Under review*).

In case of acceptance, this paper will also be published in a special issue of the ACM Transactions on Graphics journal. Journal impact factor: 6.495. Q1 (1/107) in *Computer Science, Software Engineering* category.

- **Graciano, A.**, Rueda, A.J., Ortega, L. & Feito, F.R. 2017. Towards a hybrid framework for the visualization and analysis of 3D spatial data. In Proceedings of the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics - UrbanGIS'17. November 7–10, Redondo Beach, CA (USA), pp. 1–8.

- **Graciano, A.**, Rueda, A.J. & Feito, F.R., 2017. Direct volume rendering of stack-based terrains. In proceedings of the XXVII

Spanish Computer Graphics Conference (CEIG). June 28-30,  
Sevilla (Spain), pp.41-50.

*Impossible to map the world -  
we select and make graphics  
so that we can understand it.*

– **Roger Tomlinson**

## ACKNOWLEDGMENTS

---

El trabajo que hoy culmino ha necesitado de cuatro duros años de trabajo. Cuatro años de retos, alegrías, frustraciones y aprendizaje compartidos con personas a las que quiero agradecer su aportación tanto en esta tesis doctoral como a nivel personal.

En primer lugar, me gustaría expresar mi más sincero agradecimiento a mis directores de tesis Antonio Rueda y Francisco Feito por darme la oportunidad de trabajar con ellos todo este tiempo. Gracias por confiar en mí y guiarme tan fantásticamente durante estos años. Quiero también agradecer a todos los miembros del Grupo de Gráficos y Geomática de Jaén (GGGJ) por haberme hecho sentir como en casa desde el primer día. En especial, a Lidia Ortega por dejarme colaborar con ella en todas las asignaturas que he impartido durante mi doctorado, a Juan Ruiz y Juanjo Jiménez con los que he trabajado en distintas etapas antes del comienzo de esta tesis, y finalmente a Ángel Luis García por sus revisiones finales de este texto.

De igual manera quiero transmitir mi gratitud a todos y cada uno de los compañeros con los que he compartido los laboratorios A3-102 y A3-103. Especialmente, quiero dar las gracias a Félix Paulano por darme la bienvenida al grupo y a José Negrillo por hacer amenos tantos días de trabajo.

Por otro lado, quiero agradecer a mi familia y amigos por estar a mi lado durante el transcurso de mi doctorado. A mis padres por la educación y las oportunidades que me han dado y por su apoyo a pesar de no saber qué he estado haciendo exactamente estos cuatro años. Tampoco quiero olvidarme de los *pesados* de mis amigos con los que he pasado tantas horas entre cafés y almuerzos en la universidad y fuera de ella.

De forma muy especial quisiera agradecer a Isa el apoyo, comprensión y amor que me ha dado todos estos años. Muchas gracias por hacerme ver el mundo de otra manera y por dejarme crecer como persona estos años a tu lado.

Finalmente, transmitir mi agradecimiento a la Universidad de Jaén por financiar la realización de esta tesis doctoral así como a Bedrich Benes y al grupo de investigación High Performance Computer Graphics de la Universidad de Purdue por aceptar mi solicitud para realizar mi estancia de investigación con ellos.

# CONTENTS

---

## I INTRODUCTION AND OVERVIEW

1	INTRODUCTION	3
1.1	Aims and objectives . . . . .	6
1.2	Organization of this document . . . . .	6
2	FUNDAMENTALS OF GEOLOGICAL MODELING	9
2.1	Continuous field models . . . . .	10
2.1.1	Grid-based models and data structures in geological modeling . . . . .	11
2.1.2	Grids for numerical modeling . . . . .	13
2.1.3	Hierarchical grids for geomodeling . . . . .	14
2.2	Discrete object models . . . . .	15
2.2.1	Geological modeling with simplicial complexes . . . . .	16
2.2.2	Geological modeling with cell complexes . . . . .	17
2.2.3	Ordered topological geomodeling . . . . .	18
2.3	Generalized strategies for geomodeling . . . . .	19
2.4	Comparison of data structures for geomodeling . . . . .	21
2.4.1	Topology . . . . .	21
2.4.2	Memory consumption . . . . .	23
2.4.3	Querying and updating . . . . .	23
2.4.4	Ease of construction . . . . .	23
2.5	Visualization in GIS . . . . .	24

## II DATA STRUCTURES FOR GEOLOGICAL LAYERED DATA

3	THE STACK-BASED REPRESENTATION OF TERRAINS	31
3.1	Efficiency in modeling geological fields . . . . .	31
3.2	Related work . . . . .	32
3.3	Defining the stack-based representation of terrains . . . . .	33
3.3.1	Mathematical definition . . . . .	33
3.4	Memory storage comparative . . . . .	35
4	FORMAL FRAMEWORK FOR THE STACK-BASED REPRESENTATION OF TERRAINS	37
4.1	Introduction . . . . .	37
4.2	Geo-atom theory background . . . . .	38
4.3	3D Terrains as geo-fields . . . . .	40



4.4	Stack-based terrains as geo-fields . . . . .	41
4.5	Operations with 3D terrains . . . . .	42
4.5.1	Map algebra background . . . . .	43
4.5.2	Definition of operations . . . . .	44
4.6	An implementation model . . . . .	51
4.6.1	Extending the model with SBRT domain classes . . . . .	54
4.7	Conclusion and future work . . . . .	55
5	EFFICIENT REPRESENTATION OF STACK-BASED INFORMATION . . . . .	59
5.1	Introduction . . . . .	59
5.2	The QuadStack data structure . . . . .	61
5.2.1	Group of stacks . . . . .	61
5.2.2	Group of stacks hierarchies . . . . .	63
5.2.3	QuadStack construction . . . . .	65
5.2.4	Heightfield compression . . . . .	67
5.2.5	Optimizations . . . . .	68
5.3	QuadStack sampling . . . . .	69
5.4	Conclusion . . . . .	71
<b>III VISUALIZATION OF GEOLOGICAL DATA</b>		
6	FUNDAMENTALS OF DIRECT VOLUME RENDERING . . . . .	75
6.1	Introduction . . . . .	75
6.2	Overview of direct volume rendering techniques . . . . .	75
6.2.1	The volume-rendering equation . . . . .	75
6.2.2	Transfer functions . . . . .	77
6.2.3	Direct volume rendering approaches . . . . .	77
6.3	Raycasting . . . . .	79
7	REAL-TIME RENDERING OF STACK-BASED DATA . . . . .	81
7.1	Introduction . . . . .	81
7.2	Raycasting the stack-based representation . . . . .	82
7.2.1	Surface normal vectors calculation . . . . .	84
7.2.2	Stack-based representation of terrains encoding in the GPU memory . . . . .	87
7.2.3	Visual operations . . . . .	91
7.2.4	GIS-based layer display . . . . .	92
7.2.5	Performance analysis . . . . .	96
7.3	Raycasting the QuadStack . . . . .	104
7.3.1	QuadStack encoding in the GPU memory . . . . .	105
7.3.2	Performance analysis . . . . .	107

7.4	Conclusion . . . . .	111
<b>IV CONCLUDING REMARKS</b>		
8	CONCLUSIONS	115
8.1	Summary of contributions . . . . .	115
8.2	Future work . . . . .	117
<b>V APPENDICES</b>		
A	PROGRAMMING CUSTOM VISUALIZATION ALGORITHMS ON GPU	121
A.1	Introduction to GPU programming . . . . .	121
A.2	OpenGL API . . . . .	124
A.2.1	Data model . . . . .	124
A.3	Raycasting implementation in GLSL . . . . .	125
B	DOCUMENTACIÓN EN CASTELLANO	131
	BIBLIOGRAPHY	151

## LIST OF FIGURES

---

Figure 1.1	Examples of geological observations in Renaissance. . . . .	4
Figure 1.2	Pipeline of the geological modeling process. Adapted from (Bobrowsky and Marker, 2018) and (Turner, 2006). The stages covered in this dissertation are denoted by red-dashed boxes. . . . .	5
Figure 2.1	A polygon represented by means of a) a simplicial complex, b) Nef representation and c) a G-map. . . . .	19
Figure 2.2	Classification of the geological data models according to the data model they represent. . . . .	21
Figure 2.3	Geological models visualized by the two different approaches: a) volume rendering of subsurface seismic data (Hollt et al., 2012) and b) a triangular mesh representing a strata model (Song et al., 2019). . . . .	26
Figure 3.1	A real example of borehole log extracted from a geotechnical survey performed at the University of Jaén (Spain). . . . .	33
Figure 3.2	Stack-Based Representation: A voxel dataset of resolution $w \times h \times d$ composed of layers is organized into vertical stacks consisting of intervals. Each interval $i_k = \langle a_k, h_k \rangle$ with the attribute value $a_k$ and height $h_k$ . . . . .	34
Figure 4.1	Relation between a 3D regular grid and a representative geo-atom. . . . .	41
Figure 4.2	Example of a 3D terrain model using a voxel representation (b). A DEM of the terrain is showed (a), a sample stack is generated from column of voxels (c) and an interval (d) are also represented. . . . .	42
Figure 4.3	Stack-based representation of terrains in the context of geo-atom theory. . . . .	43
Figure 4.4	Computation of the water transfer to a stack from its neighborhood in a flooding algorithm. . . . .	51

Figure 4.5	Examples of operations with a SBRT. A pipeline of operations is applied to an original SBRT geofield. . . . .	52
Figure 4.6	A simplified UML diagram depicting the main classes of an implementation of our framework and an extract of the ADE XML schema. . . . .	54
Figure 4.7	Use of GML for encoding a SBRT. . . . .	56
Figure 5.1	Vertical and horizontal spatial coherence in a same geomodel. Screenshot taken from SubsurfaceViewer software (Terrington et al., 2009). The geological data have been obtained from (Gunnink et al., 2013). . . . .	60
Figure 5.2	Overview: The input volumetric data is converted to stacks and similar stacks are then grouped into groups of stacks (gstacks) that are organized in a quadtree. When using the QuadStack the quadtree is traversed first and the topology of the given part of the volume is reconstructed. Then the corresponding layer boundaries encoded as heightfields are sampled to determine the result.	62
Figure 5.3	Details of the QuadStack construction: initial construction as a quadtree encoding groups of stacks with the same sequence of attributes (a), merging gstacks in nodes $n_1$ and $n_2$ into a gstack with common intervals and *-intervals (b), propagation of the new gstack to the parent node $n_0$ , restructuring heightfields (c) and optimization, deleting *-intervals in nodes $n_1$ and $n_2$ associated to the intervals propagated to the parent node (d). . . . .	64
Figure 5.4	The QuadStack construction: The initial QuadStack is a quadtree that is optimized in the second step by merging intervals of equal or similar attributes. . . . .	65
Figure 5.5	Finding a common mapping for two gstacks that maximizes the number of terminal intervals. . . . .	67
Figure 6.1	DVR approaches. Raycasting, shear-warp volume rendering, splatting, texture slicing and cell projection are conceptualized. . . . .	78

Figure 7.1 Deferred shading strategy used. The dotted lines indicate an usage relationship. . . . . 83

Figure 7.2 Estimation of a normal vector in an object space.  $\vec{N}$  (b) is the resulting from the sum of the vectors contained in empty space voxels by using a  $3 \times 3$  kernel (a). . . . . 84

Figure 7.3 Visual results when applying different kernel sizes to a SBRT with a grid dimension of  $800 \times 1000$ . Each subfigure shows a region with a dimension of  $200 \times 300$ . . . . . 85

Figure 7.4 Estimation of a normal vector in image space. The ray  $r_i$  provides two depth values, an actual (d) and an estimated ( $d'$ ) with which the binary value of the kernel (occupied or empty space) is calculated. . . . . 86

Figure 7.5 Memory storage patterns (b, c, d, e, f) for a stack-based geomodel (a). . . . . 90

Figure 7.6 Example of layer attenuation (bottom) of an original dataset (top). . . . . 92

Figure 7.7 Cross-section (bottom) and original model (top). 93

Figure 7.8 Example of borehole visualization. They are shown as cylinder of two materials (blue and yellow color). . . . . 93

Figure 7.9 Example of orthophoto application. Original model (a) and model with an orthophoto applied on its surface (b). . . . . 94

Figure 7.10 An original model (e) and different examples of visual operations applied: hiding of many layers (a), application of an orthophoto on the surface (b), borehole visualization combined with layer hiding (c) and visualization of cross-section (d). . 95

Figure 7.11 Internal description of the heterogeneous data layers using a camera enclosing bounding volume. 96

Figure 7.12 Example of attenuation of a field layer. A subsurface vector layer can be depicted. . . . . 97

Figure 7.13	A close-up viewing of three different layers. Two subsurface vector layers are combined with a SBRT layer. The vector layers represent a sewage network (segment layer) and a set of manholes (point layer). . . . .	97
Figure 7.14	Cross section of a field layer. The cut allows the visualization of a subsurface vector layer. In addition, an orthophoto is applied on the field layer surface. . . . .	98
Figure 7.15	Overview of the datasets used in the experiments. Dataset A is shown in the left-upper corner, Dataset B is in the right-upper corner, Dataset C is in the left-lower corner and Dataset D is in the right-lower corner. . . . .	99
Figure 7.16	Minimum MRPS reached. . . . .	101
Figure 7.17	Maximum MRPS reached. . . . .	101
Figure 7.18	Raycasting a QuadStack, first resolved at the QuadStack level, then at the interval level and finally, at the heightfield level. When an *-interval is found, a recursive call to traverse the gstacks in children nodes is required. After processing a gstack, the traverse continues with the next one, until the ray exits the volumetric model. . . . .	104
Figure 7.19	Memory layout of a given QuadStack after performing its heightfield arrangement. . . . .	106
Figure 7.20	Heightfield compression scheme. . . . .	107
Figure 7.21	Datasets used in the experiments. . . . .	108
Figure A.1	Programmable pipeline organization. The data flow between the fixed pipeline stages and the programmable shaders are shown. . . . .	123
Figure A.2	Deferred shading strategy using GLSL subroutines.	126
Figura Bo.1	Ejemplos de observaciones geológicas en el Renacimiento. . . . .	140
Figura Bo.2	Arquitectura del proceso de modelado geológico adaptado de (Bobrowsky y Marker, 2018) y (Turner, 2006). Las etapas que se tratan en esta tesis se indican con recuadros de color rojo. . . . .	141

## LIST OF TABLES

---

Table 2.1	Summary of the characteristics studied for the data structures reviewed. . . . .	25
Table 3.1	Memory consumption comparison. The octree measurements have been taken with six depth levels. . . . .	35
Table 4.1	Common discretizations for geo-fields. . . . .	39
Table 4.2	Examples of SBRT operations. . . . .	45
Table 4.3	A conceptual view of the operations of the SBRT. . . . .	46
Table 5.1	Comparison between the optimized and non optimized versions of the gstack matching. . . . .	69
Table 7.1	Relative drop in performance when using different kernel sizes over rendering without lighting . . . . .	86
Table 7.2	Storage requirements of memory layouts. . . . .	102
Table 7.3	Comparison of results for Dataset A. FPS means frames per seconds and MRPS, millions of rays per second. . . . .	102
Table 7.4	Comparison of results for Dataset B. FPS means frames per seconds and MRPS, millions of rays per second. . . . .	103
Table 7.5	Comparison of results for Dataset C. FPS means frames per seconds and MRPS, millions of rays per second. . . . .	103
Table 7.6	Comparison of results for Dataset D. FPS means frames per seconds and MRPS, millions of rays per second. . . . .	103
Table 7.7	Breakdown of the QuadStack memory requirements. The columns represent memory needed for representing the quadtree and attributes, compressed heightfields, and min-max mipmaps. . . . .	108

Table 7.8	Results measured on five test datasets. The table shows basic dataset properties, construction times and memory requirements for evaluated representations (Voxel grid, SBR, QuadStack). The bottom part of the table compares rendering performance for different methods (VTK, OSPRay, QuadStack). The results for the method with the lowest memory consumption and best rendering performance are highlighted in bold. . 110
-----------	--



## LISTINGS

---

Listing A.1	A simplified version of a raycasting function in GLSL . . . . .	127
Listing A.2	Sampling method of a SBRT encoded using SS-BOs in GLSL . . . . .	128
Listing A.3	Implementation of the deferred shading approach in GLSL . . . . .	129

## ACRONYMS

---

AABB	Axis-Aligned Bounding Box
ADE	Application Domain Extension
CAD	Computer-Aided Design
CIS	Coverage Implementation Schema
DEM	Digital Elevation Model
DVR	Direct Volume Rendering
GIS	Geographical Information System
GML	Geography Markup Language
GSIS	Geoscientific Information System
GTP	Generalized Tri-Prism
MC	Marching Cubes
SBR	Stack-Based Representation
SBRT	Stack-Based Representation of Terrains
S&IBF	Sampling and Interpolation-Based Field
TBF	Tessellation-Based Field
TF	Transfer Function
TEN	Tetrahedral Network
TIN	Triangular Irregular Network
VD	Voronoi Diagram

## NOTATION

---

Element	Representation
Point	$p$
Point coordinates	$p_{x,y,z}$
Voxel	$v_{x,y,z}$
Voxel column	$V_{x,y}$
Attribute value	$a_i$
Height value	$h_i$
Interval	$i_k$
Stack	$S_{x,y}$
Heightfield	$H_i$
Height value	$h_{x,y}$
Group of stacks	$G_i$
Tuple	$\langle t_0, t_1, \dots, t_{n-1} \rangle$
Set	$B = \{b_0, b_1, \dots, b_{n-1}\}$
Interval sequence	$i_0, i_1, \dots, i_{n-1}$
Vector	$\vec{n}$
Plane	$X$

## Part I

### INTRODUCTION AND OVERVIEW

This part introduces the general concepts that will be discussed during this document, as well as the main aims and objectives of this doctoral dissertation. Then, a review of the state-of-the-art of geological modeling is given.



## INTRODUCTION

---

Having a general knowledge of the geological structures, their organization, their properties and how they have evolved over time is of special interest to scientists in order to provide accurate resource evaluations or proper predictions of geological hazards. This understanding has largely increased thanks to the advances in data acquisition technologies, which have led to a vast amount of spatial data valuable for geoscientific and geo-information related fields. The science aimed at integrating all this heterogeneous data and generating computer-aided representations of geological structures, both at subsurface and above ground levels is **Geological modeling** or **Geomodeling**.

As many other disciplines, geological modeling has its roots during the Scientific Revolution in the 15th century. Arabic and Greek ideas were taken and renewed by scientists not only from a pure scientific point of view. Geomodeling attracted many Renaissance polymaths such as Leonardo Da Vinci, who carried out studies concerning geological formations. These studies were later used to increase the accuracy in their artistic creations. Figure 1.1.a shows a sketch dated back to 1473 in which precise geological layers in the landscape can be seen. Other artists like Albrecht Dürer, improved their results by adding further realism and increasing the details (Figure 1.1.b). These observations and principles were the foundations of geomodeling as we know it and inspired current geological mapping.

The reliability with which a geomodel is represented in a computer is crucial for its efficient management, processing and visualization. Therefore, the study of spatial data representations has been a key point in geoscientific research in the last decades. However, the study of new tools and efficient data structures that fulfill the so-called five Ms of Geographic Information Systems (GIS) continues to be of great importance. These data structures should be able to **Manage** spatial data in such a way that enable an efficient arrangement, keeping them up to date. Such possible updates should be **Monitored** so that spatial and temporal changes can be queried quickly and simply. The data organization is crucial not only for an appropriate management, but for an efficiently computing operations and **Measurements** on them. It is also desirable that the data structures **Model**

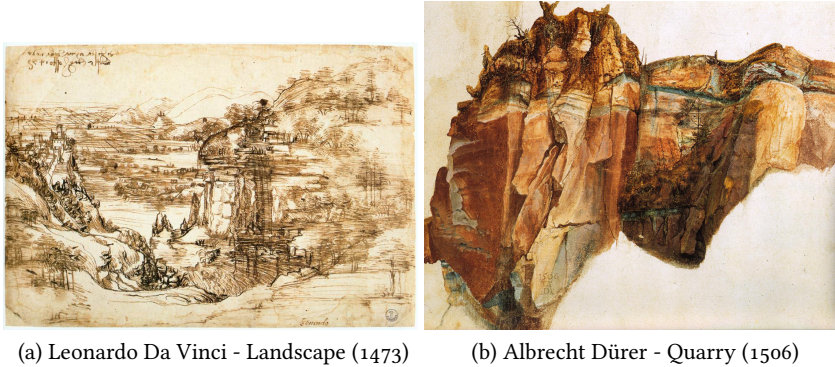


Figure 1.1: Examples of geological observations in Renaissance.

accurately the complexity of the data and integrate them in a compact manner, for example, by combining surface and subsurface information. In addition, finally yet importantly, extracting information to create **M**aps and graphical models in order to make decisions at a glance. This is a worthwhile feature for all tools and data structures that handle spatial data.

Despite the above points are defined for GIS systems, they should be met by any geospatial application. Geological modeling deals essentially with volumetric representations, rarely supported by traditional GIS. Therefore, at the end of 1980s, the concept of Geoscientific Information Systems (GSIS) emerged to define those systems devoted to managing, visualizing and operating with 3D geodata. Typically, GSIS systems follow the architecture depicted in Figure 1.2. The pipeline is split into three distinct parts or modules: the one concerning the data acquisition (1) where the input information is collected from field work or monitoring procedures. The core geological model (2) sets the related data structures representing the geological features. This model can be separated into two submodules. On one hand, interpolation methods are used in the geometry model part to generate the geometric and boundary information, and the discretization of the framework. On the other hand, the numerical model part uses mathematical and predictive models to extrapolate and simulate physical properties and phenomena. Often, both submodules are closely connected. The mathematical model can use the discretization or even take part in the generation of the data structure included in the geological model. Finally, as the ultimate goal of the pipeline (3), geoscientists can apply analysis

and visualization methods in order to extract new knowledge as well as in decision-making support.

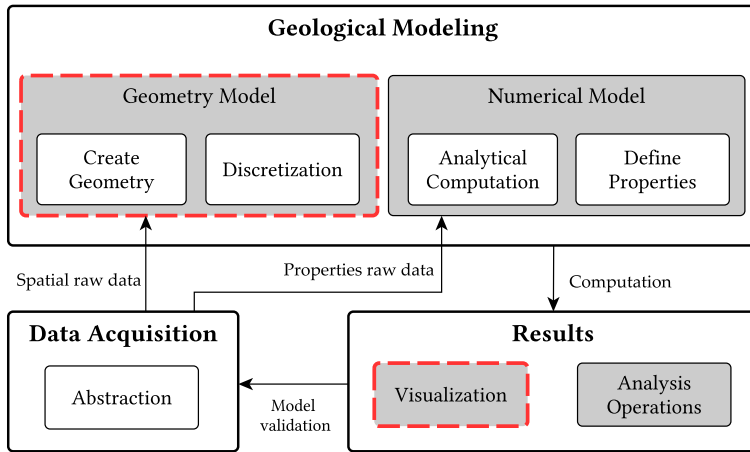


Figure 1.2: Pipeline of the geological modeling process. Adapted from (Bobrowsky and Marker, 2018) and (Turner, 2006). The stages covered in this dissertation are denoted by red-dashed boxes.

The visualization of geodata is intrinsically a powerful tool for GIS professionals. Just as in other scientific fields like medical imaging or computational fluid dynamics, having a global and real-time view of the data could provide a first insight in a rapid and effective manner. From the point of view of computer graphics, three-dimensional objects are rendered by using mainly surface or volumetric techniques. So, as mentioned earlier, the latter approach will fit perfectly to visualize geodata, being a frequently used strategy by GIS applications. There exist two principal alternatives to render volumetric data: indirect and direct volume rendering. The first method uses algorithms to obtain and render surface information from the volumetric data, yet neglecting inner information. In contrast, the second one defines techniques to visualize the data without applying any procedure to extract its surface, just by simulating the interaction between light transport models and a 3D scalar field.

The aforementioned real-time feature in the generation of images is also an aspect very appreciated when carrying out operations on the geodata. In addition, for transient operations or simulations it would be very beneficial that both operation and visualization procedures could be synchronized, smoothly displaying the operation gradually. For instance, it may be quite interesting to see how a natural phenomenon like lava flow or sediment



deposition is progressing in a simulation. The ongoing development of hardware capabilities and the emergence of technologies such as GPU computing help to make this possible. GPU computing sets a framework to use graphics hardware to execute non-related rendering operations on it. These operations can benefit from parallel processing to speed up their execution. This technology has prompted the complete integration between visualization and computation as a reality.

### 1.1 AIMS AND OBJECTIVES

Considering the GSIS pipeline depicted in the previous section and the outlined requirements, the overall aim of this dissertation is to contribute to the improvement of some stages by researching and developing data structures and methods of interest for geoscientists. As can be seen in Figure 1.2, this dissertation will cover the geometric representation and discretization procedures within the Geometry Model submodule and its visualization at interactive response times. The objectives of this work are listed below:

- The study and development of representations and data structures to be used in the geological modeling process. The data structures proposed should meet a set of specifications, in particular a simple definition and easy implementation in a computer system, reasonable space and time requirements, and ability to manage complex 3D geological information.
- The definition of a formal framework to validate the proposed representations and define properties and algorithms precisely. This formalization must be founded on standard methods and theories from the geospatial literature.
- The description of a real-time rendering method for the visualization of volumetric terrains and geological structures, using the data structures defined previously. Additionally, the implementation of useful visual features for geoscientific applications to validate the use of the representation will be discussed.

### 1.2 ORGANIZATION OF THIS DOCUMENT

This dissertation comprises four parts organized in a set of chapters:

**PART I** This part includes the current chapter and introduces the subject to the reader. Furthermore, a review of the state of the art of geological modeling and visualization in geosciences is given in Section 2.

**PART II** Here, the core representation in which this thesis is sustained is presented: the stack-based representation of terrains, proposing its use as an all-around representation for both surface and subsurface information. Section 3 deals with the description of the data structure and highlights its advantages and drawbacks compared with other representations. In Section 4, a formal framework for the representation of 3D terrains based on stacks is presented. The framework has been derived from the geotom theory, a spatial representation that generalizes vector and raster data types. The proposed formalization is completed with a set of operations inspired by the well-known map algebra. In addition, an exchange data model is suggested as an example of implementation using a standard similar to the Geography Markup Language. Finally, Section 5 describes a novel data structure that improves the advantages of the stack-based representation in terms of storage requirements and data organization: the QuadStack.

**PART III** This part explains how the data structures introduced in Part II can be used to render 3D information in an efficient way, making use of direct volume rendering techniques. Section 6 introduces some concepts and techniques used in direct volume rendering. Section 7 describes the visualization methods for the representations introduced in this thesis, including visual operations such as the visualization of cross-sections or the inspection of internal structures, as well as vector data layers. This section ends with a description of the method used to display volumetric data encoded in a QuadStack.

**PART IV** Section 8 concludes this dissertation by highlighting the main achievements and pointing out possible improvements to be done as future work.



One of the main goals of geoscientists in conjunction with mathematicians is to represent in a computational model the complexity of the subsurface of the Earth. These models tend to unify the physical reality, geometry, topology and any kind of data related to geological objects (Mallet, 2002). Seismic data (Leeuwenburgh, Brouwer, and Trani, 2011), gravity and magnetic data (Guillen et al., 2008) or resistivity models (Foged et al., 2014) are examples of typical physical properties that should be modeled with reliability. Physical properties are described by the numerical model explained in the previous chapter, and supported by any discretization method that allows measuring. For its part, geometric and topological structures store knowledge about the representation model of the faults or horizons created from the stratigraphy and their connectivity. Focusing on every aspect of the geological modeling is out of scope of this doctoral dissertation; therefore, this section will review and discuss geological modeling from a geometric and topological point of view emphasizing the data structures used and how the geological data can be visualized by means of computer graphics rendering techniques.

With regard to data modeling and representation, most of the research carried out in the GSIS field comes from GIS development. As in GIS systems, two main data models are used: discrete objects and continuous fields. These models, often categorized within a conceptual level of abstraction, are characterized by a more concrete representation of the problem (the *logical model*), which, in turn, is implemented in a computer using different data structures (Longley et al., 2015). However, the boundaries of the terms *data model*, *representation* and *data structure* are usually fuzzy in GIScience and, in particular, in geological modeling (Arroyo Otori, Ledoux, and Stoter, 2015). Examples are the ambiguity or lack of difference between data structures and data models (Le et al., 2013; Penninga and Van Oosterom, 2008), or the use of the *conceptual model* term in a very uneven way. Traditionally, in areas belonging to computer sciences, such as database modeling, conceptual models have been used to organize and define the concepts and their relations of an actual problem with the use of schematic diagrams. In GSIS, some works insisted on this idea by

suggesting conceptual models by means of ER or UML diagrams, either conceptualizing high level data models (Kjenstad, 2013; Wang et al., 2018), or depicting more specific geological and environmental features (Pinet, 2012; Tegtmeier et al., 2014). In contrast, the term of conceptual model has also been suggested when any sketch or equation has become necessary to clarify an idea (Agterberg, 2018; Florian Wellmann, Croucher, and Regenauer-Lieb, 2012). Because of this, the delimitation of these concepts as well as the generalization of the data structures and algorithms is seen as a necessity in the geological modeling literature (Wang et al., 2016; Wycisk et al., 2009). Accordingly, this section introduces a taxonomy in which a GIS abstraction model and a classification of the data structures used to implement the different data models are explained.

Shen et al. (2013) classify the existing approaches for spatial modeling into three categories: *surface-based*, *volume-based* and *hybrid models*. This classification, while intuitive, is not very formal, since it includes within the same category data structures like octrees and data models such as 3D raster or geocellular models. Other surveys like the ones presented by Natali, Lidal, and Parulek (2012) or Galin et al. (2019) aim attention at the classification of terrain and geological models but for computer graphics applications. In order to follow a top-down approach, and inspired by the widespread geospatial research, the taxonomy presented here classifies any geological representation into three high-level models: to the common *continuous fields* and *discrete objects* models, *general models* are included as a different attempts to unify every kind of spatial information into the same representation.

## 2.1 CONTINUOUS FIELD MODELS

In continuous field-based models, a property is distributed in a continuous manner along a spatio-temporal domain. This property can be height measurements (Li, Zhu, and Gold, 2004), the variation of pollution in the air (Jjumba and Dragičević, 2015) or the distribution of geological materials or physical parameters (Pryet et al., 2011). More formally, a field model is defined by an injective function (Worboys and Duckham, 2004):

$$f: \mathbb{R}^n \rightarrow S \tag{2.1}$$

where the domain is a space of any dimension, usually the 2D or 3D Euclidean space, and the range  $S$  is the set of sampleable values.

This distribution can be categorized into two discretization approaches: *Tessellation-Based Fields* (TBF) and *Sampling and Interpolation-Based Fields* (S&IBF) (Liu et al., 2008). In TBF the whole space is discretized into tesserae of regular/structured or irregular/unstructured shapes. Examples are regular grids, triangular networks or spatial models based on Voronoi diagrams. S&IBF models define a set of geometric objects to be sampled, such as points or lines. For any  $n$ -dimensional position  $p$ ,  $f(p)$  is calculated as an interpolation of the values for the geometric elements in the neighborhood of  $p$  (Li and Heap, 2014). Regular and irregular point clouds or isoline maps are included in this category. Geological models are essentially defined by a 2D/3D tessellation of a geographical space, thus the models and data structures defined below are considered TBF approaches.

### 2.1.1 *Grid-based models and data structures in geological modeling*

Grid partition is the typical representation when modeling field-based features. Depending on the application, this approach has different denominations. In terrain modeling, where 2D regular grids are mainly used, they are called *heightfields*, *heightmaps* or *Digital Elevation Models (DEM)* (Wilson and Fotheringham, 2008). In the GIS field, the most widespread term is *2D raster*. However, this refers not only to height values, but also to any scalar value (Longley et al., 2015). In other areas with three-dimensional scope such as computer graphics, the term *voxel model* is the most used (Hadwiger et al., 2006). In geological modeling, it is named in a different way depending on the purpose of the representation. For a discrete representation of a 3D field of a given physical or chemical property, the terms *voxel model* and *3D grid/raster* are normally used. On the other hand, in numerical modeling, the so-called *hexahedral meshes* are frequently used in the calculation of finite difference simulations (Owen et al., 2017).

#### 2.1.1.1 *Structured grids for surface modeling*

A geological surface is usually described by means of the descriptions of its geometry (height values) and its geological features (material, soil resistivity, etc.). The space is partitioned into a  $w \times h$  grid in which each individual element can take values within a scalar range (e.g., height, physical properties) or a nominal one if the grid encodes discrete categories

(e.g., materials). Due to their simplicity, the amount of work devoted to the study of 2D grids, their properties and applications is huge. We can find notorious examples of the use of this data structure in applications such as surface geometry modeling (Li, Zhu, and Gold, 2004), the monitoring of the soil evolution (McBratney, Mendonça Santos, and Minasny, 2003), modeling of rockfall and hazard assessment (Lan, Derek Martin, and Lim, 2007) or representation of geological maps (Carré and Girard, 2002; Mateo Lázaro et al., 2014) among many others.

However, it is not possible to represent complex geological structures with two-dimensional grids, since more than one height value for a 2D position  $(p_{x,y})$  may be required. Voxel models are the most direct extension to 2D grids that allows this possibility. In a voxel model, the 3D space is tessellated into a  $w \times h \times d$  grid, though the set of possible voxel values is usually categorical: either a binary value (ground, air) or a geological material. Regarding the state of the art, Jones et al. (2010) presented a method for the simulation of directable weathering of exposed concave rock using this data structure. Koca and Güdükbay (2014) used voxel models in combination with levels of detail and heightmaps to create a hybrid model for representing these volumetric features. Jjumba and Dragičević (2015) suggested voxel-based automata for simulating natural processes such as the prediction of snow avalanches or the landscape evolution. The same authors computed the fractal dimension over voxel models to assess environmental three-dimensional objects like vegetation or geological formations in the landscape (Jjumba and Dragičević, 2016). Recently, an approach to generate arches, caves and other complex structures in the terrain surface from user input data has been proposed (Becher et al., 2019).

#### 2.1.1.2 *Structured grids for subsurface modeling*

At the end of the 1980s, voxel models began to replace CAD techniques for the representation of volumetric features at the subsurface level like Bezier surfaces or NURBS (Turner, 1992). Since then, voxel models have been the traditional representation in geomodeling. Many works manage geomodeling like a pipeline in which each stage receives new input data like borehole logs, cross-sections or surveys captured with Airborne Electromagnetic Methods (AEM), improving the quality of the model. In addition to the data model, interpolation methods such as Kriging, Inverse Distance Weighting or regression models are indispensable to achieve a reliable grid (Li and Heap, 2014). Mallet (1997) was one of the first researchers

who represented geological models using grids. He filled a 3D structured grid with seismic velocity data in which discontinuities corresponding to the intersections with horizons and faults were taken into account. In fact, one of the main modelers for geological applications, GOCAD, was designed by him and his team (Mallet, 1992); although, ad-hoc geomodelers have been also introduced (Hollt et al., 2012). Over time, GOCAD included three-dimensional grids, from which many other geoscientists benefited. For example, Smirnoff, Boisvert, and Paradis (2008) used a support vector machine method in combination with GOCAD to reconstruct a 3D model in a 3D grid from well data in an automatic manner. In addition, this software package has been used to measure the gravel percentage in a groundwater flow model (Bonomi, 2009), or to gather geological information from different inputs in a single voxel model (Kaufmann and Martin, 2009). Other example of a methodology developed to discretize a geological model in a 3D grid using different sources was presented by Kessler, Mathers, and Sobisch (2009). One of the most common sources to compute 3D grids are AEM surveys (Jørgensen et al., 2013, 2015). Clay fraction is also a valuable parameter used in geomodeling (Foged et al., 2014; S. Høyer et al., 2015). Lastly, voxel models have been recently used for rock tunneling applications (Cacciari and Futai, 2017).

### 2.1.2 Grids for numerical modeling

Simulations of geological processes using numerical methods usually require either a 2D or a 3D regular discretization of the space. There exist works that simulate water flow and compute drainage networks using a 2D grid (Moore, Grayson, and Ladson, 1991; Ortega and Rueda, 2010). Also in the hydrological field a 3D model for simulating sediment transport, erosion and deposition was described by Karszenberg and Bridge (2008). Hoffmann et al. (2017) presented a work that simulates the heterogeneity in subsurface models by means of image quilting techniques. Finite difference simulations usually cannot be run on triangular meshes or other data structures representing the boundaries of the geological model; therefore, methods for generating finite element volume meshes have been presented (Spear et al., 2016; Zehner et al., 2016).

Numerical models often make use of deformable voxel models known as *geocellular* models (Denver and Phillips, 1990). A geocellular model is a 3D structured grid in which a voxel does not have a cubic shape, but a more general hexahedral structure. By using this geocellular model, among other



representations, a geological model of the Italian Alps was constructed from a set of stratigraphic surfaces (Zanchi et al., 2009). Physical properties like the resistivity of geological layers were also modeled with hexahedral meshes (Pryet et al., 2011).

Nevertheless, Lie et al. (2012) claimed that reliable geological models require grids which are not forced to follow a strict arrangement and therefore, fit with the complex geometry of faults, fractures or eroded zones. On this matter, they released an open-source MATLAB toolkit that eases the creation of unstructured complex grids. Other open-source packages have been developed trying to provide a more accurate discretization tool for numerical simulations. For example, Berry et al. (2014) proposed integrating a set of structured/unstructured grids in a GIS system in order to improve the reliability and accuracy in numerical modeling.

Nevertheless, not only patterns based on hexahedral meshes exist. Ledoux and Gold (2008) presented an approach for modeling 3D geological fields through unstructured Voronoi grids and their dual in 3D, the Delaunay tetrahedralization. Sentís and Gable (2017) also introduced a work in which the LaGrit toolbox (Los Alamos National Laboratory, 2016) was enhanced with the inclusion of Voronoi meshes. Recently, unstructured prism grids have been introduced for the simulation of reservoirs with complex forms (Li, Li, and Zhang, 2018).

### 2.1.3 *Hierarchical grids for geomodeling*

Regular grids have a major memory overhead that has led to designing data structures which apply adaptive resolution depending on the heterogeneity of the data. In 3D modeling, octrees are the best-known hierarchical data structure and have been successfully adopted in geomodeling. In one of the first research works in geomodeling, Dunstan and Mill (1989) suggested the use of octrees for modeling geological structures. Later, Jørgensen et al. (2013) used octrees to model AEM data in an efficient manner. Another example is the implementation of an octree refinement algorithm for hexahedral meshes suggested by Jansen, Sohrabi, and Miller (2017). Combinations of coarse/fine grids are also common, such as the works presented by Lie et al. (2017) or Arnone et al. (2016) who used different spatial resolutions to describe terrain morphologic properties. In surface modeling, solutions which make use of mipmap pyramids have also been proposed (De Cola and Montagne, 1993; Losasso and Hoppe, 2004).

To conclude this section and, as mentioned above, the traditional approach for advanced representation of surface features and/or subsurface structures is based on heightmaps or voxel models. An integral and efficient solution for both cases consists of using a hybrid strategy: instead of storing a single value for each cell in a regular grid, a list of intervals is stored in what is called a material stack. Each stack compacts a vertical sequence of voxels with a common material or physical property. This representation, called the Stack-Based Representation of Terrains (SBRT), was originally introduced by Benes and Forsbach (2001) for the simulation of erosion processes in 3D terrains, and has since proven its usefulness in several subsequent papers (Löffler, Müller, and Schumann, 2011; Natali, Klausen, and Patel, 2014; Peytavie et al., 2009). This dissertation shows its suitability for geomodeling through a formalization of its definition, properties and basic algorithms including efficient visualization. Moreover, we will later describe an extension called QuadStack that delivers an excellent compression ratio for big datasets.

## 2.2 DISCRETE OBJECT MODELS

Geological formations can also be represented by means of their boundaries. For example, a lithological layer could be defined by a set of polygons assembled in a polyhedra, being these polygons the representation of its different surrounding horizons. This approach is flexible enough to model the complexity of the morphological features commonly found in geological data. However, in turn, the methods proposed within this approach are notably more convoluted than field-based approaches. This difficulty in the definition of the related data structures, as well as the necessity of studying the resulting properties has led to an outstanding research activity in this area.

The formalization of geological models has played a major role in the geographic information sciences and geoscientific evolution. One of the first geospatial issues addressed by the definition of formal models was the representation of spatial relations among discrete objects (implemented by vector data models in GIS terminology) such as topological, order or metric relations (Egenhofer and Franzosa, 1991). Many were the theoretical models and data structures proposed for this purpose. For example, Molenaar (1992) suggested one of the first data structures that validated this type of relations for three-dimensional data: the 3D Formal Data Structure (3D FDS). 3D FDS defines nodes, edges and arcs as basic elements, which can be

combined into surfaces and bodies as higher level objects. This work was continued by Zlatanova (2000), who presented a formal model focused on the improvement of spatial queries, and by Coors (2003) who introduced a urban model from 3D FDS among others.

This background conceived for GIS was rapidly adopted by geomodeling scientists (Mallet, 1997). Due to the high number of specific applications, many different goals have led to diverse data structures. Thus, a classification seems necessary to catalogue every emerged solution. In this part of the chapter, this dissertation will embrace the categories suggested by Arroyo Ohori, Ledoux, and Stoter (2015).

### 2.2.1 *Geological modeling with simplicial complexes*

Several approaches are based on the mathematical formulation of the cell and simplicial complexes theory. A simplex or  $n$ -simplex is the convex hull of  $n + 1$  points in a Euclidean space of dimension less than  $n$ . Hence, a 0-simplex is a node, a 1-simplex is an edge or arc, a 2-simplex is a triangle and a 3-simplex is a tetrahedron.

This kind of representation encompasses data structures such as the Triangular Irregular Network (TIN), composed uniquely by triangles, and the TEtrahedral Network (TEN), which comprises tetrahedrons. In the literature, TINs have been treated both as a field-based data structure, mainly in GIS-related works (Goodchild and Kyriakidis, 2005), and as a discrete object data structure in geological approaches (Zehner et al., 2016). This is explained by its different uses. When TINs are used as a pure surface terrain model, they fall under the definition of field, since a property, in this case the height values, can be computed as a function of a 2D spatial position and, consequently, it can be tessellated in a piecewise model. In geomodeling, TINs are mostly used to represent the boundaries of the geological bodies or to represent geological horizons, which may not map unequivocally a 2D position into a single height value (Galera et al., 2003). Since for this dissertation only the applications in the geomodeling field are of relevance, TINs will be considered as simplicial complex data structures, even if they are embedded in a dimension higher than two (Arroyo Ohori, 2016).

Lemon and Jones (2003) were among the earliest to avoid the use of CAD methods in order to build a reliable 3D geological model. They defined a set of primary and secondary TINs to define clear horizons between the geological structures. TINs have also been used for modeling free-form

stratigraphic layers (Caumon et al., 2009). A software that implements a methodology to process geological information using different sources was released by Kessler, Mathers, and Sobisch (2009): GSI3D. In their work, TINs were used to delimit what they define as *lithoframes*. Similar ideas have been presented in the works of Maxelon et al. (2009), Ming et al. (2010) or Guo et al. (2016), among others. Other software libraries provide specific meshing methods for the generation of consistent geological models (Pellerin et al., 2017). Moreover, as mentioned above, TINs have also been used for representing the boundaries of geological structures through triangular meshes (De Oliveira Miranda et al., 2014).

Tetrahedral meshes are analogous to triangular meshes, but they have the advantage of being able to model volumetric features. TEN was introduced as a 3D geological/topographic data structure by Pilouk (1996). Penninga (2008) introduced a new data structure that modeled both surface and subsurface spatial datasets, partitioning the 3D space into a set of connected 3-simplexes. Caumon et al. (2012) used tetrahedral meshes for gathering data from different sources such as remote sensing images and DEMs, and resolving the generated discontinuities. Additionally, hierarchical data structures have been introduced in order to perform efficient topological queries on tetrahedral meshes (Weiss et al., 2011). Finally, a simplex-based approach to implementing dimension-independent operations in geoscientific applications was presented by Karimipour, Delavar, and Frank (2010).

Figure 2.1.a illustrates an example of a polygon represented by means of simplicial complexes.

### 2.2.2 Geological modeling with cell complexes

The elements used to define cell complexes are somewhat similar to the primitives associated to simplicial complexes. However, the mathematical background of the former is stronger. A  $d$ -dimensional cell complex in a Euclidean space  $E^n$  is a homeomorphic image of a  $d$ -dimensional open ball, i.e., an  $i$ -cell with  $0 \leq i \leq d$  is homeomorphic to an  $i$ -ball (0-ball is a point, 1-ball is an open arc, 2-ball is an open disk, etc.). Each pair of  $i$ -cells are pairwise disjoint and they are adjacent if they share a  $k$ -face ( $k < i$ ). Given two cells, an  $i$ -cell is incident to a  $j$ -cell if the former is a face (that could have any dimension  $d > 0$ ) of the latter (Čomić et al., 2014).

Cell complexes are mainly modeled by their  $(d-1)$ -cell boundaries, while a  $d$ -simplicial complex is usually modeled by  $d$ -dimensional primitives

(for example, a volume is partitioned into a set of tetrahedrons). The most common representations of cell complexes are the incidence graph data structure (Sobhanpanah, 1989) and the Nef polyhedra (Bieri, 1995) (see Figure 2.1.b). Although there are some papers focused on geospatial modeling that use this kind of data structures (Ledoux, 2013), single-handedly cell complexes have not emerged as an interesting approach in geological modeling research. However, it is remarkable that CGAL, one of the main software libraries for programming geospatial applications, implements Nef polyhedra representation (The CGAL Project, 2019).

### 2.2.3 Ordered topological geomodeling

In GIScience, the usual manner of defining topological relationships is by means of the 9-intersection model (Egenhofer, 1989). This model describes these relationships with a matrix that is capable of representing the eight possible spatial relations between two objects: disjoint, meet, contain, inside, cover, coverby, equal and overlap. Further to this model, the point-set topological theory is introduced in order to generalize the topology of simplicial complexes (Egenhofer and Franzosa, 1991). Taking the 9-intersection model and the point-set theory as inspiration, formal frameworks for geomodeling have been proposed (Wang et al., 2016).

Ordered topological models are the result of the combination of both simplicial complexes and cell complexes methods, together with some notions of the point-set model. Each cell has a set of simplices as primitives, called vertices despite having a dimensionality higher than 0. The vertices are connected in combinatorial maps creating an incidence graph in which strong topological relations are modeled. This graph can be seen as an adaptation of the *meet* relation defined in the 9-intersection model (Thiele et al., 2016).

Two leading representations have been proposed for using ordered topological models: cell-tuples introduced originally by Brisson (1993) and Generalized-Maps or G-Maps presented by Lienhardt (1994).

Given a cellular partition, for each  $n$ -cell a set of cell-tuples can be built. A cell-tuple  $\langle c_0, c_1, \dots, c_n \rangle$  represents the topological path from an  $n$ -cell to a 0-cell. In other words, an  $n$ -cell is  $i$ -incident to the  $c_i$  elements of their cell-tuples. Each of these tuples can be resumed into an incidence graph, which is the topological representation of the cellular partition. This graph can result in two possible orientations. For example, given a 1-cell segment, it is connected (or its cell-tuple is 0-incident) to

two different 0-cell points, therefore the 1-cell would have two cell-tuples with a different orientation. G-Maps are similar to cell-tuples but with different notation terms. Likewise, vertices are the most basic elements in the structure, but they take the name of darts instead of 0-cell and the incidence relations are called involutions (see Figure 2.1.c).

Le et al. (2013) presented an object-relational data model for geosciences in which the G-map is featured as a topological data structure. Then, a system for erosion and sedimentation simulation using three-dimensional G-Maps was presented (Crespin et al., 2014). This work proved the compactness of this data structure when modeling very simple layered geological objects. Breunig et al. (2016) introduced a web-service geo-database that supports a wide variety of geomodeling data structures, including cell-tuples.

To conclude, these topological models are no totally suitable for geomodeling since the elements represented lack a specific location in space, and only their association relationships are defined. Also, in order to compose the whole model, it would be necessary to traverse the graph iteratively. Consequently, this approach is not the most feasible to perform fast geometric queries (Poupeau and Bonin, 2006).

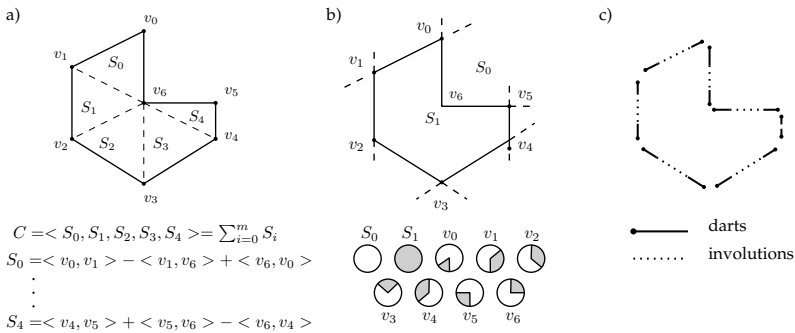


Figure 2.1: A polygon represented by means of a) a simplicial complex, b) Nef representation and c) a G-map.

2.3 GENERALIZED STRATEGIES FOR GEOMODELING

Although the solutions described in the previous sections can deal with many problems, there are situations in which both discrete objects and continuous fields are used (Brooks and Whalley, 2008; Wang, Wan, and Palmer, 2010). Different proposals that deal with these situations have been

presented, on the one hand, performing conversions between the different data (Shen et al., 2013) or, on the other hand, defining a more general framework that integrates both approaches in a hybrid one. Regarding the latter strategy, it has been a productive research field in GIScience (Gold and Mostafavi, 2000). A recurrent approach is to take the field data model as the most general one and formalize it to be able to represent discrete objects. Camara et al. (2014) proposed fields as a generic type to represent spatio-temporal data. Using a similar approach, Liu et al. (2008) introduced the concept of G-Field (from General Field). In addition, they classified different common geospatial operations and defined them in their framework. Kjenstad (2013) proposed a more general framework in which the concept of hyperfields or field-of-field structure was introduced.

The works presented by Goodchild, Yuan, and Cova (2007) and Goodchild et al. (1999) suggest the generalization of every kind of geographical information within the same framework: the geo-atom theory. This scheme reduces a spatial feature in an  $nD$  position to an atomic form. More complex elements are formed by aggregations of geo-atoms. A more thorough description of this theory will be carried out in Chapter 4, since an important part of this dissertation is based on a formalization of the previously seen SBR (Section 2.1.1) for geological modeling based on this framework.

The previous works generalize objects and fields in a broad GIS scope. Specifically, with regard to geomodeling, Wu (2004) suggested a general 3D data model: the Generalized Tri-Prism (GTP), as an improvement of a prior data model, the Tri-Prism (Gong et al., 2002). The GTP model comprises six primitives: node, TIN-edge, side-edge, TIN-face, side-face and GTP component. This model fits nicely with those data pipelines where borehole samples are the main data acquisition method, since adding a new sample is a trivial operation in the data structure. Different extensions to this model have been proposed such as one designed especially for irregular geological objects, the QTPV (from Quasi Tri-Prism Volume) (Gong, Cheng, and Wang, 2004). Even though this method seems to fall in the category of pure discrete objects, a partition of the whole space can be handled with this structure and hence, interpolation procedures may be defined over this 3D tessellation (Cui et al., 2017). Finally, an isosurface extraction method inspired by the Marching Cubes algorithm has been presented for visualization purposes (Li et al., 2018).

Figure 2.2 depicts the classification considered in this section.

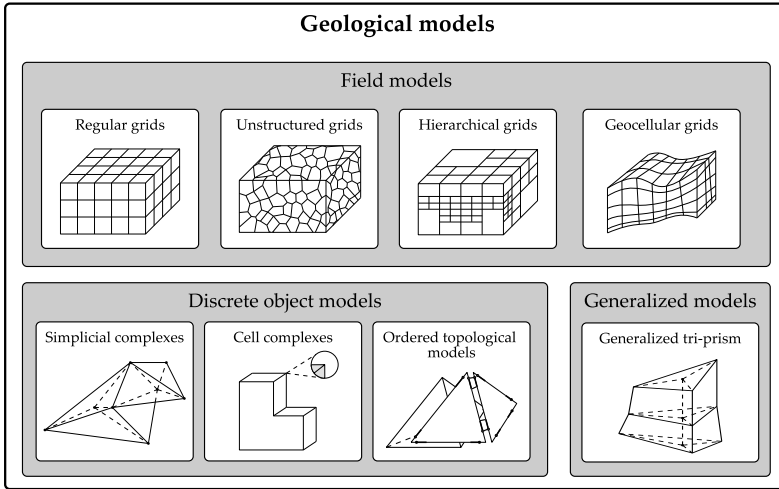


Figure 2.2: Classification of the geological data models according to the data model they represent.

## 2.4 COMPARISON OF DATA STRUCTURES FOR GEOMODELING

The aim of this section is to compare some of the different data structures given above. A representative data structure has been chosen for each data model, and they are studied in terms of topology, ease of construction and query, and storage efficiency. Voxel models, Voronoi Diagrams (VD), unstructured grids and octrees are the data structures representing the field-based data models. Regarding discrete object models, TEN data structure (simplicial complexes), Nef polyhedra (cell complexes) and G-maps (ordered topological models) will be compared. Finally, we will study the data structure proposed for the GTP model by Wu (2004) as a representative of the generalized models.

### 2.4.1 Topology

Having a topology-friendly data structure that provides direct access to connected components is a key factor when simulating geological processes. Thiele et al. (2016) described three types of topology: a basic *cellular topology*, which relates primitives of the data model, and other two higher level models: *structural* and *lithological topology*. Structural topology reduces the complexity of the cellular topology by creating connections among the different features of the geomodel, such as faults, reservoirs or



drill holes; while lithological topology stores the adjacency among lithological formations in a matrix form. Structural topology is useful only in certain problems and it is very dependent on the context, therefore the data models are designed ad-hoc to solve each particular issue (Pellerin et al., 2015). Lithological topology, by definition, neglects the spatial information; hence, it is not very interesting for this analysis. Therefore, the comparison will be done with respect to cellular or primitives' topology.

Some field-based data structures establish their topology implicitly by means of a spatial discretization. For instance, given a voxel with index  $(i, j, k)$  from a voxel grid of  $w \times h \times d$  dimension, accessing to its neighborhood is trivial. However, this topology does not provide information about connections of more generic geobodies. In order to include this information, an additional supporting data structure has to be used (Mücke, Saias, and Zhu, 1999). Nevertheless, hierarchical data structures bring a partial topological structure based on resumed zones. Neighboring zones can be reached by traversing the tree hierarchy. This implicit topology is not available when working with unstructured grids. For example, as previously stated, Ledoux and Gold (2008) endowed their data model based on VD with a tetrahedral mesh, which in conjunction with the VD structure produces topological information.

For TEN data structures, *meeting* or *touching* relationships are defined for simplices with the same dimension i.e., relations between two nodes, two edges, or two triangles. A Nef polyhedron consists of a point set formed by the combination of a finite number of open half-spaces. These half-spaces have in-out orientation, and they are gathered from set complement and set intersection operations. From these combinations, set of *local pyramids* that define the  $i$ -faces (a 0-face is a vertex, a 1-face is an edge, a 2-face is a surface and a 3-face is a volume) of the polyhedra is obtained. The local pyramids represent the neighborhood of a face and they are stored in a sphere map data structure (Bieri, 1995). Finally, the last discrete object-based data structure, the G-Map, bases their topology in a combinatorial map between two intrinsic objects: the set of darts and their involutions (the connection paths from an  $n$ -cell to a 0-cell). To sum up, only the G-Map, since it is a topological data structure, includes within its own definition topological information, the rest require auxiliary data structures in order to define it.

Finally, in GTP, the topological relationships are embodied in the data structure. This data structure can represent every potential connection among GTP primitives through indexing tables.

### 2.4.2 *Memory consumption*

Voxel models are clearly the most memory-consuming data structures, even for moderate discretization resolutions. However, the number of primitives that discrete-object data structures require to model complex geodata, and their auxiliary topological structures, can also lead to high storage consumption. Likewise, the VD and the GTP data structure also demand auxiliary indices or tables to refer to the topological relationships. Lastly, octrees, like other hierarchical data structures, are intended to compress zones with high isotropy; therefore, they are expected to consume less memory than the rest.

### 2.4.3 *Querying and updating*

It is well-known that, queries and updates can be performed in  $\mathcal{O}(1)$  in voxel models and in logarithmic time in hierarchical data structures due to their structured form. Mücke, Saias, and Zhu (1999) suggested a *walking through* algorithm to sample VD with an efficiency close to the theoretical optimal  $\mathcal{O}(\log n)$  from the dual Delaunay structure. In addition, updates can be carried out without reconstructing the entire VD. Instead, object-based data structures would need a spatial index in order to perform a random point sampling, such as a k-d tree (Hachenberger, Kettner, and Mehlhorn, 2007); otherwise, queries can be solved in  $\mathcal{O}(n)$ ,  $n$  being the number of connected primitives. In the case of Nef polyhedra, updating and carrying out operations like translations, scalings or rotations can be done in  $\mathcal{O}(n \log n)$ . GTP has an easy update mechanism when adding a new drill hole to the data, needing just local updates. If an efficient random sampling is required, an indexing data structure is mandatory.

### 2.4.4 *Ease of construction*

In order to make this comparison, three levels have been established based on the construction difficulty: easy, medium and hard. Voxel models can be tagged as the easiest data structure to be built, since this representation is the input of many other construction methods. Also, octrees can be considered straightforward to compute. The VD and its dual, and the TEN data structure are included in the intermediate category. The construction of the VD and the Delaunay tetrahedralization can be done

simultaneously thanks to their duality. The procedure of populating the TEN data structure is similar to the latter. Finally, the remaining data structures are catalogued as hard, in comparison with the previous ones, because of the amount of additional index structures that have to be built.

Table 2.1 shows a summary of the features reviewed in this section.

## 2.5 VISUALIZATION IN GIS

Geodata visualization is fundamental for most GIS applications. It helps geoscientists to discover information and take decisions at a glance. From the early adoption of CAD procedures based on the representation of curves and surfaces (De Kemp, 1999; Fisher and Q., 1992), to modern solutions based on virtual or augmented reality (Lee, Suh, and Park, 2015; Maciejewski, Pomianek, and Piszczek, 2018) or by using tangible user interfaces (Petrasova et al., 2015), GIScience has benefited from the advances in computer graphics research, specially from sophisticated and efficient methods that allow the visualization (in real-time in many cases) of the large amount of data that GIS generates.

Roughly speaking, there exist two main techniques to visualize three-dimensional models: Surface-based and volume-based techniques. The former involve the extraction and rendering of isosurfaces. An isosurface is a surface defined by the implicit function  $\phi(x, y, z) = f$ , where  $f$  is a constant scalar value, usually 0. Using this approach, 3D objects are simply represented by their boundaries. In geomodeling, these surfaces can represent horizons, faults or the Earth surface. Once the set of isosurfaces has been extracted, a set of polygonal meshes is constructed, usually triangular meshes (Lorenzen and Cline, 1987). Volume rendering techniques manage volumetric data as a 3D scalar field. Mathematically, this field can be seen as a mapping function  $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ , which is discretized in order to be represented and visualized by a computer. Chapter 6 gives a comprehensive description of this field of computer graphics, since the visualization methods presented in this dissertation are mainly based on which is known as *Direct Volume Rendering* (DVR).

The differences between these techniques remind somehow of the discrete object/field models dichotomy discussed earlier in this chapter. Therefore, it is not coincidence that most of the works devoted to model discrete objects visualize them by means of surface-based methods and that field data models are visualized with DVR algorithms.

<b>Data model</b>	<b>Data structure</b>	<b>Geological Topology</b>	<b>Ease of construction</b>	<b>Queries and update</b>	<b>Memory</b>
<b>Field-based</b>	Voxel model	Not present	Easy	$\mathcal{O}(1)$	High
	Voronoi diagram	By means of Delaunay tetrahedralization	Medium	$\mathcal{O}(\log n)$ with its dual	Medium
	Octree	Tree hierarchy	Easy	$\mathcal{O}(\log n)$	Low
<b>Object-based</b>	TEN	Tetrahedra connected by their boundaries	Medium	$\mathcal{O}(\log n)$ with an index data structure	Medium
	Nef polyhedra	Local pyramids	Hard	$\mathcal{O}(\log n)$ with and index data structure. Updates in $\mathcal{O}(n \log n)$	Medium
	G-Map	By means of an i-involutions structure	Hard	$\mathcal{O}(\log n)$ with an index data structure	Medium
<b>Generalized</b>	GTP	By means of indexing tables	Hard	$\mathcal{O}(\log n)$ with an index data structure	Medium

Table 2.1: Summary of the characteristics studied for the data structures reviewed.

Some geological features are modeled directly with polygonal meshes; therefore, its visualization is straightforward. An example of this is the work presented by Losasso and Hoppe (2004), who used traditional rasterization methods to visualize structured/unstructured grids. A mathematical framework to interpolate and extract 3D geological surfaces from multivariate data using radial basis functions was presented by Hillier et al. (2014). Surface rendering can also be enhanced by overlaying new information (She et al., 2017). Brooks and Whalley (2008) extend the typical 2D GIS layer visualization to support three-dimensional layers. That layer arrangement is depicted in the same general 3D model and visualized by means of rasterization. Song et al. (2019) proposed a work in which a combination of diverse methodologies to model and visualize geological bodies was performed. The visualization was made by rasterizing triangular meshes as well. Isosurfaces have also been extracted from gridded scalar data with uncertainty (Zehner, Watanabe, and Kolditz, 2010). Finally, many of the previous works examined in previous sections use this type of visualization technique (Breunig et al., 2016; Kessler, Mathers, and Sobisch, 2009; Koca and Gdkbay, 2014; Li et al., 2018; Zehner et al., 2016).

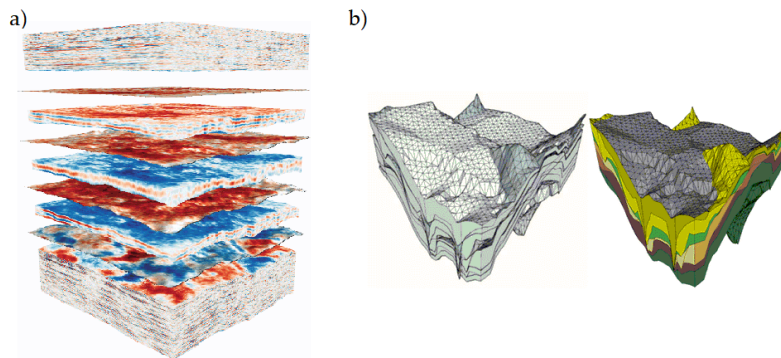


Figure 2.3: Geological models visualized by the two different approaches: a) volume rendering of subsurface seismic data (Hollt et al., 2012) and b) a triangular mesh representing a strata model (Song et al., 2019).

Regarding volume rendering methods, they can be applied to visualize in conjunction 3D and 4D environmental data. By using a multi-CPU and GPU approach, Li et al. (2013) distributed the cost of volumetric rendering of dust concentration and evolution into several processors. Hollt et al. (2012) proposed a system that generates subsurface information from seismic data. After a horizon extraction method, the generated surfaces are converted into a deformable voxel model that can be visualized with a

volumetric exploded view approach. Patel et al. (2009) combined different rendering techniques such as volume-based and texture-based algorithms to visualize subsurface seismic data. With an illustrative purpose, a geological modeling system in which a set of surfaces are organized into a 3D irregular grid was proposed. These surfaces depict several geological attributes like oil saturation, rock type, porosity or permeability (Rocha et al., 2018). There are also works focusing on the surface of the terrain that use GPU raycasting without rendering volumetric features (Ammann, G enevaux, and Dischler, 2010; De Toledo, Wang, and Levy, 2008; Mantler and Jeschke, 2006; Treib et al., 2012).

Hybrid visualization methods in which the model is rendered part by part using both approaches have gained attention in the literature. Examples are the works presented by Caumon et al. (2005) who proposed an incremental slicing algorithm and a generic data structure to render unstructured grids by means of their isosurface and applying DVR in a combined manner. Seeking a non-realistic visualization, Lidal, Hauser, and Viola (2012), presented a cutaway visualization solution for 3D subsurface models in which both polygon and volume rendering are used.

A couple of rendering examples extracted from the literature is shown in Figure 2.3.



## Part II

### DATA STRUCTURES FOR GEOLOGICAL LAYERED DATA

This part is splitted into three chapters. The first one draws the conclusions from the previous chapter and introduces an efficient data structure for geological models: the stack-based representation of terrains. The second chapter formalizes this representation in the GIScience context. This formalization defines the operations allowed with this representation as well as an implementation model for data exchange. The main contribution of the last chapter is an extension of this representation, which provides an extra level of compression by means of a hierarchical scheme: the QuadStack.





## THE STACK-BASED REPRESENTATION OF TERRAINS

---

### ABOUT THIS CHAPTER

This chapter describes the data structure on which this doctoral dissertation is articulated, and serves as introduction of this part. First, a general overview both in an intuitive and mathematical manner is provided. Afterwards, its current use in the state-of-art is presented. Then, we address its benefits and propose the use of this representation for geological surface-subsurface structures. Finally, in order to demonstrate its efficiency in terms of memory usage, a comparative assessment with traditional data structures is done. This chapter also lays the foundation for the research work carried out by our research group in the last four years and in collaboration with other international teams, with regard to terrain and geological modeling.

### 3.1 EFFICIENCY IN MODELING GEOLOGICAL FIELDS

Chapter 2 discussed how geological structures, both at the surface and subsurface levels, are typically represented by means of DEMs when just a 2.5D is required, or voxel data when modeling actual three-dimensional field data. Therefore, if we need to model complex surface features like natural overhangs or caves, or the model includes stratigraphic information or the subsurface location of groundwaters, cavities or fractures, a volumetric representation must be used. However, as noted earlier, this representation raises the problem of large memory requirements, which can be a relevant factor during the processing and visualization of high resolution models.

A more efficient representation is to extend DEMs to store in each cell a sequence of vertical intervals of the same material or attribute instead of a single elevation value. This is a straightforward way to pack stacks of voxels with a common attribute at the same  $p_{x,y}$  coordinate. This is not a novel idea; indeed Benes and Forsbach (2001) already introduced the *Stack-Based Representation of Terrains* (SBRT) in the context of modeling terrain erosion. A main strength of this representation is that it keeps the simplicity

of DEMs, making it possible to implement raster operations in an easy way. Having a simple representation that serves both for implementing raster operations on the terrain and for efficient rendering is important for many geoscientific applications.

### 3.2 RELATED WORK

So far, the stack-based terrain representation and its variations have been used in a few scientific works, mostly focusing on the visualization at the ground level. Peytavie et al. (2009) got a realistic visualization by proposing a hybrid model in which a stack-based representation (referred to as *material layer stacks representation*) serves as support for generating an implicit surface for rendering. Also, some sculpting and erosion terrain tools were added. This work was extended by Löffler, Müller, and Schumann (2011) by creating a pipeline for the acceleration of this surface generation. Their system achieves real-time frame rates in the rendering of high resolution models. Natali, Klausen, and Patel (2014) were the first to take advantage of this structure for modeling subsurface geological structures. In their work, they present a system for sketching and visualization of geomodels to help geologists to teach geological concepts: an expert sketches a series of material layers and geological elements (e.g., rivers, lakes, etc.), which are converted into multiple heightfields for rendering. Unfortunately, the use of heightfields only allows elements that can be represented in 2.5D, excluding features like overhangs, caves, aquifers or petroleum reservoirs. Similarly, Cordonnier et al. (2017) presented a system for modeling subsurface geology in an interactive way. More recently, this layered representation has been used for modeling dynamic, snow-covered landscapes (Cordonnier et al., 2018).

In the previous approaches, the stack-based model plays a secondary role, to visualize complex terrain structures or erosion processes at the ground level or as an intermediate representation generated from a logical model. In this chapter, we propose to use the stack-based model as a primary representation for geological structures at both the surface and subsurface levels.

## 3.3 DEFINING THE STACK-BASED REPRESENTATION OF TERRAINS

Conceptually, this terrain representation can be considered as a generalization of common heightfields. As described above, whereas heightfields contain a single value for each  $x, y$  coordinate position, stack-based representation stores a *stack* of intervals. Each of these intervals is formed by a start height and the attribute within it.

Phreatic level	Depth (m)	Geological section	Sample	N. Glows S.P.T.	Description of the material
3.80m (24/05/06)					(0.00-0.90m) Anthropogenic material.
	2.00		M-9	6-7-7-6	(0.9.-4.30m) High-plasticity clayey or muddy loam. The material has a beige color.
			SPT-9	5-6-6-6	
	4.00				(4.30-6.60) High-plasticity clayey or muddy loam. The material has a blue color.
	6.00		M-10	10-12-12-13	
				SPT-10	11-11-12-12

Figure 3.1: A real example of borehole log extracted from a geotechnical survey performed at the University of Jaén (Spain).

The stack-based representation of terrains is a natural representation for data generated by borehole logging. A borehole provides a top to bottom sequence of materials at a given  $p_{x,y}$  position (Figure 3.1). A common way to obtain a geomodel of the subsurface is by means of an interpolation/extrapolation procedure of the boreholes samples, obtaining a layer-cake model as a result (Turner, 2006). This generated model fits perfectly with the stack-based representation since each cell of the terrain can store a single borehole record as a stack, including both materials and height of the geological formations (e.g., water, petroleum, clay, rock, etc.) and geological properties (e.g., density, permeability, resistivity, etc.).

### 3.3.1 Mathematical definition

Given a voxel grid  $V$  with resolution  $w \times h \times d$ , each voxel  $v_{x,y,z} \in V$  stores one or more *attributes* such as color, material or density, that depend on the application. Layers are the maximal sets of connected voxels with a constant value for a given attribute.

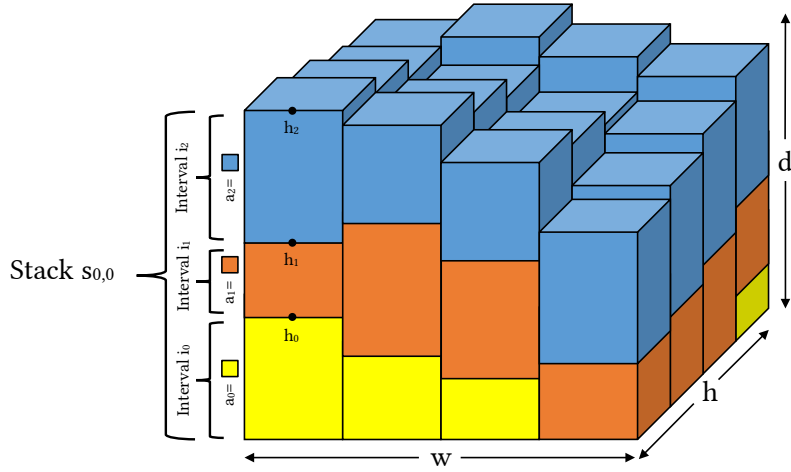


Figure 3.2: Stack-Based Representation: A voxel dataset of resolution  $w \times h \times d$  composed of layers is organized into vertical stacks consisting of intervals. Each interval  $i_k = \langle a_k, h_k \rangle$  with the attribute value  $a_k$  and height  $h_k$ .

The stack-based representation of  $V$  (see Figure 3.2) is its decomposition into a set  $S$  of vertical stacks, where each stack  $S_{x,y} \in S$  comprises the space defined by the column of voxels at position  $p_{x,y}$ :

$$S_{x,y} = V_{x,y} = \bigcup_{i=1}^d v_{x,y,i}$$

A stack is compacted as a run-length encoding of voxels with the same value for the attribute. Therefore, we define the stack  $S_{x,y}$  as a sequence of intervals  $i_1, i_2, \dots, i_n$  along the  $\vec{z}$  axis where  $i_k$  is a tuple  $i_k = \langle a_k, h_k \rangle$  that represents the space comprised by the range of voxels of the column  $V_{x,y}$  with identical attribute value  $a_k$ :

$$h_k = \begin{cases} \bigcup_{i=1+h_{k-1}}^{h_k} v_{x,y,i}, & \text{if } k > 0 \\ 0, & \text{if } k = 0. \end{cases}$$

The intervals are sorted by height in ascending order:  $h_k < h_{k+1}$  for any given  $k$  such that  $1 \leq k \leq n$ . Figure 3.2 shows an input volumetric structure encoded as a SBR.

The complexity of the SBR construction is  $\mathcal{O}(n)$ , where  $n = w \times h \times d$  is the number of voxels, because each voxel needs to be visited exactly

once. The SBR construction can be done totally in parallel, because the construction of a stack does not require information about neighboring stacks.

Table 3.1: Memory consumption comparison. The octree measurements have been taken with six depth levels.

Dataset	Voxel dimension	Memory usage (MB)			Compression ratio (%)	
		Voxel model	Octree	SBR	SBR - Voxel model	SBR - Octree
Dataset A	200 × 250 × 320	61.03	9.87	2.55	95.8	74.1
Dataset B	400 × 500 × 400	305.17	50.06	10.10	96.7	79.8
Dataset C	800 × 1000 × 800	2441.41	570.90	52.02	97.9	90.9

### 3.4 MEMORY STORAGE COMPARATIVE

Although the stack-based representation is appropriate for 3D terrain models and geological structures, it is not efficient for other kinds of volumetric data, such as medical datasets. The main problem is that such datasets may not be clearly formed by a set of horizontal layers, thus presenting quite complex and thin structures (e.g., blood vessels or nerves). As a result, the proposed representation could use even more memory than a voxel model. Moreover, volumetric medical datasets are usually acquired from CT and MRI scanners as a stack of images in which every pixel stores an intensity value coded by a 16 or 32-bit floating point value. Therefore, in order to encode then in a stack-based structure, a quantization procedure must be performed first, by labeling each intensity range with a single value. Of course, this introduces an error, and even if done carefully may discard intensity values that are important for the analysis of the images.

Despite the fact that certain properties or blended materials could also be considered continuous in natural geological formations, this representation must be discrete for compression to be effective and to be able to carry out analysis operations in an efficient way. Nevertheless, an interpolation procedure can be performed for visualization purposes.

A test to measure the memory usage of different representations, namely voxel model, octree and stack-based representation is summarized in Table 3.1. The datasets were obtained from the DINOloket database, which provides subsurface data from several spots in Netherlands (Gunnink et al., 2013). The results show a much smaller memory usage of the SBRT

compared to the rest. Our method requires 95% less space than a voxel representation. Even the memory usage of a SBRT is much lower than that of an octree, 74% less for Dataset A, 79% less for Dataset B and 90% less for Dataset C.

Figure 7.15 in Chapter 7 shows a visual depiction of the datasets used in this experiment.

## FORMAL FRAMEWORK FOR THE STACK-BASED REPRESENTATION OF TERRAINS

---

### ABOUT THIS CHAPTER

This chapter presents a formal framework for the representation of three-dimensional geospatial data and the definition of common GIS spatial operations. The main contribution of this chapter is fitting the SBRT into the geo-atom theory in a seamless way, providing it with a sound formal geospatial foundation. In addition, a set of common spatial operations on this representation are defined by using the tools provided by map algebra. More complex geoprocessing operations or geophysical simulations using the SBRT as representation can be implemented as a composition of these fundamental operations. Finally, a data model and an implementation extending the coverage concept provided by the GML standard are suggested. The results of this work have been published in *International Journal of Geographical Information Science*.

*Graciano, A., Rueda, A.J. & Feito, F.R.*

A Formal Framework for the Representation of Stack-based Terrains.  
*International Journal of Geographical Information Science*, 32(10), 2018.

### 4.1 INTRODUCTION

When we introduced the generalized strategies for geomodeling in Chapter 2, it was observed that there are many scenarios that require a general or hybrid representation that deals at the same time with field and discrete data types. In the early 1990's, Goodchild (1992) already noted the necessity to unify both data models in a general spatial representation, introducing for this purpose the geo-atom theory later (Goodchild, Yuan, and Cova, 2007).

The formal framework for the stack-based representation of terrains here presented is based on the geo-atom theory. This formalism is completed by describing a set of relevant operations that provide value to the system (Liu et al., 2008). Due to the abstraction provided by the geo-atom theory,



the framework is able to deal not only with a 3D distribution of geological materials or volumetric surface features, but also with any other physical or environmental property useful for terrain analysis or simulation. This aims to support geoscientists with a simple and well-defined model for 3D terrain and subsurface modeling.

The geo-atom is still subject of further research: in the paper presented by Jjumba and Dragičević (2015) voxel automata are used together with geo-atoms for the simulation of geospatial processes. Extending this theory, Wang, Duckham, and Worboys (2015) developed a formal framework for the modeling of movement, and Voudouris (2010) a general framework of fields to deal with uncertainty. Recently, Zhu, Kyriakidis, and Janowicz (2017) have underscored the limitations of geo-dipoles (explained below) for covering spatial patterns for multiple locations, showing some examples in which statistics operations require a more general approach.

#### 4.2 GEO-ATOM THEORY BACKGROUND

A geo-atom is a tuple  $\langle p, Z, z(p) \rangle$ , which contains information about the value ( $z(p)$ ) of a property ( $Z$ ) at a specific position ( $p$ ). A property and the set of possible values that can be obtained are sometimes referred to as the domain and its range respectively. The dimensionality of  $p$  can be arbitrary any space, therefore, a 3D space or 4D space-time are suitable to be adopted.

Geo-atoms are assembled into geo-objects or geo-fields depending on the type of information provided. Geographic data like points, lines, polygons or solids are represented as geo-objects, while geo-fields are the aggregations of geo-atoms over a space-discretized domain.

As explained in Chapter 2, the possible discretizations for a geo-field can be categorized into two groups: Tessellation-Based Geo-Fields and Sampled and Interpolated-Based Geo-Fields (Liu et al., 2008). In TBGF the sampling value obtained from a position within a domain is the value associated with the tile identified at this position. In other words, all the geo-atoms contained in a tile have the same property value. A remote sensing image that characterizes coastal zones (discretized as a regular grid of pixels) is an example of a TBGF. On the other hand, an S&IBGF uses some kind of interpolation to sample a value from a specific position. For instance, in order to retrieve the height from a 2D position in a digital elevation model (using a regular 2D array of points as discretization as exposed in Table 4.1), an interpolation value is computed using the nearest sampling points.

The most common approaches of discretization, identified by Goodchild, Yuan, and Cova (2007), are shown in Table 4.1. These approaches were introduced for a 2D domain to the best of our knowledge; nevertheless, they can be extended to any dimension.

Table 4.1: Common discretizations for geo-fields.

Discretization	Type	Objects discretized	Example
F <sub>1</sub>	TBGF	Non-overlapping polygons	Choropleth map
F <sub>2</sub>	TBGF	Triangular elements	Triangular Irregular Networks
F <sub>3</sub>	TBGF	Regular grid of cell	Digital Elevation Models
F <sub>4</sub>	S&IBGF	Irregular spaced points	LiDAR data
F <sub>5</sub>	S&IBGF	Points in a regular array	Air Pollution
F <sub>6</sub>	S&IBGF	Isolines	Topographic Maps

Geo-atoms can describe all the spatial phenomena for isolated locations such as pollution level or river networks. However, many properties need interaction among locations.

In contrast with a geo-atom which links a property to a location, a geo-dipole links two locations ( $p_1$  and  $p_2$ ), a new property ( $Z$ ) and the value of applying the property to the pair of locations ( $z(p_1, p_2)$ ). The interaction is summed up by the tuple  $\langle p_1, p_2, Z, z(p_1, p_2) \rangle$  (Zhu, Kyriakidis, and Janowicz, 2017). Geo-dipoles are very useful for terrain modeling. There are several properties very common in geoscientific applications which require a connection between locations. Problems like viewshed analysis (Zhao, Padmanabhan, and Wang, 2013), drainage network determination (Ortega and Rueda, 2010) or the assessment of optimal flood protection levels in urban flood risk management (Wang, Wan, and Palmer, 2010) can be accomplished with the use of geo-dipoles. For example, given a visibility property for a terrain surface that indicates whether a location  $p_1$  is visible from another location  $p_2$ , the geo-dipole for each point is  $\langle p_1, p_2, is\_visible, true \rangle$ . Geo-dipoles provide a generalization to several interaction models proposed in the literature, namely object fields (Cova and Goodchild, 2002), metamaps (Takeyama, 1997), object pairs (Goodchild, 1992) and association classes (Zeiler, 1999).

From these models, the concept provided by object fields is especially relevant for our framework. In an object field each point maps not to a

value but to a geo-object with the set of locations which meet a property given by a specific geo-dipole. This is interesting since many geoscientific and GIS problems do the gathering or an analysis of a set of locations that fulfill a requirement. Taking the example of the visibility property again, a geo-object  $O(p_1)$  can be generated with all the locations that are visible from  $p_1$ . This concept is also valid for a neighborhood since we can construct a geo-object with the locations of those geo-atoms that are within an area of influence.

#### 4.3 3D TERRAINS AS GEO-FIELDS

Most of the properties that can be measured in the subsurface are continuous (e.g., material distribution, terrain resistivity or erodibility) and therefore, can be represented with a geo-field. Since a geo-field can aggregate an infinite set of geo-atoms, it is necessary to define a discretization scheme in order to simplify and make the representation manageable.

In terrain and geological modeling, one of the properties that can be sampled in a geo-field is the material at a specific position. This property can be mapped by a scalar TBGF that establishes the aggregation of a set of geo-atoms characterized as:

$$\langle p, M, f_m(p) \rangle \quad (4.1)$$

where  $M = \{m_0, m_1, m_2, \dots, m_n, u\}$  is the set of materials sampled (e.g., sand, clay, air, etc.),  $u$  the material classified as unknown,  $p_{x,y,z} \in \mathbb{R}^3$  is a space location and  $f_m$  a surjective function defined as:

$$f_m : \mathbb{R}^3 \rightarrow M \quad (4.2)$$

Among the many ways to discretize a geo-field for terrain and subsurface modeling (Section 2.1.1), it is well-known that the most commonly used discretization is a 3D regular grid or voxel model in a 3D extension of the  $F_3$  type (Table 4.1). Then, given a grid with dimensions  $w \times h \times d$  and resolution  $r = (r_x, r_y, r_z)$ , each point maps to a voxel unambiguously. For each voxel, a *representative* geo-atom  $g_r$  can be defined. This geo-atom serves as input in several of the operations in our framework and may be

located at any point of the voxel. In this case, the centroid of the voxel is set as  $g_r$  (Figure 4.1).

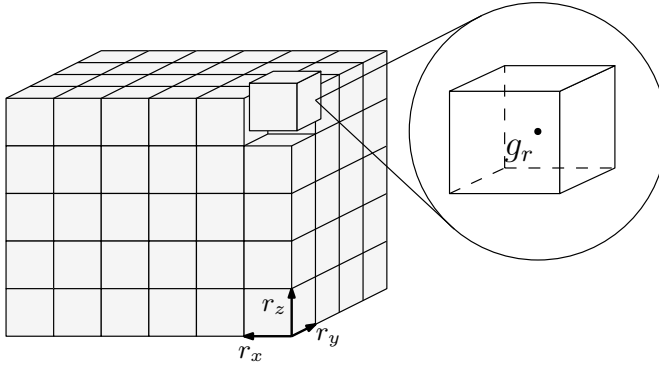


Figure 4.1: Relation between a 3D regular grid and a representative geo-atom.

A space tessellation as a regular grid is a straightforward way to conceptualize geo-fields but it can lead to implementations with high memory requirements. Therefore, the election of the stack-based representation of terrains is a perfect solution to overcome this shortcoming.

#### 4.4 STACK-BASED TERRAINS AS GEO-FIELDS

In contrast with voxel models, a SBRT discretizes the  $XY$  plane in a 2D regular grid whereas the compressed voxels are represented as a non-regular grid along the  $z$  coordinate. In this way, the minimal sampling elements (the intervals or *cuboids* due to their geometric shape) are organized in *stacks* along the  $z$  direction. Given the set of voxels packed by an interval, its  $g_r = \langle p, M, f_m(p) \rangle$  is the representative geo-atom of the upper voxel. Therefore, in the tuple  $\langle m, h \rangle$  defined above,  $m$  corresponds with  $f_m(p)$ , and the maximum height of the interval,  $h$ , is determined by  $p_z$  and the resolution of the 3D regular grid in the  $z$  direction, i.e.,  $h = p_z + r_z/2$ . As a result, in order to compute the function  $f_m$  from a 3D position, it is necessary to select a stack from the  $x, y$  coordinate to finally obtain the corresponding interval of the stack from the  $z$  coordinate. More formally, given a 3D position  $p_{x,y,z}$ , we calculate  $f_m(p_{x,y,z})$  as follows. First, a stack  $S_{x,y}$  must be selected from the  $x, y$  coordinates.  $S_{x,y}$  has to verify  $|p_x - x| \leq r_x/2$  and  $|p_y - y| \leq r_y/2$ , where  $p_{x,y}$  is the location of any of the representative geo-atoms of its intervals.  $S$  is a set of intervals  $S = \{I_1, I_2, I_3, \dots, I_n\}$  sorted by the heights of their rep-

representative geo-atoms in ascending order. Therefore,  $f_m(p_{x,y,z}) = m_i$ , being  $i$  the index of the interval that verifies  $h_{i-1} < z \leq h_i$ , with  $i > 1$ , or  $m_1$  in the case of  $h_{min} < z < h_1$ , where  $h_{min}$  is the  $z$  coordinate of the bottom face of the cuboid  $I_1$ . Otherwise,  $f_m(p_{x,y,z}) = u$ .

Following this, a SBRT geo-field can be seen as an aggregation of two geo-fields at different levels, forming a hyperfield (Kjenstad, 2013). The higher level geo-field, which represents the grid of the terrain, follows a discretization of type F3. Regarding the lower level geo-fields, namely the set of stacks, they are defined, according with an F1 discretization, as an irregular but geometrically shaped tessellation, i.e., a structured tessellation whose elements follow a pattern in their form (all of them are cuboids), but they have not the same size. Figure 4.2 shows an example of a 3D terrain modeled by voxel data, the resulting DEM and a stack of intervals. Moreover, in Figure 4.3 the contextualization of the SBRT within the geo-atom framework is represented. In this scheme, the more general data types and concepts are included in the conceptual level, while the concrete implementations given a problem are part of the logical model.

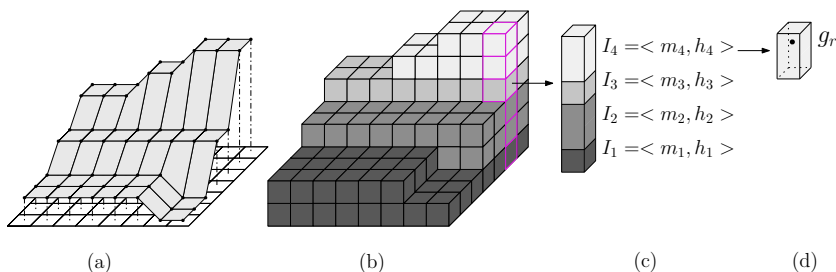


Figure 4.2: Example of a 3D terrain model using a voxel representation (b). A DEM of the terrain is showed (a), a sample stack is generated from column of voxels (c) and an interval (d) are also represented.

#### 4.5 OPERATIONS WITH 3D TERRAINS

Once SBRT has been described, this section focuses on the characterization of common operations for his representation, relying on map algebra as formal framework.

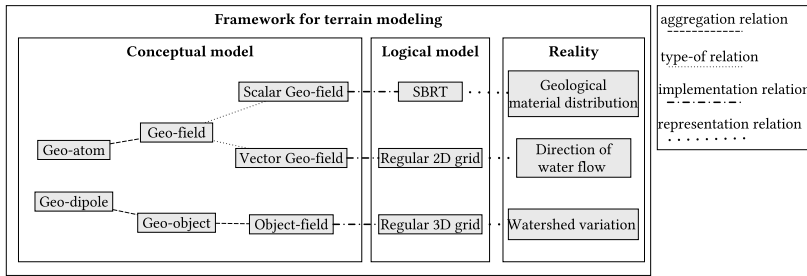


Figure 4.3: Stack-based representation of terrains in the context of geo-atom theory.

#### 4.5.1 Map algebra background

GIS and many geoscientific applications commonly require operations such as feature clipping or intersection, buffering, scale change, format conversion, distance measurement and so on. One of the reasons why researchers make efforts to improve the way in which spatial data is represented is to implement these operations in an efficient way. Traditionally, many manipulation and spatial analysis operations for fields have been defined by means of map algebra (Tomlin, 1990). Inspired by the methods of Image Processing (Ritter, Wilson, and Davidson, 1990), this defines raster layers as inputs. Map algebra allows operations such as Boolean intersections, union; arithmetic operations like subtraction or addition; statistical or relational operations.

Map algebra was originally introduced for the 2D domain. An evident improvement is to extend this framework to a 3D space and, therefore, use voxel models as operands. Mennis, Viger, and Tomlin (2005) benefited from this additional dimension to handle spatio-temporal models. Likewise, in order to obtain dynamic models, Takeyama and Couclelis (1997) merged map algebra with cellular automata developing the geo-algebra generalization. However, map algebra operations can be defined not only on regular grids. Ledoux and Gold (2008) suggest that Voronoi diagrams can be used in combination with map algebra for modeling 3D geoscientific fields.

This framework classifies spatial operations depending on their context. Initially three kinds of operations were defined: *local*, *focal* and *zonal*. In local operations (1) each cell of the new map layer is created from the gathering of overlapping isolated cells of the input layers. Consequently, regarding the number of operands, local operations can be unary as well

as n-ary operations. In contrast, in a focal operation (2) on a position  $p$ , the output cell is computed from the input cell at that position and its neighborhood from the input layers. Usually the neighborhood is based on pixel adjacency in a similar way to image convolution. Finally, in zonal operations (3) each output cell is computed from a zone of a unary input layer in a similar approach to focal operations. The zones of the input layer must be non-overlapping; therefore a cell only belongs to a single zone. Every output cell within an input zone is provided with the same value, which is the result of the aggregation of the cells inside the zone.

In addition to these operations, some authors have included global operations. These unary operations are performed on an entire layer. For instance, algorithms based on distances, such as least cost path or geometric conversions can be considered as belonging to this category (DeMers, 2002).

Liu et al. (2008) for their general field conceptualization, extended the map algebra operations by reformulating some of them and adding new ones. For instance, local operations were separated, depending on the number of input layers, into *reclassification* operations and *overlay* operations for unary and n-ary operations respectively. Also, they defined the *subset* (or *slice*) and *object identification* operations which can be used for extracting some part of a layer. In addition, an operation for the application of a *generalization* or level of detail was described.

The previous operations are irreversible since an input layer cannot be obtained from its output. The authors denoted this type of operations as Order-Increasing Operations (OIOs); otherwise, they are Non-OIO (NOIO). Operations such as geometrical transformations, exact interpolation and Fourier transformations are examples of the latter.

#### 4.5.2 Definition of operations

Although map algebra was designed taking into account 2D raster layers, its principles can be extended to the SBRT. However, not all operations make sense. Subset and object identification have the same meaning for a SBRT and therefore can be combined into a single operation. Moreover, an exact interpolation operation is obviously not appropriate because a SBRT is defined by a TBGF model.

In map algebra and its subsequent developments (Cordeiro et al., 2009; Schmitz et al., 2013), the operations receive as inputs a layer (unary operation) or a set of layers (n-ary operation) although the actual computed

Table 4.2: Examples of SBRT operations.

Operation type	Example	Output geo-field	Order
Reclassification	Binarization	Scalar TBGF	OIO
Overlay	Boolean Union	Scalar TBGF	OIO
Focal operation	Normal surface calculation	Vector S&IBGF	OIO
Zonal operation	Material percentage calculation	Scalar TBGF	OIO
Global operation	Euclidean distance to a position	Scalar/Vector S&IBGF	OIO
Subset operation	Cross section extraction	Scalar TBGF	OIO
Object identification	Material layer removal	Scalar TBGF	OIO
Generalization	Level of detail	Scalar TBGF	OIO
Geometrical transformation	Coordinate system conversion	Scalar TBGF	NOIO

elements are the individual map cells values. In 3D map algebra (Mennis, Viger, and Tomlin, 2005), the computable elements are the voxels values. Accordingly, in our framework the input elements of the operations are the elements contained in the intervals, i.e., the geo-atoms / geo-dipoles / geo-objects. For the sake of simplicity, we define and give examples of just unary and binary operations.

First of all, for a pair of intervals to be computed in a binary operation, the input geo-fields must be spatially congruent. This requirement is divided into two constraints:

1. The SBRT geo-fields must be congruent at grid cell level. This means that the implied geo-fields should have the same grid resolution. If not, a generalization operation (see below) must be performed to the geo-field with larger resolution to avoid loss of precision. If required, a readjustment of the position of the grid can be carried out to make them fit perfectly.
2. The SBRT geo-fields must be congruent at interval level. Operations between stacks have to be applied to pairs of intervals with the same position and height. Two intervals can overlap in three different ways: (1) full match when both representative geo-atom position and height match, (2) partial match when only a part of both intervals overlap and (3) full inclusion when one of the intervals is entirely included in the other. Therefore, it is necessary to split the intervals in the cases 2 and 3, in order to have the required full match before applying the operations.



Table 4.3: A conceptual view of the operations of the SBRT.



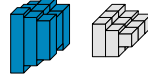

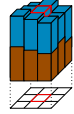


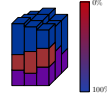


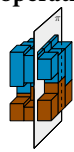







Input(s)	Output	Function
<b>Reclassification</b>		
		$f : M_1 \rightarrow M_2$
$M_1 = \{m_1, m_2, m_3, u\}$	$M_2 = \{m_4, u\}$	
<b>Overlay</b>		
		$f : M_1 \times M_2 \rightarrow M_1$
$M_1 = \{m_1, u\}$	$M_1 = \{m_1, u\}$	
$M_2 = \{m_2, u\}$		
<b>Focal operation</b>		
		$f : M_1 \times \dots \times M_1 \rightarrow M_1$
$M_1 = \{m_1, m_2, m_3, u\}$		
<b>Zonal operation</b>		
		$f : M_1 \times \dots \times M_1 \rightarrow M_2$
$M_1 = \{m_1, m_2, m_3, u\}$	$M_2 = [0, 100]$	
<b>Global operation</b>		
		$f : M_1 \rightarrow M_2$
$M_1 = \mathbb{R}^3$	$M_2 = \mathbb{R}$	
<b>Subset operation</b>		
		$f : M_1 \rightarrow M_2$
$M_1 = \mathbb{R}^3$	$M_2 = \{m_1, m_2, m_3, u\}$	

Table 3 (continuation).

Input(s)	Output	Function
<b>Object identification</b>		
		$f : M_1 \rightarrow M_2$
$M_1 = \{m_1, m_2, m_3, u\}$	$M_2 = \{m_2, u\}$	
<b>Generalization</b>		
		$f : M_1 \times \dots \times M_1 \rightarrow M_1$
$M_1 = \{m_1, m_2, m_3, u\}$		
<b>Geometrical transformation</b>		
		$f : M_1 \rightarrow M_1$
$M_1 = \mathbb{R}^3$		

An operation is specified by two components besides the input geo-fields: a function  $f$  that maps an input set into an output one and a position  $p$ . The position is used for sampling the input set and also as output location. Thus, the operation can be described in tuple notation as  $\langle f, p \rangle$ .

The input set can be formed by a single property for most unary operations or by the product set of several properties in the case of binary operations. Some unary operations in which a single object field acts as input, for instance in focal operations, also use a product set.

The nature of the geo-fields can be very diverse depending on the type of operation. The input and output sets are not limited to the SBRT encoded attributes. For example, there are several operations that perform distance measurements or geometrical transformations, which use as input the set of locations. Hence, besides sampling materials or physical attributes among other SBRT attributes,  $p$  can also be used to sample an input vector geo-field, an object-field or a geo-field that aggregates relations between geo-atoms in the form of geo-dipoles.

For the sake of illustration, several of the operations depicted in Tables 4.2 and 4.3 are described in detail below:

- **Binarization.** A change of the property measured in a location is referred as a reclassification of a geo-field. Due to the unary and local nature of the operation, it is performed independently for each interval. Therefore, the location  $p$  of the operation is the representative geo-atom of the interval and its function is defined by  $f : Z_1 \rightarrow Z_2$  that takes the property of the input geo-field ( $Z_1$ ) and map each interval to an output property ( $Z_2$ ).

A simple example for a geo-field representing a SBRT is the binarization of the materials: given the set of materials  $M$ , a function  $f$  can be defined as  $f : M \rightarrow B$  where  $B = \{\text{solid, air}\}$ . The output position  $p$  is the position of the representative geo-atom belonging to the input interval. This example can be seen in Figure 4.5.a.

- **Boolean operations.** In contrast with reclassification operations, overlay operations require a set of spatially congruent geo-fields as input in order to be performed. In this case, the function is defined by  $f : Z_1 \times Z_2 \rightarrow Z_3$ .

Boolean operations are a typical example in map algebra. Let us define a terrain subsurface dataset and a groundwater dataset, both geo-fields represented by a SBRT. Assuming that both layers are

overlapped and congruent in space, they can be combined into a single geo-field by using a Boolean operation. In this case, a *splitting* operation has to be performed over pairs of stacks. Once the geo-fields are totally congruent, the operation can be computed. The transition function  $f : M_1 \times M_2 \rightarrow M_3$  with  $M_3 = M_1 \cup M_2$  where  $M_1 = \{\text{air, ground, u}\}$  and  $M_2 = \{\text{water, u}\}$ , can be defined with the conditional rule:

$$f(m_1, m_2) = \begin{cases} m_2, & \text{if } m_2 = \text{water} \\ m_1, & \text{otherwise} \end{cases} \quad (4.3)$$

With  $m_1 \in M_1$  and  $m_2 \in M_2$ . Finally, the location of the operation is the representative geo-atom of the interval once the splitting operation is applied (Figure 4.5.c).

- **Convolution operations.** During the computation of an operation of this type at a specific position, a sampling of the neighbor locations is necessary in order to obtain the final value, i.e., a focal operation must be applied at every location  $p$ . For each location a geo-atom or a geo-dipole (if an interaction between the actual and neighbor locations is required) is generated with the new sampled value. Their locations can be gathered in a geo-object  $O(p)$  that forms a geo-field in which a location maps to a geo-object. These geo-objects are the input for the function  $f : Z_1 \times Z_1 \times \dots \times Z_1 \rightarrow Z_2$ . The computation of the normal vector of an isosurface is an illustrative example of this kind of operation. This is a common operation for both terrain analysis (for instance, for slope or drainage network calculation) and visualization purposes. In this context, we define an isosurface as the set of face portions of the cuboids associated with the intervals, which are in contact with a given material (called isomaterial). A normal vector can therefore be computed at each point of this isosurface. In many applications, it is common to use air or water as isomaterials; in these cases, the vectors are perpendicular to the plane tangent to the ground at each point. For this operation, a 3D Moore neighborhood or kernel can be adopted, aligning the point to be processed with the kernel center. Both the cell size of the kernel and its dimension are setup values chosen depending

on the desired accuracy of the method. For each neighbor location  $p_{i,j,k}$ , placed at the center of a kernel cell  $k_{i,j,k}$ , a geo-dipole  $\langle p, p_{i,j,k}, Uv, uv(p, p_{i,j,k}) \rangle$  is calculated where  $Uv$  is the property *unit vector* and  $uv(p, p_{i,j,k})$  the unit vector from  $p$  to  $p_{i,j,k}$ . A geo-object  $O(p)$  is defined as the set of locations of those geo-dipoles that point towards an interval with the isomaterial. This is the input of the function  $f : Uv \times Uv \times \dots \times Uv \rightarrow Uv$  that calculates the normal vector to the surface as the sum of these geo-dipole values (unit vectors). Formally, this function can be described by the equation  $f(f_m(p_{0,0,0}), f_m(p_{0,0,1}), f_m(p_{0,0,2}) \dots f_m(p_{n-1,n-1,n-1})) = \sum_i \sum_j \sum_k u(p, p_{i,j,k}) [f_m(p_{i,j,k}) = \text{isomaterial}]$ , being  $n \times n \times n$  the dimension of the 3D kernel. This example is shown in Figure 4.5.e and in Section 7.2.1 can be seen an implementation of this operation.

Another example is the extraction of drainage networks (Rueda, Noguera, and Martínez-Cruz, 2013). Given a SBRT geo-field constructed from a DEM ( $M = \{\text{ground, water, u}\}$ ), a new interval can be added to each stack representing a water layer. Then, a flooding simulation is performed until there is no water transfer between an interval and its neighbors. Formally, an  $S_{x,y}$  stack whose upper interval contains water, floods their neighbor stacks if its total height is higher. This procedure is repeated until a state of balance is reached. A flooding algorithm can receive as input a geo-object ( $O(p)$ ) formed with a set of geo-dipoles indicating the difference of height ( $H$ ) between an interval filled with water and located in  $p$  and its neighbors. The function is therefore defined by  $f : H \times H \times \dots \times H \rightarrow H$ . Figure 4.4 depicts a step of this operation. Notice that in contrast with the original implementation described in (Rueda, Noguera, and Martínez-Cruz, 2013), an adaptation of the algorithm to SBRTs could also deal with underground water.

- Level of detail.** The application of a generalization, also called Level of Detail (LoD), is a fundamental operation in GIS packages. It is required as a previous step for an  $n$ -ary operation as we stated before, but it also has applications as a compression method and for visualization. The result of this operation is a geo-field representing the same dataset but with a different cell resolution (only  $r_x$  and  $r_y$  are modified). The cells of the new grid can overlay one or more cells from the input grid. In the first case (the output cell overlays one

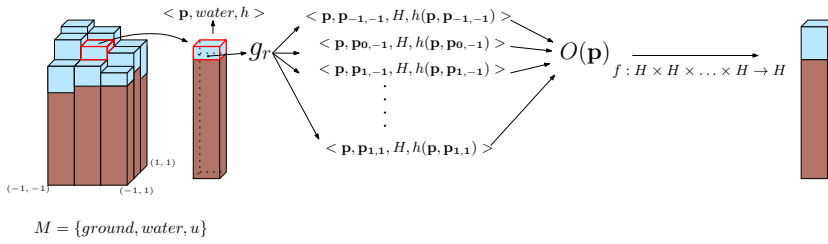


Figure 4.4: Computation of the water transfer to a stack from its neighborhood in a flooding algorithm.

single cell), the new cell will take as value the input stack. Otherwise, a splitting operation must be carried out with the overlapped cells in order to combine their stacks using, for instance, a weighted sum. To illustrate this, in Figure 4.5.d a geo-field is subjected to a generalization operation.

#### 4.6 AN IMPLEMENTATION MODEL

Section 4.5 gives a conceptual overview of the proposed representation for 3D terrains. In this section, we discuss an implementation model based on the *de facto* standard Geography Markup Language (GML) (OGC, 2007a).

The development of an independent information model that provides an adequate exchange format and eases the reuse of the spatial information is a current trend in GIScience. Studies concerning the management and integration of geoinformation in civil engineering projects have revealed this: the exchange of the information and its monitoring are troublesome because of the use of non-standard data models file formats and tools (Tegtmeier et al., 2009). To overcome this issue, there are efforts to release application-independent open standards to manage geospatial information, such as those led by the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). For this reason, the first step to define our implementation model is choosing one of the published standards as a starting schema to be augmented with our conceptual proposal.

The ISO and the OGC have collaborated in the creation of several geospatial standards such as the ISO 19107:2003 "Geographic Information - Spatial Schema" (OGC, 2003), which provides the conceptual foundations previous to GML (ISO 19136:2007) or the ISO 19123:2005 "Geographic Information -

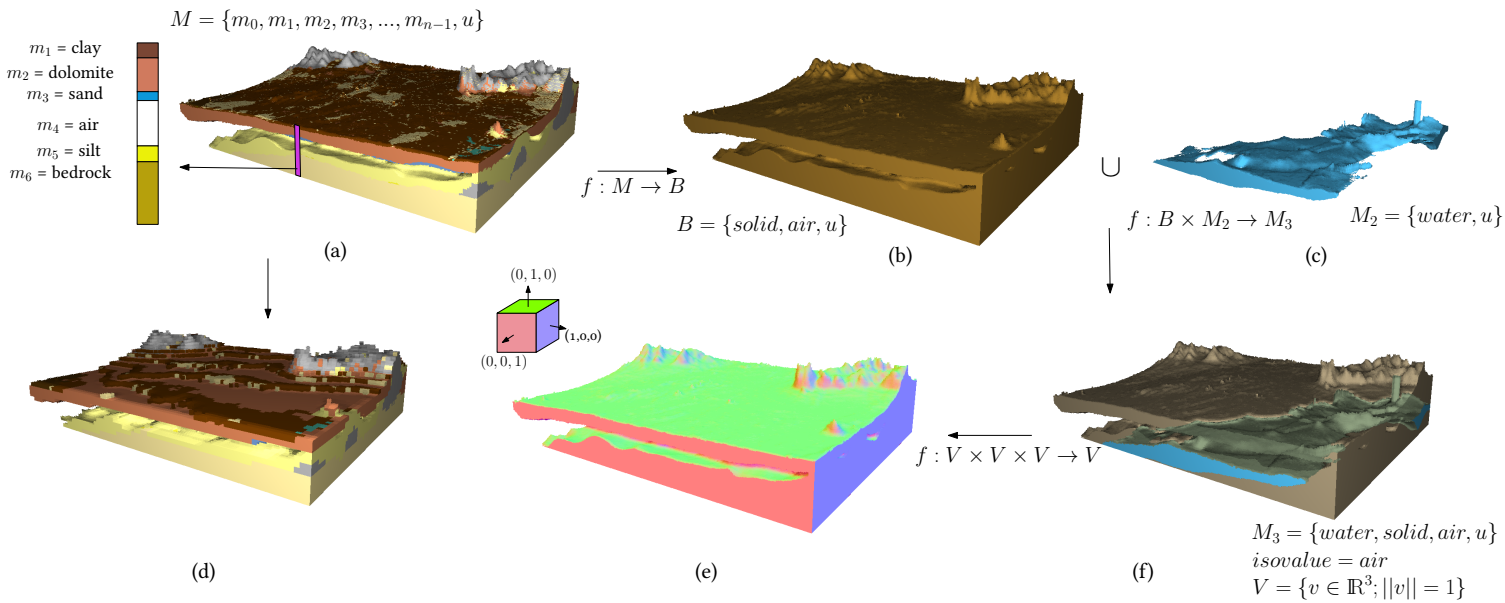


Figure 4.5: Examples of operations with a SBRT. A pipeline of operations is applied to an original SBRT geo-field.

Schema for Coverage Geometry and Functions" (OGC, 2007b). The latter document defines the conceptual architecture for fields, referred to as *coverages*, and the implementation of this abstract specification by means of GML. We use the more recent proposal of coverage implementation (OGC, 2015) rather than the original document because it fits better with the hyperfield concept (coverage partitioning). In addition, this Coverage Implementation Schema (CIS) adds some features to previous coverage specifications. For instance, it is not only XML-compliant, but it also considers an extension for a JSON encoding, which is planned to be included in future releases. Moreover, it provides new modeling capabilities, such as the addition of point cloud fields or a distinction between topological and geometric dimension of a grid.

Although this schema is appropriate for modeling the proposed representation, it needs to be extended with some concepts such as stacks or materials. GML does not supply a formal mechanism for the inclusion of new semantic features; nevertheless, CityGML (Gröger and Plümer, 2012) provides the inclusion of domain-specific information by two procedures. In the first approach, new classes and attributes can be represented by using its generic module. This mechanism does not augment the model in a formal manner, limiting the semantic and syntactic interoperability. The second approach, called Application Domain Extension (ADE), consists of the definition of an additional schema including in it the new classes that are not yet modeled in CityGML. We have taken this second approach in order to enlarge the GML schema with the SBRT classes.

Brink, Stoter, and Zlatanova (2013) proposed several alternatives for modeling ADEs using UML. They suggested that the set of new subclasses should be added to the ADE package, inheriting them from the actual GML classes, but excluding them from the generated XML schema. In our case, the omission of the new classes in the XML schema is irrelevant since that choice was made in order to allow the extension with other ADEs, keeping the same semantic schema. We follow this approach by creating a package with the inherited SBRT classes.

Tegtmeier et al. (2014) proposed the use of CityGML rather than GML for including surface as well subsurface geological objects in an integrated 3D geomodel as an ADE. However, the proposed model does not take into account the concept of coverage partitioning and misses the interpolation procedures, thus only TBGF can be represented. Finally, the XML schema can be automatically generated using a tool that implements conversion rules from UML classes and relations to GML (Portele, 2018).





abstract class. Moreover, this class can include a set of metadata including descriptive information about the coverage.

In order to model the higher level geo-field of a SBRT (the 2D grid in the XY plane), we use the *coverage partitioning* feature in combination with a regular grid coverage (class `CIS::Grid`) which acts as domain set. In the grid feature a Spatial Reference System (SRS) must be defined as well as the number of axes or dimensions, two in this case, and other axis-related data. Coverage partitioning has a set of sub-coverages, which represent each stack of the terrain. Every partition must have the same or lower dimension than the main coverage. Our representation complies with this requirement since the stacks are one-dimensional. An irregular grid through the `CIS::GeneralGridCoverage` class models the actual lower level geo-field. In order to represent the intervals of the stacks, we use the `CIS::Displacement` class. This class models a partition in the grid axes by storing a sequence of the valid positions in this domain. Therefore, the relative positions of the intervals can be encoded in this class. Similarly to the previous field, the data regarding axes and dimension must be set for these fields.

Figure 4.6 shows a simplified UML diagram of a possible implementation of our framework using CIS and GML. Nodes in gray belong to GML/CIS framework, while yellow nodes are custom classes of our framework and Figure 4.7 shows a small encoding example of a naive SBRT in GML. The SBRT contains a  $2 \times 1$  support grid with two intervals for each stack.

#### 4.7 CONCLUSION AND FUTURE WORK

In this chapter, we have presented a formal framework for the representation of 3D terrains based on stacks. We have built our framework around the geo-atom theory, making use of many concepts like geo-dipoles, geo-fields, geo-objects and object-fields. Also, we have defined a set of operations for our system inspired by the well-known map algebra. Due to these sound foundations, many other high level operations such as the removal of layers of materials or the calculation of a line-of-sight graph can be added to the framework in a straightforward way. In addition, we have suggested an example of how our representation can be implemented using a standard as GML.

In this chapter, we have solely focused on field data type, and more specifically in 3D terrains. However, several geospatial problems require processing field and discrete data in a combined way. For example, sur-

```

<SBRT:StackBasedTerrain>
  <domainSet>
    <SBRT:RegularGrid dimension="2">
      <CIS:Axis axisLabel="Lat" lowerBound="0" upperBound="2" resolution="1"/>
      <CIS:Axis axisLabel="Long" lowerBound="0" upperBound="1" resolution="1"/>
    </SBRT:RegularGrid>
  </domainSet>
  <metadata>
    <CIS:Extension materialSet="ground 0, water 1, undefined u"/>
  </metadata>

  <CIS:Partition>
    <CIS:EnvelopeByAxis>
      <CIS:axisExtent axisLabel="Lat" lowerBound="0" upperBound="1"/>
      <CIS:axisExtent axisLabel="Long" lowerBound="0" upperBound="1"/>
    </CIS:EnvelopeByAxis>

    <coverage>
      <domainSet>
        <SBRT:IrregularGrid dimension="1">
          <CIS:DistortedAxis>
            <CIS:Axis axisLabel="Height" lowerBound="0" upperBound="5"/>
            <CIS:Displacement directPosition="0, 2.5"/>
          </CIS:DistortedAxis>
        </SBRT:IrregularGrid>
      </domainSet>
      <materialSet> 0 1 </materialSet>
      <GML:CoverageFunction>
        <sequenceRule Linear </sequenceRule>
        <startPoint> 0 </startPoint>
      </GML:CoverageFunction>
    </coverage>
  </CIS:Partition>

  <CIS:Partition>
    ...
  </CIS:Partition>
</SBRT:StackBasedTerrain>

```

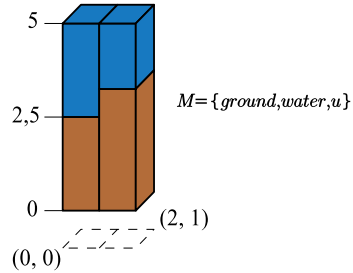


Figure 4.7: Use of GML for encoding a SBRT.

veying engineers may require terrain data modeled as a field as well as discrete objects representing pipelines or tunnels in order to perform a cost analysis. This cannot be considered either a field-related operation or an object-related operation, but a hybrid operation. The inclusion of discrete objects in our system can be easily accomplished since they are also supported by the geo-atom theory. Thus, as future work we plan to develop a whole 3D GIS for the management, analysis and visualization of not only volumetric terrains but also discrete objects, extending the theoretical model presented for terrains in this dissertation, to deal with problems involving hybrid data. In addition, since our formal framework allows its use in an  $n$ D dimension, dynamical phenomenon (or 4D) can be represented. For this reason, we are also working on the implementation of several physical-based simulation operations, such as erosion caused by water or air or in sedimentary evolution models. These algorithms can be

designed as an extension and combination of the operations shown in this chapter, being their efficiency similar to the operations implemented by map algebra. For example, the efficiency of updating a stack in a dynamic local operation such as the evolution of drainage networks is  $\mathcal{O}(n * k)$ ; where  $n$  is the neighborhood size and  $k$  the number of intervals in one stack. Eventually, the system will be released as open-source in order to benefit GIS professionals and geoscientists.

Furthermore, we can enhance our framework by allowing the representation of heterogeneous materials. Material distributions are in fact heterogeneous by its very nature since a material is usually mixed with those around it. For instance, in the transition between water and clay materials, there are points that present different proportions of them. The representation of this type of materials is a complex task since the proportions of the different mixes of materials must be managed in a proper way. This feature would provide a more accurate representation and modeling of the strata at surface and subsurface levels.

Finally, the exchange model can be implemented also in well-known formats in other fields than GIScience. For example, NetCDF (Network Common Data Form) is a set of interfaces to store and exchange array-oriented scientific data, making it suitable for geoscientific field data structures. This is widely used in engineering applications, due to its flexibility and interoperability with other formats and because it is an OGC standard.



## EFFICIENT REPRESENTATION OF STACK-BASED INFORMATION

---

### ABOUT THIS CHAPTER

The work here presented has been done in collaboration with the High Performance Computer Graphics Laboratory of the Purdue University (USA) and the Czech Technical University in Prague. This collaboration was established during my PhD stay in the Purdue University. The idea behind this work emerged after the observation that many scientific datasets present a layered structure, not just the geological information. The data structure introduced in this chapter, together with a direct rendering algorithm (presented in Chapter 7, discussed later in this dissertation) has been submitted for its publication in *SIGGRAPH Asia 2019* conference.

*Graciano, A., Rueda, A.J., Bittner, J., Pospíšil, A. & Benes, B.*  
QuadStack: An Efficient Representation and Direct Rendering of Layered  
Datasets.  
*SIGGRAPH Asia, 2019. (Under review).*

### 5.1 INTRODUCTION

Three-dimensional geological models not only present a stratified structure prone to a vertical compaction. We can also benefit from their anisotropy and high coherency in the XY plane. Figure 5.1 shows this stated coherency in large zones in which the same geological material is distributed. This continuity in the material distribution is given at sub-surface level as well, since even though certain volumetric materials are not composed of clearly visible layers; they include organized stacks of uniform material. Nevertheless, the use of a stand-alone 3D hierarchical data structure such as an octree performs worse than a simple stack-based representation as shown in Chapter 3. Therefore, based on this pair of observations, we can extend a two-dimensional hierarchical data structure in order to identify and represent layered structures at the internal nodes, and try to benefit from both vertical and horizontal coherence. In this

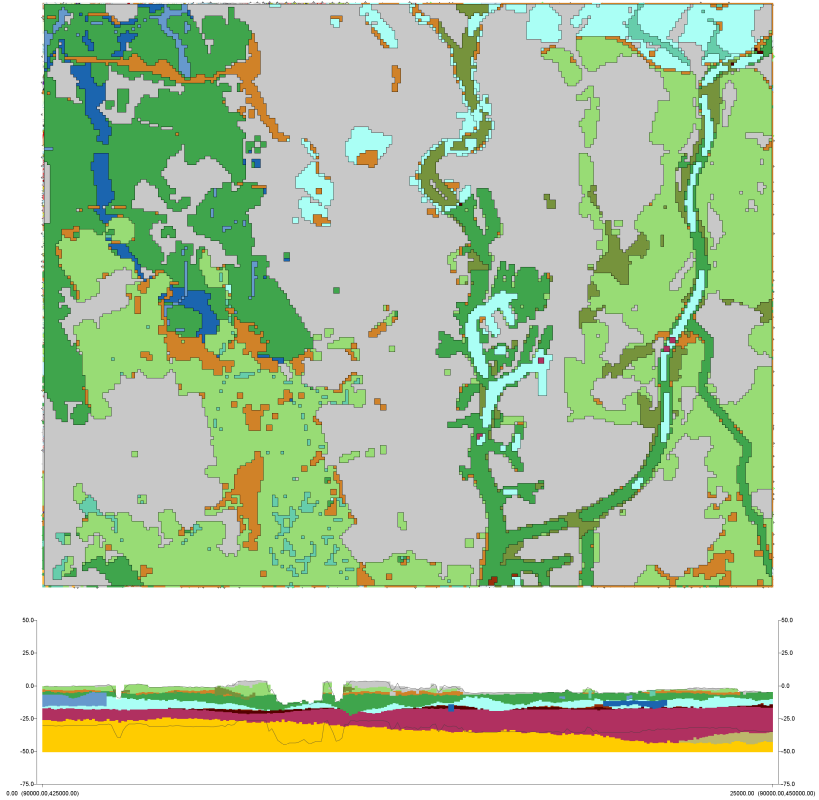


Figure 5.1: Vertical and horizontal spatial coherence in a same geomodel. Screenshot taken from SubsurfaceViewer software (Terrington et al., 2009). The geological data have been obtained from (Gunnink et al., 2013).

way, we can obtain a data structure more compact than the stack-based representation.

In this chapter, we introduced the efficient *QuadStack*, a novel data structure for volumetric data based on the stack-based representation. *QuadStack* uses a quadtree for data representation while efficiently encoding the layers in the tree.

Once a geomodel is encoded with this data structure, the data structure decouples its property values from their height values. We also introduce an algorithm for value retrieving from the *QuadStack* representation and we show that the access requires  $\mathcal{O}(\log(n) + m)$  time, where  $n = w \times h$  for stack-based representation support grid and  $m$  is the maximum stack size. In practice,  $m$  is small compared to  $n$  in a layered model so the method can be assumed to run in logarithmic time.

In Chapter 3, we discussed why the stack-based representation is not an efficient solution for some scientific volumetric data. However, this does not imply that only geomodels are arranged in a layered fashion; for example, many biological materials such as skin or leaves are also layered, but on a much smaller scale.

## 5.2 THE QUADSTACK DATA STRUCTURE

A simple approach to compressing a SBR data would be to use a hierarchical data structure such as a quadtree that would efficiently encode the neighboring stacks with the same sequence of intervals, i.e., stacks that have identical attribute values and heights. However, variations in the interval heights are common and they lead to low compression rates due to the high number of tree subdivisions.

Our observation is that while stacks differ significantly in their heights values, their attribute values do not change very often between neighboring stacks. This change happens only when a layer disappears or a new layer appears as can be seen in the top-left image in Figure 5.2. Therefore, our approach is to pack groups of neighboring stacks with an identical sequence of attribute values into a single structure denoted as group of stacks or *gstack*.

### 5.2.1 *Group of stacks*

Having the decomposition of the input volume  $V$  into stacks  $S$ , a *gstack*  $G$  represents a rectangular region of  $S$  with the same number of intervals  $n$  and identical sequence of attributes values  $a_1, a_2, \dots, a_n$ . More specifically,  $G$  represents the stacks  $S_{x,y}$  of a rectangle  $[x_{\min}y_{\min}, x_{\max}y_{\max}]$  of  $S$ , where  $1 \leq x_{\min} \leq x \leq x_{\max} \leq w$  and  $1 \leq y_{\min} \leq y \leq y_{\max} \leq h$ . Since the attribute information of the stacks  $S_{x,y}$  is identical,  $G$  can be encoded in a compact way as a sequence of intervals  $i_1, i_2, \dots, i_n$ . Each interval  $i_k = \langle a_k, H_k \rangle$  contains the attribute value  $a_k$ , common to all the intervals  $i_k$  of  $S_{x,y}$ , and a heightfield  $H_k$  with dimensions  $(x_{\max} - x_{\min} + 1) \times (y_{\max} - y_{\min} + 1)$  that stores the heights of these intervals. More specifically, the height  $h_k$  of  $S_{x,y}$  is mapped to the height  $h_{x-x_{\min}, y-y_{\min}}$  of  $H_k$ . A consequence of the fact that the number of intervals and attribute values are consistently ordered for all stacks  $S_{x,y}$  is that the heightfields  $H_k$  never intersect i.e., given  $h_{i,j,k} \in H_k$  and



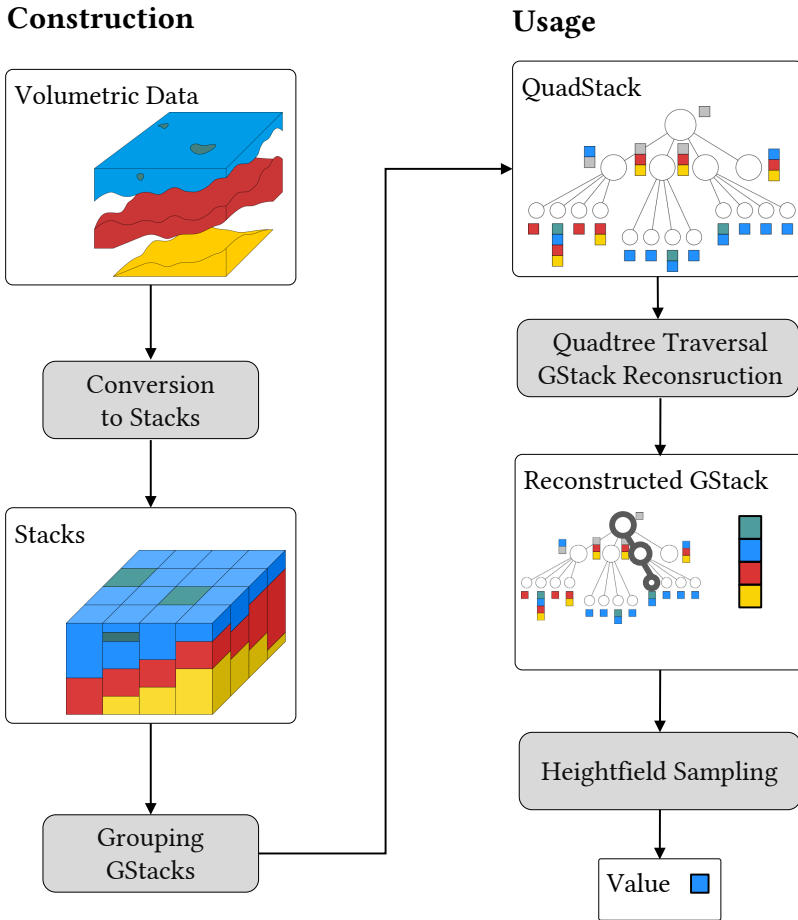


Figure 5.2: Overview: The input volumetric data is converted to stacks and similar stacks are then grouped into groups of stacks (gstacks) that are organized in a quadtree. When using the QuadStack the quadtree is traversed first and the topology of the given part of the volume is reconstructed. Then the corresponding layer boundaries encoded as heightfields are sampled to determine the result.

$h_{i,j,k+1} \in H_{k+1}$ , where  $1 \leq k \leq n$ , the condition  $h_{i,j,k} < h_{i,j,k+1}$  is always met.

A *gstack* is a simple and space-efficient encoding for a group of stacks with identical attribute information. Although only attribute information is compressed, the geometry information is stored as a set of non-intersecting heightfields that can also be compressed by using any existing heightfield encoding method.

*Gstacks* are built during the construction of a *QuadStack* that combines the spatial decomposition of a quadtree with the compact representation for groups of similar stacks given by *gstacks*.

### 5.2.2 Group of stacks hierarchies

A *QuadStack* represents the stacks as a hierarchy of *gstacks*. It divides the volume in the direction of  $\vec{x}\vec{y}$  recursively until a quadrant can be represented by a *gstack*. These quadrants are not guaranteed to be squared or a power of two, since there are no restrictions in the dimensions of the volumetric space.

Moreover, a *QuadStack* stores information not only in leaf nodes, but also in internal nodes that further improves the compression. An internal node can contain a *gstack* grouping intervals common to all its descendants. Since these intervals are not necessarily consecutive, the gaps between them, corresponding to one or more intervals that are stored elsewhere (i.e., in a descendant or ancestor node) are represented with a new type of interval called *wildcard interval* or *\*-interval*. This enables a flexible form of *gstack* that combines intervals (hereinafter referred to as *terminal intervals*) with areas without information at this level of the *QuadStack*, in many cases corresponding to intervals with different attribute information that could not be directly represented by the *gstack*.

Figure 5.3.a shows a 2D depiction of a SBR and its corresponding *QuadStack*, represented as a binary tree. The *gstacks* in nodes  $n_1$  and  $n_2$  encode the stacks as intervals of different attributes with their associated heightfields. Figure 5.3.c shows an equivalent *QuadStack* where the blue intervals, common to nodes  $n_1$  and  $n_2$ , has been stored in the *gstack* at  $n_0$ , together with a *\*-interval* that represents the rest of intervals of the block of stacks (intervals orange and yellow). On the other hand in the *gstacks* of the leaf nodes the blue interval is replaced by *\*-intervals*, since it is already stored in an ancestor.

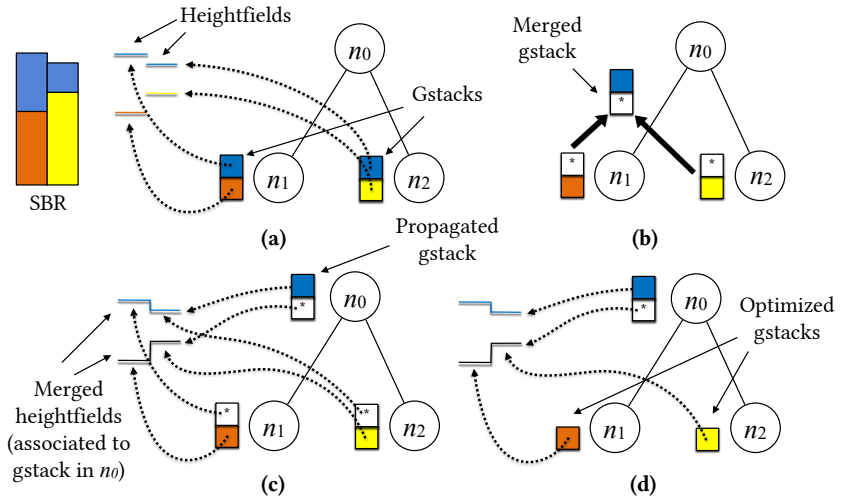


Figure 5.3: Details of the QuadStack construction: initial construction as a quadtree encoding groups of stacks with the same sequence of attributes (a), merging gstacks in nodes  $n_1$  and  $n_2$  into a gstack with common intervals and \*-intervals (b), propagation of the new gstack to the parent node  $n_0$ , restructuring heightfields (c) and optimization, deleting \*-intervals in nodes  $n_1$  and  $n_2$  associated to the intervals propagated to the parent node (d).

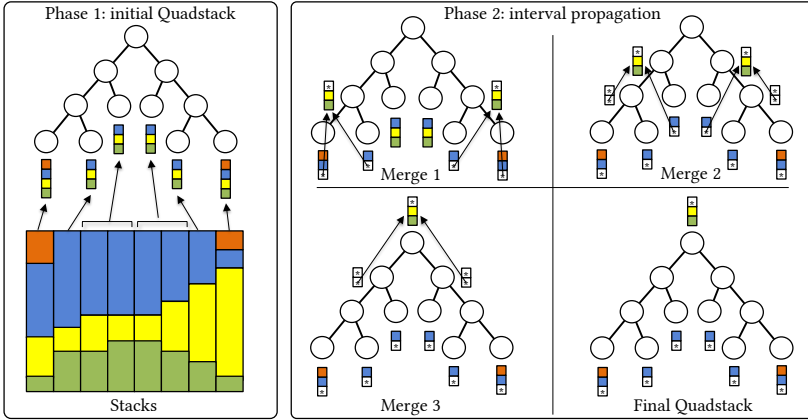


Figure 5.4: The QuadStack construction: The initial QuadStack is a quadtree that is optimized in the second step by merging intervals of equal or similar attributes.

Just like a terminal interval, an  $*$ -interval has an associated heightfield. Let  $G$  be a gstack stored in node  $n$ , and let  $i_w \in G$  be an  $*$ -interval, defined as  $i_w = \langle *, H_w \rangle$ . If  $i_w$  represents the intervals  $i_k, i_{k+1}, \dots, i_{k+j}$  of gstack  $G_d$  in a descendant  $n_d$  of node  $n$ , then the heightfield  $H_{k+j}$  of the top interval  $i_{k+j}$  corresponds to a quadrant of  $H_w$ . In this way,  $H_w$  combines the heightfields of the top intervals of all the sets of intervals existing in descendant nodes which are grouped by  $i_w$ . This is depicted in Figure 5.3.c as the heightfield associated to the  $*$ -interval in  $n_0$ . If the  $*$ -interval in  $G$  refers to a group of intervals in a gstack  $G_a$  from an ancestor  $n_a$  of  $n$ ,  $H_w$  corresponds to a quadrant of the heightfield  $H_{k+j}$ , since  $G_a$  covers a larger part of the  $XY$  plane than  $G$ . This case corresponds to the heightfields of the  $*$ -intervals in nodes  $n_1$  and  $n_2$ .

### 5.2.3 QuadStack construction

The QuadStack is constructed in two steps: a top-down *subdivision* and a bottom-up *merging*. The subdivision step is a standard quadtree construction for the stacks by using the criterion of same attribute sequence. This criterion ensures that the blocks of stacks at each leaf node can be encoded as a gstack, and the resulting data structure is already a QuadStack. Figure 5.4 illustrates the resulting QuadStack.

Although the first step already generates a more compact representation for the volumetric model than a SBR, layers of common attributes lead to duplicate intervals in many leaf nodes, as the blue, yellow, or green attributes shown on the left part of Figure 5.4. The second step extracts and merges these duplicate intervals in ancestor nodes. It proceeds from bottom to top by propagating to a node every interval common to all its children (i.e., having the same attribute value). We map the attribute values of the intervals of the four gstacks in the children to a common sequence of attribute values, using  $*$  to group non-matching attributes (Figure 5.5).

We are interested in mapping with the highest number of terminal intervals. The search space can be very large and our problem is related to finding common motifs with gaps, with applications in text mining and the analysis of DNA sequences. Many solutions have been proposed (Antoniou et al., 2006, 2007; Iliopoulos et al., 2005) that assume certain restrictions (e.g., motif size, maximum gap size, etc.) and require an exact match of the motifs or accepting certain degree of similarity.

We propose a fast brute-force solution for this problem. The input are two gstacks and the output are two gstacks with a matching sequence of interval attribute values. The algorithm takes every possible pair of intervals from the first and second gstack and tests if their attribute values match; if a matching pair  $i_{1,s}, i_{2,t}$  is found, the intervals are added to their corresponding result gstack and the function calls itself with the remaining intervals  $i_{1,s+1}, \dots, i_{1,n}$  and  $i_{2,t}, \dots, i_{2,m}$ . If  $i_{1,s}$  and  $i_{2,t}$  are not in the first positions of their gstacks (i.e.,  $s > 0$  and  $t > 0$ ), the predecessors intervals  $i_{1,1}, \dots, i_{1,s-1}$  and  $i_{2,1}, \dots, i_{2,t}$  are grouped into two  $*$ -intervals associated to the heightfields of the top intervals  $i_{1,1}$  and  $i_{2,1}$ . The number of terminal intervals of the solution obtained is computed, and finally the best solution is returned. Generalizing this solution to the four children of a QuadStack node is straightforward. The theoretical complexity of this algorithm is  $\mathcal{O}((m!)^4)$  for four stacks with  $m$  intervals, but it performs much better in practice with the heuristics described in Section 5.2.5.

If the derived gstacks have at least one terminal interval, they are merged into a single gstack at the parent node (Figure 5.3.b-c). The heightfields of the newly created gstack are generated by packing the four heightfields of the intervals of the derived gstacks. Finally the derived gstacks are deleted, and every propagated interval is converted to an  $*$ -interval at the children gstacks, grouping adjacent intervals if necessary (Figure 5.3.c). During this

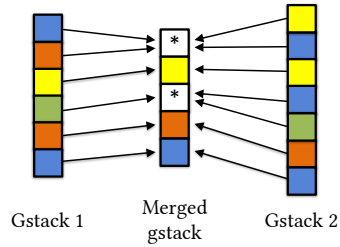


Figure 5.5: Finding a common mapping for two gstacks that maximizes the number of terminal intervals.

process, if a gstack ends up as a single  $*$ -interval, it can be safely deleted from the node since it does not provide any information.

The time complexity is  $\mathcal{O}(n \log n)$  for the initial quadtree construction and  $\mathcal{O}(n \times (m!)^4)$  for the interval propagation phase, where  $n$  is the number of stacks in the SBR ( $n = w \times h$ ) and  $m$  the maximum number of intervals in a gstack.

#### 5.2.4 Heightfield compression

QuadStack implements a compact encoding of the ordered sequence of attributes, but does not deal with the compression of the interval heights. In our approach attribute and heightfield representation are decoupled, therefore heightfields can be stored in a raw form, or compressed by any existing method such as the algorithm of Franklin (1995).

A simple delta encoding provides good results, since height values usually vary progressively. However, the main problem is that the access to any data requires decompressing the whole dataset that limits the practical use in applications where efficient queries or traversal are required (e.g., real-time visualization). Other approaches are based on image compression techniques, but again, most of them do not allow a random access (Hussain, Al-Fayadh, and Radi, 2018). In the field of texture compression, most of the solutions are based on a hardware block encoding. This approach features selective zone decompression without the need to decompress the entire texture (Munkberg et al., 2006; Nystad et al., 2012). We based on this idea to design our heightfield compression algorithm.

We propose a method that provides a trade-off between compression and access time and it is inspired by the work of Andújar (2010). This method partitions the heightfield into equal-sized blocks and compresses the values

in each block independently. Consider a heightfield  $H$  partitioned into  $w \times h$  blocks with the same dimensions  $m \times n$ . For each block  $B_{i,j} \in H$  the lower height value is taken as the base value, encoding the  $m \times n$  elements in the block as differences from this base value. Using blocks of a relatively small size makes these differences close to zero, allowing them to be encoded with a reduced number of fixed bits. This enables random access to a particular location with only two reads (difference stored at the location and base value of the block). Other predictors for a value in a block are possible: for instance it can be encoded as the difference with respect to the bilinear interpolation of the height values at the four corners of the block.

### 5.2.5 Optimizations

The QuadStack construction algorithm described in Section 5.2.3 generates a compact representation of the attributes of the model that can be improved if certain  $*$ -intervals that do not provide useful information for sampling operations are removed. Good candidates are the  $*$ -intervals representing information that can be found elsewhere in an ancestor node, such as, the top  $*$ -intervals in the gstacks of nodes  $n_1$  and  $n_2$  in Figure 5.3.c. The information represented by these intervals is included in the blue interval in the gstack of the node  $n_0$ . These intervals are never reached during sampling (Section 5.3), and can be discarded during the bottom-up phase of the QuadStack construction algorithm. When an interval is propagated to an ancestor (Figure 5.3.b-c), it is completely removed from the initial gstack, instead of converted to an  $*$ -interval. Note that the result is no longer a gstack since it represents only a subset of intervals instead of the full range of the stacks. We refer to this as *partial gstack*. Under the described conditions, gstacks can be converted into *partial gstacks* reducing the QuadStack space requirements without affecting its performance in sampling or rendering operations. Figure 5.3.d shows the resulting QuadStack after this optimization.

An optimal match of the gstacks during the bottom-up phase is the key for a good attribute compression. The algorithm `matchGS` (Algorithm 1) finds the optimal matching, although its time complexity is high. We use two efficient heuristics to reduce its computation time:

1. The exploration of a new solution can be avoided if the minimum of the lengths of the two lists of intervals to explore is less than the

best score so far, since a better score cannot be found. Notice that the score of a solution is the number of terminal intervals, so at least two lists with a higher number of intervals are required to be able to find a better solution.

2. A cache for precomputed values accelerates the function dramatically, since during the search for the optimal solution certain sub-lists are explored repeatedly. We use a simple map for this purpose, with a key computed from the length of the two sub-lists of intervals.

Table 5.1 shows the matching computation time before and after including these two improvements.

Stack size	Optimized matching (s)	Non optimized matching (s)
30	0.026	0.764
40	0.317	173.640
50	3.085	>1 hour
60	8.799	-
70	51.320	-

Table 5.1: Comparison between the optimized and non optimized versions of the gstack matching.

### 5.3 QUADSTACK SAMPLING

The QuadStack decompression can be performed selectively by using *point sampling* of the QuadStack representation. Algorithm 2 shows the regular structure of a point sampling procedure in a hierarchical data structure, adapted to a QuadStack. Querying a point  $p$  is carried out by recursive traversing the quadtree data structure. First, an inclusion test between  $p$  and the bounding box of the data must be computed. If the test is successful, the query can start by sampling the root node. In order to sample a node, the heightfield  $H_k$  of each interval in the gstack must be sampled at the  $p_{x,y}$ , comparing its height with the  $z$  coordinate. The iteration stops when the lowest interval which height is above  $z$  is found. If it is a terminal interval, regardless whether a leaf node is reached or not, the attribute is returned. When an  $*$ -interval is found, the traversal continues in the successive nodes.



**Algorithm 1:** Algorithm matchGS.

---

**Input:** gstacks  $G_1^i = \{i_{1,1}, \dots, i_{1,n}\}$  and  $G_2^i = \{i_{2,1}, \dots, i_{2,m}\}$ .  
Interval  $i_{i,k} = \langle a_{i,k}, H_{i,k} \rangle$  where  $a_{i,k}$  and  $H_{i,k}$  are its attribute value and heightfield respectively.

**Output:** gstacks  $G_1^o$  and  $G_2^o$  with the same sequence of attribute values.

```

if  $G_1^i \neq \emptyset$  and  $G_2^i \neq \emptyset$  then
  |  $G_1^o \leftarrow \{\langle *, H_{1,1} \rangle\}$ 
  |  $G_2^o \leftarrow \{\langle *, H_{2,1} \rangle\}$ 
else
  |  $G_1^o \leftarrow \emptyset$ 
  |  $G_2^o \leftarrow \emptyset$ 
 $sc^{best} \leftarrow 0$ 
foreach interval  $i_{1,s}$  from  $G_1^i$  do
  | foreach interval  $i_{2,t}$  from  $G_2^i$  do
  | | if  $a_{1,s} = a_{2,t}$  and  $a_{1,s} \neq *$  and
  | | |  $(s > 1 \text{ xor } t > 1)$  and
  | | |  $(s < n \text{ xor } t < m)$  then
  | | | |  $G_1^r, G_2^r \leftarrow \text{matchGS}(\{i_{1,s+1}, \dots, i_{1,n}\},$ 
  | | | | |  $\{i_{2,t+1}, \dots, i_{2,m}\})$ 
  | | | |  $sc \leftarrow \text{numTerminalIntervals}(G_1^r)$ 
  | | | | if  $sc > sc^{best}$  then
  | | | | |  $sc^{best} \leftarrow sc$ 
  | | | | | if  $s > 0$  then
  | | | | | |  $G_1^o \leftarrow \{\langle *, H_{1,1} \rangle\} \cup \{i_{1,s}\} \cup G_1^r$ 
  | | | | | |  $G_2^o \leftarrow \{\langle *, H_{2,1} \rangle\} \cup \{i_{2,t}\} \cup G_2^r$ 
  | | | | | else
  | | | | | |  $G_1^o \leftarrow \{i_{1,s}\} \cup G_1^r$ 
  | | | | | |  $G_2^o \leftarrow \{i_{2,t}\} \cup G_2^r$ 
  | | | end
  | end
end
return  $G_1^o, G_2^o$ 

```

---

The overall time complexity is  $\mathcal{O}(\log n + m)$  since in the worst case it is necessary to reach a leaf of the tree to retrieve the interval, checking at most  $m$  intervals during the traverse. Point sampling can also be easily generalized to decompress an arbitrary rectangular region or box of the model into stacks or further, into voxel data.

---

**Algorithm 2:** Function `sampleQS`.
 

---

**Input:** node  $n$  to be sampled; sampling point  $p$ .  
**Output:** final node and attribute value sampled at  $p$ .

```

 $a_r \leftarrow \text{null}$ 
 $n_r \leftarrow \text{null}$ 
if  $p_z > 0$  then
  foreach interval  $i_k = \langle a_k, H_k \rangle$  from gstack  $G$  in  $n$  do
    if  $h_{p_x, p_y} \geq p_z$  then
       $a_r \leftarrow a_k$ 
       $n_r \leftarrow n$ 
      break
    end
  if  $r = *$  then
     $C \leftarrow \text{getChildren}(n)$ 
    if  $C \neq \emptyset$  then
      Given  $C = \{c_0, c_1, c_2, c_3\}$ 
      if insideQuadrant( $c_0, p$ ) then
         $a_r, n_r \leftarrow \text{sampleQS}(c_0, p)$ 
      else if insideQuadrant( $c_1, p$ ) then
         $a_r, n_r \leftarrow \text{sampleQS}(c_1, p)$ 
      else if insideQuadrant( $c_2, p$ ) then
         $a_r, n_r \leftarrow \text{sampleQS}(c_2, p)$ 
      else if insideQuadrant( $c_3, p$ ) then
         $a_r, n_r \leftarrow \text{sampleQS}(c_3, p)$ 
  return  $a_r, n_r$ 

```

---

## 5.4 CONCLUSION

This chapter introduced QuadStack, a novel and efficient data structure to represent layered datasets such as geological models. The key inspiration for our work is the common output of many science and engineering applications and measurements that produce data with strong directional anisotropy in form of layers. QuadStack compresses the layers into stacks

and then compresses the stacks into a quadtree while considering the representing patterns among neighboring layers.

The third part of this dissertation provides a GPU implementation for real-time rendering of this data structure. For a performance evaluation in terms of both compression and sampling speed, the reader is referred to Chapter 7.

While the field of data compression and rendering has been active for many years, there are still many open where QuadStack could be used. Our data structure could be applied to different fields, such as BRDF compression, or time-varying datasets. Also, many datasets are cylindrical and it would be an interesting extension to apply QuadStack to a non-linear domain. We have not fully explored the internal structure of the layers and its relation to the compression factor. It would be possible to first sample and rotate the input data to detect a direction that would provide good compression factor. The construction algorithm uses rather simple matching and a possible extension would improve its efficiency for scenes with many layers. A possible solution is to accelerate the matching algorithm by using any kind of heuristics to help its convergence to a nearly optimal solution, or by using specialized data structures for this type of operations such as suffix-trees.

## Part III

### VISUALIZATION OF GEOLOGICAL DATA

This third part of the dissertation is devoted to rendering methods and how they can be applied to visualize geological information. The first chapter gives an introduction of some terms used in direct volume rendering. The second chapter presents the contributions of this dissertation in the field.



## 6.1 INTRODUCTION

Visualizing a high volume of geological information is a challenging task. In the field of computer science, there has always been a conflict between the space requirements and the execution speed of an algorithm, since the improvement of one is only at the expenses of the other one. Therefore, designing methods with a good space-time trade-off is one of the yearnings of computer scientists. We have already demonstrated the efficient use of memory of stack-based representation and QuadStack. Thus, our next objective is the implementation of efficient algorithms capable of visualizing the data stored in these data structures in a direct manner, without sacrificing its compact representation for geological data. Because we deal with volumetric field data, the most straightforward solution is through the use of DVR techniques as stated in Chapter 2.

## 6.2 OVERVIEW OF DIRECT VOLUME RENDERING TECHNIQUES

The basic idea behind DVR techniques is to model the light propagation when it interacts with a 3D scalar volume. DVR algorithms must evaluate the light model at each visible position of the volume, which results in a computational intensive task. Traditionally, this fact has been considered as an important drawback of this kind of methods. However, recent hardware advances have overcome this limitation, enabling fast implementations of this method. Taking as a reference the book published by Hadwiger et al. (2006), this section provides a brief theoretical background of the volume rendering topic.

### 6.2.1 *The volume-rendering equation*

Depending on the kind of data encoded in the 3D volume, the light can interact taking into account three different phenomena. If the material is mostly gas (1), an emission of light increasing its radiance can

occur. Conversely, the material can absorb the light (2) incident over the data, reducing its energy. Finally, the light can be scattered (3) by participating media. These light sources can come from inside or outside the volume. Usually, in DVR this complex interaction is simplified using a mixed absorption-emission approach. In scientific visualization, where photorealistic rendering is not a priority, only a single light source is considered. These features are modeled by the so-called *volume-rendering equation*, whose integral form is shown in Equation 6.1.

$$I(D) = I_0 e^{-\int_{s_0}^D k(t) dt} + \int_{s_0}^D q(s) e^{-\int_s^D k(t) dt} ds \quad (6.1)$$

$I_0$  is the light radiance entering the volume at position  $s_0$ , while  $I(D)$  represents the radiance leaving the volume at position  $D$ . The terms  $k$  and  $q$  represent the optical material properties. These terms simulate the absorption and emission part respectively. The emission coefficient is associated with the transparency of the material ( $T$ ), thus the equation can be reformulated as follows:

$$I(D) = I_0 T(s_0, D) + \int_{s_0}^D q(s) T(s, D) ds \quad (6.2)$$

The volume-rendering equation cannot be solved analytically in a computer; therefore, a numerical approximation is necessary. Setting  $s_n = D$ , we can set an interval  $s_0, s_1, s_2, \dots, s_n$  in which a number of partitions of the integral domain is split. Each of the intervals represents a location where the model will be evaluated or sampled. As a result, the discrete form of the volume-rendering equation can be written as:

$$I(D) = \sum_{i=0}^n c_i \prod_{j=i+1}^n T_j \quad \text{with } c_0 = I(s_0) \quad (6.3)$$

The number of partitions must be chosen carefully. A subsampled reconstruction will result in artifacts such as Moiré patterns while supersampling may considerably affect the performance. Usually, the Nyquist-Shannon theorem is applied to sample a signal accurately. This theorem states that, given a signal to be reconstructed, the sampling rate must be at least twice as high as the maximum frequency of the signal. In a DVR context, the

maximum frequency ( $f_m$ ) is a 3D vector in which each component corresponds to the minimum resolution in each dimension: the resolution of the grid cell for  $x$  and  $y$  components and the minimum relative height from among all the stacks for  $z$  component. Therefore, in sampling mode the step frequency ( $f_s$ ) will be  $f_s \geq 2f_m$ .

### 6.2.2 *Transfer functions*

A transfer function (TF) is a scalar function that maps the encoded volumetric data to its optical properties. The TF can be seen as a lookup table in which  $k$ ,  $q$  (combined as RGBA tuples) coefficients or Phong parameters are stored for each possible material. The TFs can be classified on the basis of its dimensionality, style of rendering or the amount and complexity of the parameters encoded. In the algorithms presented in this chapter, we simplify this function by means of a color lookup table since we encode the data as discrete values (i.e., materials). The color palette of the TF has been chosen in order to provide a good contrast among the materials and provide a visual clue to the kind of material (e.g., sand in yellow or water in blue tones). The work of (Ljung et al., 2016) provides a great revision of the TF research.

### 6.2.3 *Direct volume rendering approaches*

Equation 6.3 can be computed by using different DVR methods that can be grouped into image-space or object-space approaches. In the former, the volume is sampled by taking the image pixel as the origin of the traversal method and in the latter the 3D object is projected directly into the final image following different strategies. The most important methods are outlined below.

**TEXTURE SLICING** This object-space method intersects a set of parallel 2D slices with the 3D volume and apply a compositing order to obtain the final rendered image. This is a considerably efficient method, however, it presents the drawback that it can only be used with regular grids.

**SHEAR-WARP VOLUME RENDERING** Likewise the previous method, a set of 2D slices are placed in the scene but the samples are not taken orthogonally to the image plane if the object is rotated. In this case, the



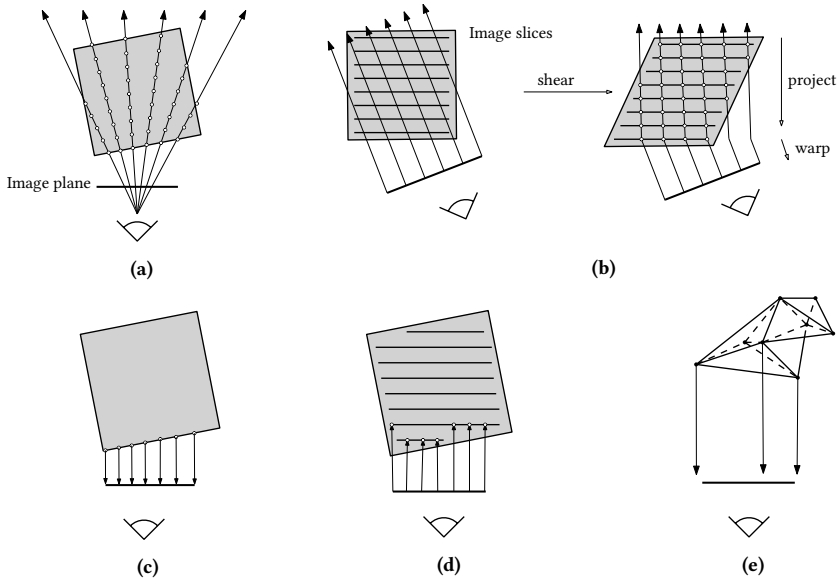


Figure 6.1: DVR approaches. Raycasting (a), shear-warp volume rendering (b), splatting (c), texture slicing (d) and cell projection (e) are conceptualized.

volume is sheared, then sampled and finally the viewed points are warped to the final image plane.

**SPLATTING** Here, each volumetric element of the grid is projected or *splattered* into the image plane. In this object-space algorithm, a Gaussian kernel is usually projected with every splat to supply surrounding information.

**CELL PROJECTION** This algorithm is used to traverse more complex grids than voxel models like tetrahedral grids. By using this method, a set of triangles is projected onto the final image blending it in an ordered manner.

**RAYCASTING** This image-space method is the most popular of the DVR algorithms. The visualization methods introduced in this thesis are based on this strategy; hence, this method will be explained in more detail in the next section.

Figure 6.1 sketches the differences between these algorithms.

## 6.3 RAYCASTING

The key idea of this well-known algorithm is to cast rays iteratively from the virtual camera through every pixel in the framebuffer, combining the color values sampled along the ray. Raycasting can be implemented in both CPU and GPU, however the GPU-based solution is more efficient, enabling real-time rendering of large datasets. For this reason, we decided to implement our visualization algorithms using the graphics processor.

The main steps of this rendering method are explained below.

**GEOMETRY SETUP** Rendering starts by sending a simple geometry to the GPU, which serves as start and exit points of the rays. Often, as in our case, the geometry sent to the GPU is the bounding box of the volumetric dataset. This approach can lead to the problem of *empty space skipping*: A significant number of rays may pass across empty space (i.e., air, transparent material, etc.), penalizing the performance without any effective contribution to the final result. Because of this, several alternative approaches replace the bounding box with a tighter bounding geometry (Hadwiger et al., 2006; Knoll et al., 2011). We decided to use the bounding box in order to keep the simplicity of the model, and also because it fits the shape of a volumetric terrain fairly closely. In next chapter, we propose partial solutions for each data structure that minimize the effects of the empty space.

**DATA ACCESS** This step is directly influenced by the data storage layout and the memory structures chosen. The particularities of each proposed method are discussed in Chapter 7. Once the data value has been retrieved, we need to compute the pixel color by means of the TF.

**COMPOSITING** The color computed at each iteration of the loop has to be combined with those calculated in previous iterations. Essentially, the composition (also called alpha blending) in DVR can be accomplished in a front-to-back or in a back-to-front order. In the algorithms described in this chapter, the former was used since it is more suitable to optimizations such as *early ray termination*

Equations 6.4 and 6.5 correspond with the front-to-back and back-to-front approaches, respectively.  $\widehat{C}_i$  and  $\widehat{T}_i$  are the values of color and transparency to be calculated for the next iteration.  $C_{i-1}$  and  $T_{i-1}$  are the values accumulated so far and  $C_i$  and  $T_i$  the new values obtained from

the TF. These equations are easily derived from Equation 6.3 by renaming  $T_i$  as  $(1 - \alpha_i)$ .

$$\widehat{C}_i = C_{i-1}(1 - \alpha_i) + C_i \quad \widehat{T}_i = T_{i-1}(1 - \alpha_i) \quad (6.4)$$

$$\widehat{C}_i = C_i(1 - \alpha_{i-1}) + C_{i-1} \quad (6.5)$$

**ADVANCE RAY POSITION** As explained in Section 6.2.1, the sample rate must be carefully selected. By means of adaptive samplings (Section 7.2) or the use of hierarchical data structures (Section 7.3) we implemented this step in an accurate and efficient way.

**RAY TERMINATION** In the methods here presented, the criterion used for an early ray termination is based on the composed opacity. If the opacity is above some threshold  $(1 - \epsilon)$  we can assume that new contributions to the alpha blending will be irrelevant, and therefore the traverse is interrupted.

REAL-TIME RENDERING OF STACK-BASED DATA

---

## ABOUT THIS CHAPTER

This chapter brings together our research in rendering geospatial information. This work has been carried out in different phases, covering the four years of my thesis project. The first part of this chapter is devoted to the development of a visualization algorithm for the stack-based representation, introduced in Chapter 3. Likewise, the second part is the complement of Chapter 5 and presents a method to visualize layered data encoded by a QuadStack. The results of this chapter have been published in different papers.

*Graciano, A., Rueda, A.J. & Feito, F.R.*  
Real-time Visualization of 3D Terrains and Subsurface Geological Structures.  
Advances in Engineering Software, 115, 2018.

*Graciano, A., Rueda, A.J., Ortega, L. & Feito, F.R.*  
Towards a Hybrid Framework for the Visualization and Analysis of 3D Spatial Data.  
In Proc. 3rd ACM SIGSPATIAL UrbanGIS'17 Workshop 2017.

*Graciano, A., Rueda, A.J. & Feito, F.R.*  
Direct Volume Rendering of Stack-based Terrains.  
In Proc. of CEIG 2017.

## 7.1 INTRODUCTION

As already mentioned in Chapter 2, several geoscientific visualization research works carry out a visualization based on triangle meshes. The most usual technique for generating a triangle mesh from a volumetric dataset is *Marching Cubes* (MC) (Lorenson and Cline, 1987). This algorithm and all its variants generate a set of triangles which represents an isosurface from a given isovalue. Real-time visualization of cross-sections or a specific stratum/material would require generating multiple meshes by applying

MC on each isosurface as preprocessing. This may result in redundant geometry and in a model that would take up much more space than a SBRT. For instance, Figure 7.10, later in this chapter, illustrates how complex a surface/subsurface model can be, and the amount of triangles which would be required to render each feature. Therefore, the use of DVR is more suited for the visualization of geological models.

In the previous works that previously used the SBRT, the stack-based model serves as an auxiliary structure, mainly as an intermediate data structure for visualization purposes. One of the main contributions in this chapter is the direct visualization of surface and subsurface information encoded using a stack-based representation. The proposed solution provides real-time rendering of volumetric terrains and subsurface geological structures using a compact data representation for the stacks in the GPU (Section 7.2). A further variation is suggested then by extending this algorithm to handle a direct rendering of datasets stored in a QuadStack (Section 7.3).

## 7.2 RAYCASTING THE STACK-BASED REPRESENTATION

In raycasting a ray sample is translated to a data texture/buffer access. In our method, first the position  $(p_{x,y})$  is obtained by projecting the ray to the terrain grid. Therefore, a single stack is obtained from this position. Then, the final value is sampled by iterating on the stack attributes. Depending on the sampling ray position, the stack is iterated starting at the top, if the ray is above the bounding box equator, or at the bottom, if it is below. This trivial optimization improves the performance by 15% on average. DVR methods usually store volume data in a 3D texture (or in a 2D array texture). However, there are multiple possible strategies to encode a stack-based representation of a model in GPU memory, as we discuss in Section 7.2.2.

Once the value is sampled, we need to obtain a mapping between it and a color by using the TF. An illumination model can also contribute to the final color in order to simulate the light interaction and increase realism. Our method uses a deferred strategy: In a first pass only the value of the TF is retrieved. Then, in a second pass a normal vector is estimated from the depth buffer information, and the final pixel color is calculated (Figure 7.1). We use a Lambertian diffuse model for the illumination according to Equation 7.1. Components  $i_a$  and  $i_d$  control the intensity of ambient and diffuse lighting respectively,  $k_d$  is the diffuse Lambertian reflectance,  $\vec{L}$  is

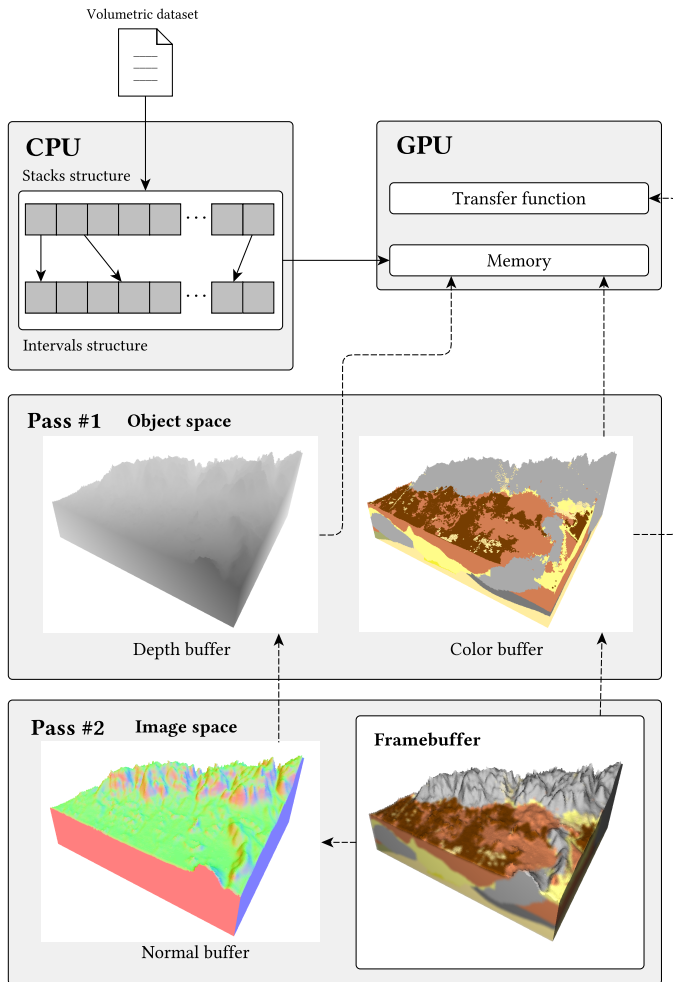


Figure 7.1: Deferred shading strategy used. The dotted lines indicate an usage relationship.

a direction vector from the object to the light source and  $\vec{N}$  is the normal vector to the surface in a point  $p$ .

$$I_p = i_a + k_d(\vec{L} \cdot \vec{N})i_d \quad (7.1)$$

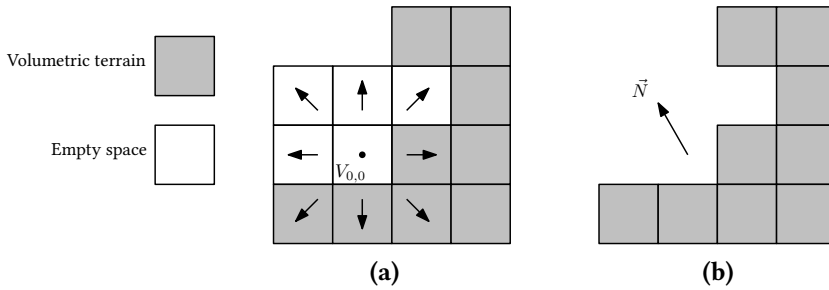


Figure 7.2: Estimation of a normal vector in an object space.  $\vec{N}$  (b) is the resulting from the sum of the vectors contained in empty space voxels by using a  $3 \times 3$  kernel (a).

### 7.2.1 Surface normal vectors calculation

Normal vectors are usually computed by simulating a density gradient, but this solution requires volume data representing intensity values, as for instance in medical imaging. Much of the literature focused on the calculation of surface normals for discrete volume data takes an image space approach (Yagel, Cohen, and Kaufman, 1992) (Kadosh, Cohen-Or, and Yagel, 2003). Surface normals are typically obtained by calculating the horizontal and vertical gradients for each pixel locally from the depth buffer data. The resulting shaded models using this approach might not present a smooth surface, since a voxel may cover many pixels if the model is near to the virtual camera. In contrast, an object space approach provides a better approximation of the surface since it takes into account the actual geometry of the model. For this purpose, a discrete voxel model can be represented as a binary voxel model simply by labeling each voxel as occupied or empty space. A straightforward method to obtain the normal is to convolve a  $k \times k \times k$  kernel at the current position where the ray is located ( $v_{0,0}$ ). A voxel  $v_{i,j}$  of the kernel contains the unit vector direction from  $v_{0,0}$  to  $v_{i,j}$ . The final normal vector  $\vec{N}$  is calculated by adding the vectors associated to each voxel whose center is in an empty space (see Figure 7.2). The time consumed by this method as well as the smoothness of the surface is strongly determined by the kernel size chosen. The larger the kernel size, the smoother the surface, but more computation time is required due to the increase in the number of data accesses.

The approach we propose is hybrid: we calculate the normal vector with the convolution method described above but in image space rather than in object space. Instead of calculating the normal vector in the raycasting

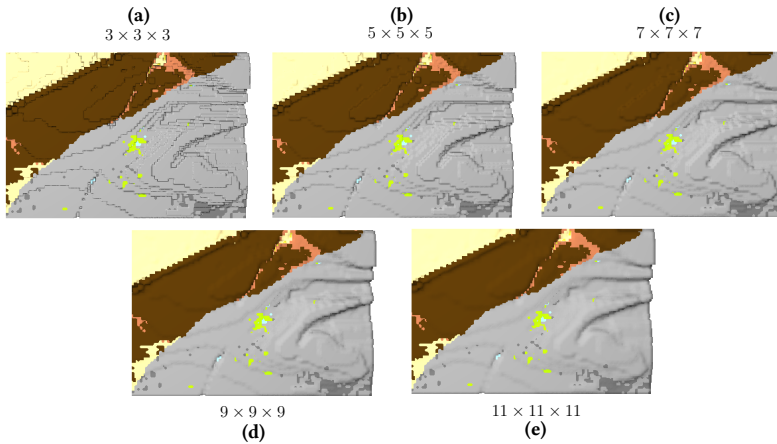


Figure 7.3: Visual results when applying different kernel sizes to a SBRT with a grid dimension of  $800 \times 1000$ . Each subfigure shows a region with a dimension of  $200 \times 300$ .

pass we store the color retrieved by the transfer function in a framebuffer object (without any illumination applied). In the next pass, the resulting world position ( $p$ ) of every ray casted through each pixel ( $r$ ) is unprojected from the depth buffer. This 3D position is the central point of an axis-oriented grid which, represents a convolution kernel. Then, we calculate the depth value ( $d'$ ) for the center of each cell of the grid. Following this, we project the center of the cells ( $p'$ ) to image space and retrieve the actual depth value from the pixels obtained ( $d$ ). With these values, we can assume that if  $d' > d$  the cell that contains  $p'$  is labeled as occupied or as empty space otherwise. Once we calculate the occupation value for all the cells, the convolution can be performed. The time consumed by this approach is less dependent on the size of the grid than in a strategy based on the object geometry. Table 7.1 and Figure 7.3 show a comparison of the computational cost and visual quality achieved using different kernel sizes. For a kernel size  $5 \leq k \leq 9$ , the drop in performance is affordable, therefore, in our framework, the kernel size is an adaptive parameter depending on the SBRT dimensions within this range.

Figure 7.4 depicts our hybrid strategy. Note that this method only can compute a reduced number of different normal vectors in comparison with gradient estimation techniques, but in practice they are enough for a good visual quality in scientific visualization.

Several authors suggest that the normal vectors can be calculated in a previous step, stored in a buffer and passed in a buffer to the raycasting



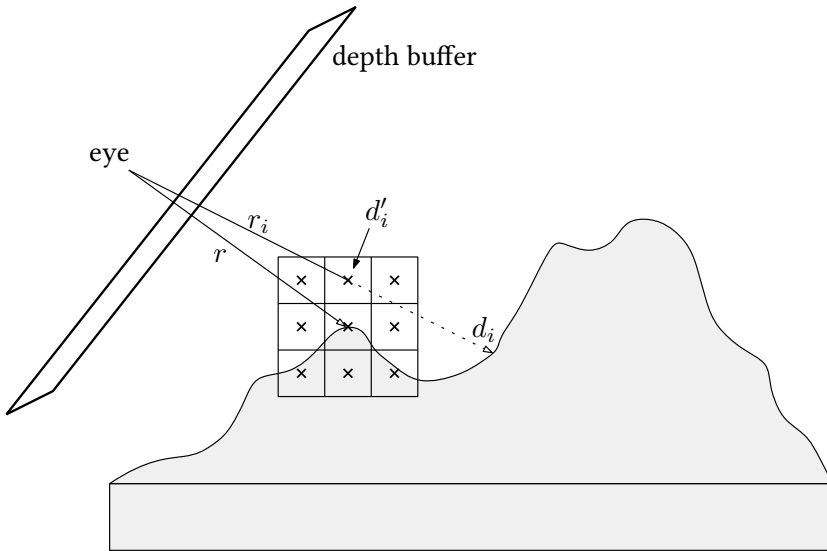


Figure 7.4: Estimation of a normal vector in image space. The ray  $r_i$  provides two depth values, an actual ( $d$ ) and an estimated ( $d'$ ) with which the binary value of the kernel (occupied or empty space) is calculated.

pipeline (Baert, Lagae, and Dutré, 2012; Sigg et al., 2006). However, this out-of-core approach requires significant extra memory in comparison with the techniques explained previously. This out-of-core approach would require an extension of the original SBRT including solely the normal vectors of the visible intervals. Assuming each component is encoded with a single precision floating-point number (32 bits), a typical small dataset with a voxel dimension of  $200 \times 250 \times 320$  would need an extra amount of memory of approximately 16 MB; A medium dataset with  $400 \times 500 \times 400$  voxels would need 64 MB, while a large dataset ( $800 \times 1000 \times 800$ )

Table 7.1: Relative drop in performance when using different kernel sizes over rendering without lighting

Kernel size	Drop in performance		
	Dataset A	Dataset B	Dataset C
$3 \times 3 \times 3$	3%	4%	3%
$5 \times 5 \times 5$	18%	21%	16%
$7 \times 7 \times 7$	43%	48%	41%
$9 \times 9 \times 9$	62%	66%	59%
$11 \times 11 \times 11$	75%	78%	72%

would need more than 256 MB. This memory footprint remains very large in comparison with the memory usage required by a SBRT. Applying a method for normal vector compression (Dado et al., 2016) would only save a 16% of its memory requirements on average. Moreover, it should be noted that this precomputation procedure must be carried out each time a different visual operation is requested which would have a significant impact on the overall system performance.

In a similar way, the color buffer resulting from the first rendering pass is subjected to a filtering process, using a Gaussian smoothing as convolution kernel.

### 7.2.2 *Stack-based representation of terrains encoding in the GPU memory*

In the following, we show different memory layouts for the storage of a stack-based representation on GPU memory. The procedure to sample a particular attribute by the raycaster is divided into two steps: the stack identification and its iteration. These two steps are common to every proposed layout; the only differences are the number of textures (or buffers) used and how they are traversed. The use of textures to send any type of information to the GPU can be considered a standard, therefore most of the approaches provided make use of them to store the stack information.

An issue that produces a major impact on the performance of any rendering method is the texture access. Fetching a texel is one of the most time-consuming operations on the GPU, but fortunately texture access exhibit spatial and temporal coherence that can be exploited to improve performance. Modern GPU architectures use caches where recent texture reads are stored following an LRU strategy (Akenine-Möller, Haines, and Hoffman, 2009). A further refinement is the use of *texture swizzling* strategies (Wang, Yang, and Cao, 2014). To improve cache coherency, texture swizzling uses space-filling curves such as Peanno-Hilbert or Morton sequence for data organization in the texture. Also, approaches based on the view direction in raycasting have been proposed (Jönsson et al., 2012). In our case, we focus on cache-friendly strategies at the stack representation level.

In a first approach (namely 1-Textures, Figure 7.5.b) we maintain a couple of 2D textures: an indices texture and an intervals texture. The aim of the former is to serve as a spatial index for the stacks descriptions stored in the intervals texture. Indices texture has the same dimensions than the support grid of the stack-based representation ( $\text{grid}_{\text{width}} \times \text{grid}_{\text{height}}$ ); hence, the

related stack is obtained by projecting the  $x, y$  components of the ray position on this grid. This projection is used as 2D coordinate to access the indices texture. For each texel of this texture, we use the components R and G from the original RGBA. In R component we encode a pointer (i.e., a 1D index) to the intervals texture while the number of intervals in this stack is encoded in G component. The intervals texture stores each stack in a row-major order. Here, we also use two components to encode an interval: the RG components store its attribute and the accumulated height respectively. The pointer ( $i_0$ ) and the number ( $n$ ) of intervals stored in the indices texture marks the beginning and the end of the stack in the intervals texture, so that the required interval is obtained by iteration between the  $i_0$  and  $i_{n-1}$  positions.

In the second approach (2-R-Supertexels, Figure 7.5.c), a single 2D texture is used. In this texture we encode two levels, one for indices and one for intervals. We construct the texture by first obtaining the maximum number of intervals among all the stacks,  $m$ . Then, in order to store each stack, we create regions of  $\lceil \sqrt{m} \rceil \times \lceil \sqrt{m} \rceil$  dimensions, where  $\lceil x \rceil$  is the ceiling operator. These regions serve as "supertexels" to provide a pseudoindices scheme. Therefore, the actual dimensions of the texture are calculated by:

$$\begin{aligned} \text{texture}_{\text{width}} &= \text{grid}_{\text{width}} * \lceil \sqrt{m} \rceil \\ \text{texture}_{\text{height}} &= \text{grid}_{\text{height}} * \lceil \sqrt{m} \rceil \end{aligned} \quad (7.2)$$

This strategy allows a straightforward way to get the related stack since all the regions are squares. Similarly, to the previous approach, the ray position must be projected in the support grid, but, in contrast, an offset has to be added to the cell obtained. Once the region has been identified, we only have to iterate through it to locate the interval. Also, this texture encodes the attribute and the height of an interval in their RG components. Alternatively, the stacks can be saved in non-squared regions by finding the rectangle with the minimum perimeter. This can be formulated as a simple optimization problem:

$$\begin{aligned} \underset{\text{width, height}}{\text{argmin}} \quad & \text{width} + \text{height} \\ \text{subject to} \quad & \text{width} * \text{height} \geq m \\ & \text{width} > 0 \\ & \text{height} > 0 \end{aligned} \quad (7.3)$$

This is the preferred layout in this approach when  $m$  is not prime. Otherwise, the former layout (square regions) should be chosen.

Another possible approach is that proposed by Natali, Klausen, and Patel (2014). They used a 3D texture of width  $\times$  height  $\times$   $m$  dimensions in which the intervals are stored along the  $z$  axis (approach 4-3D-Texture, Figure 7.5.e). In contrast, we use a 2D texture of (width  $\times$   $m$ )  $\times$  height dimensions (approach 3-L-Supertexels, Figure 7.5.d). The stack identification is accomplished in a similar way as explained above, but with the difference that we only have to add an offset to the  $x$  index. Then, the stack iteration will be done in a linear manner.

Despite in DVR the prevailing memory schemes use textures to store volumetric data; current graphics visualization APIs provide other ways to send data to the GPU such as Uniform Buffer Objects (UBO). Recently OpenGL in its 4.3 version has included a new method for storing large amount of data in the GPU, the Shader Storage Buffer Objects (SSBO). Actually, a SSBO is an enhanced and more flexible version of UBO: OpenGL implementations must support UBOs of at least 16KB whilst the guaranteed minimum for a SSBO is 128 MB. Furthermore, the specification of SSBO allows the definition of a non-fixed size array. A SBRT can be represented in a straightforward way as an array of intervals, being this scheme well suited for its storage in a SSBO due to its internal memory layout: In conjunction with SSBOs, OpenGL introduced the `std430` layout which packs scalar arrays more efficiently than the old `std140` (Sellers, Wright Jr, and Haemel, 2016). An approach based on SSBOs is also proposed (approach 5-SSBO). Similarly to approach 1-Textures, we populate two SSBOs, one for indices and another for intervals (Figure 7.5.f).

In order to clarify these approaches, an illustrative example is depicted in Figure 7.5. Given a SBRT with a grid of width  $\times$  height cells and a maximum stack size of 5 intervals (Figure 7.5.a), the memory storage patterns are described next: in approach 1-Textures (Figure 7.5.b), the indices texture has the same dimension that the grid of the stack-based representation (width  $\times$  height), being the interval data sequentially added through the interval texture. To set the size of the interval texture, we first need to know the total number of intervals of the stack-based terrain. Then, with Equation 7.3 we can determine the texture dimension assigning to  $m$  the total number of intervals. If this value is prime, the Equation 7.2 can be used. Also, this value can be increased in order to use Equation 7.3. Approach 1-Textures can waste some memory since the last cells of last row usually are not used in this latter case. For approach 2-R-Supertexels

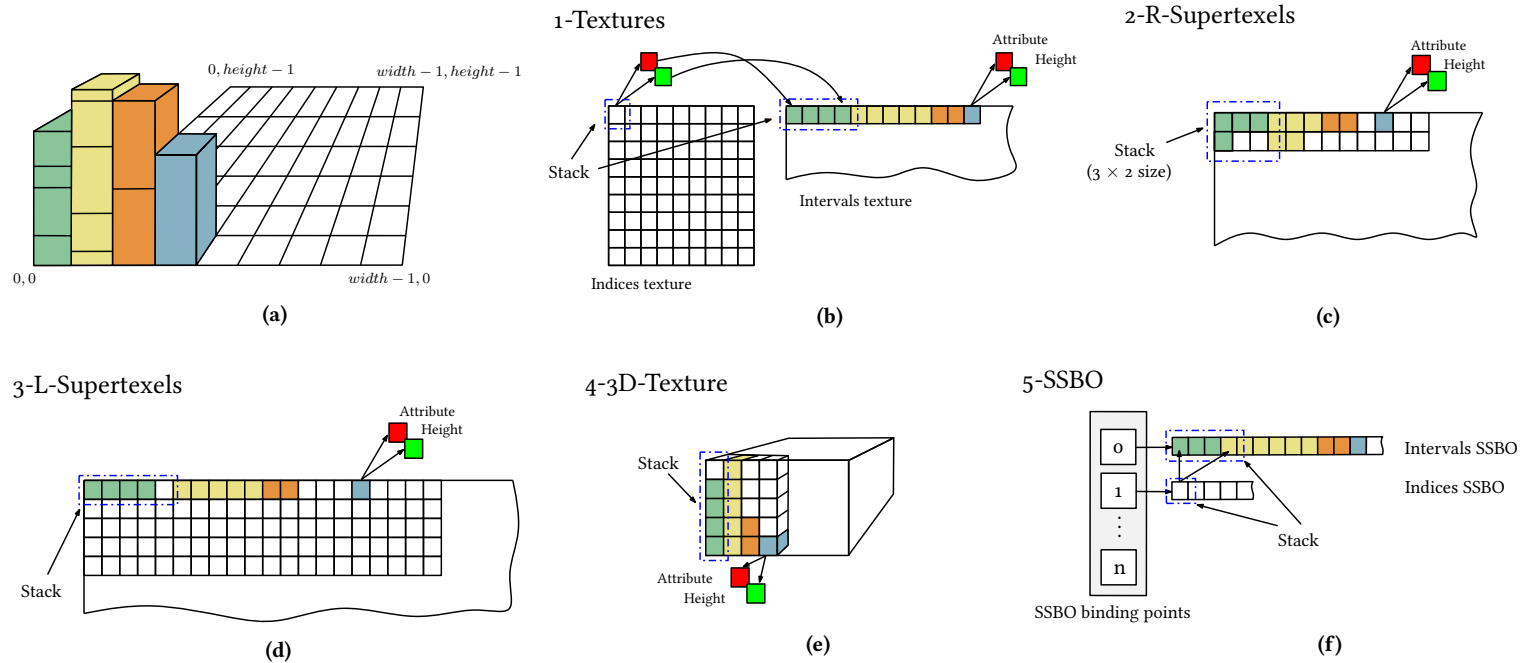


Figure 7.5: Memory storage patterns (b, c, d, e, f) for a stack-based geomodel (a).

(Figure 7.5.c) it is necessary to set the size of the supertexels. The largest stack determines this value (5 intervals, in the example). The texture contains supertexels of  $3 \times 2$  texels. The approaches 3-L-Supertexels (Figure 7.5.d) and 4-3D-Texture (Figure 7.5.e) are similar. Finally, for the approach 5-SSBO (Figure 7.5.f) two arrays containing the indices and the intervals are stored in two separate SSBOs.

### 7.2.3 *Visual operations*

In this subsection, we describe a set of operations common in geoscientific applications and their implementation in our system. These operations, such as the selective visualization of boreholes or the hiding of strata of materials, provides geoscientists with visual tools to take decisions at a glance.

#### 7.2.3.1 *Strata visualization*

With the aim of showing hidden elements of the subsurface, each stratum can be attenuated or directly excluded from visualization. This is achieved by simply modifying the opacity of the material color in the color lookup table. By using raycasting for rendering, it is not necessary to construct a new geometry when hiding certain layer. Figure 7.6 shows some examples of layers attenuation.

#### 7.2.3.2 *Cross section visualization*

Another way to visualize internal structures is via geological cross sections. These are 2-dimensional slices of the subsurface, usually vertical, used to study the distribution of rock types, including their ages, relationships and structural features. Our system not only allows to perform sections by vertical cutting planes, but also by any freely-oriented plane in 3D space. In this case, the rendering algorithm will start at the intersection of the casted ray with the cutting plane. Figure 7.7 illustrates this feature.

#### 7.2.3.3 *Borehole log visualization*

Borehole records are obtained by drilling the rock core and they contain relevant information on lithology, stratum thickness or physical properties of the geological formations. The samples brought to the surface or the measurements made by the instruments lowered into the hole are mostly

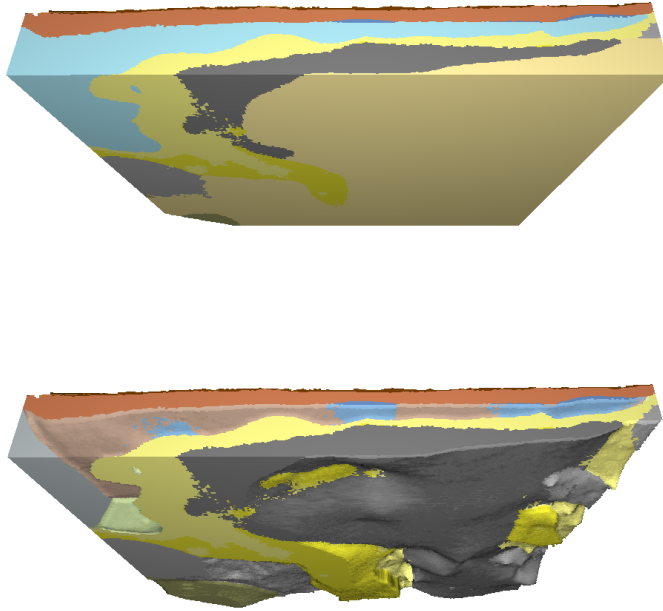


Figure 7.6: Example of layer attenuation (bottom) of an original dataset (top).

the input data for the construction of subsurface models. We can simply label a stack as corresponding to a borehole by using a Boolean flag in the indices texture. When borehole log visualization is selected, only the stacks with this flag set are shown. Figure 7.8 shows an example of borehole log visualization.

#### 7.2.3.4 *Applying textures to terrain*

We also provide an operation for adding an image texture to the surface of the terrain such as orthophotos or topographic maps. In a direct way, the pixel color can be obtained from the image and optionally combined with the material color, provided that the ray intersects with the terrain surface. In Figure 7.9 an example of this feature is shown.

#### 7.2.4 *GIS-based layer display*

The mechanism that GIS applications use to display heterogeneous geographical data is by means of layer overlay. The layers are visualized in

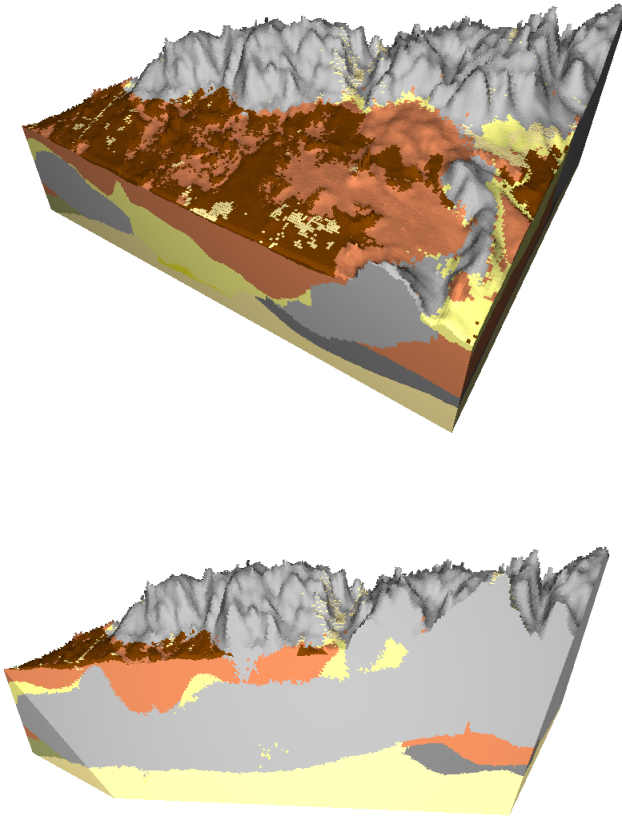


Figure 7.7: Cross-section (bottom) and original model (top).

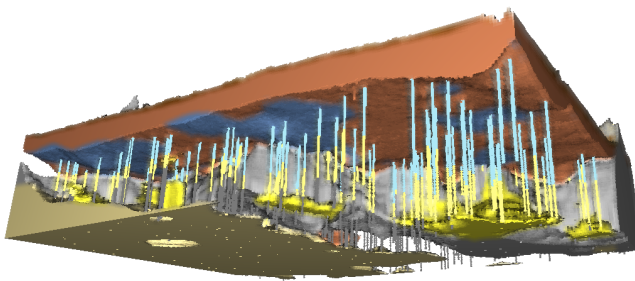


Figure 7.8: Example of borehole visualization. They are shown as cylinder of two materials (blue and yellow color).



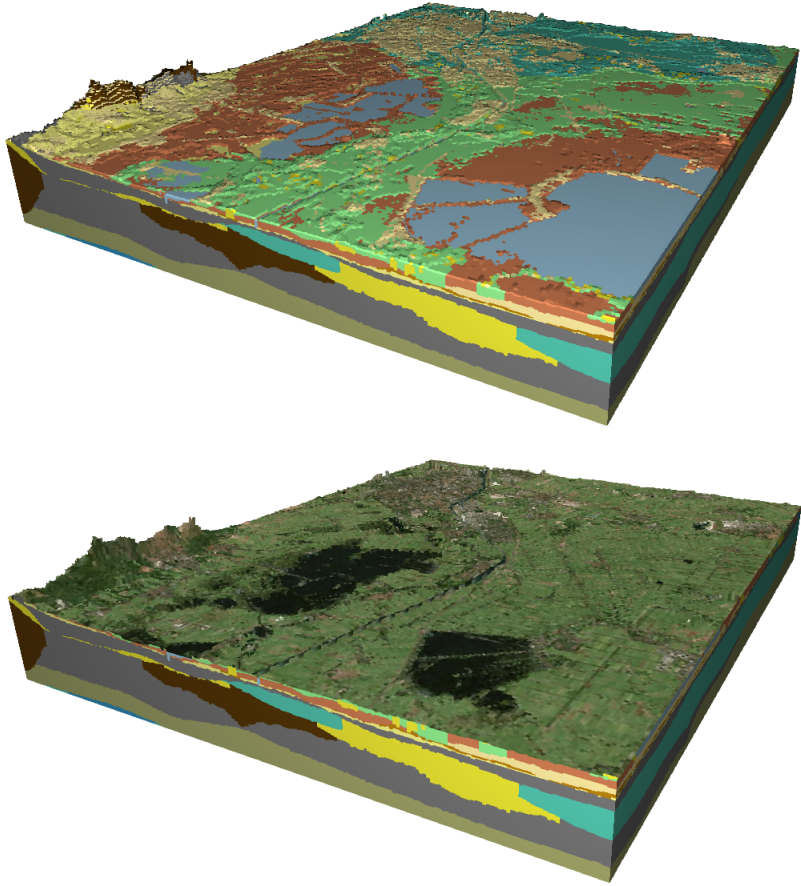


Figure 7.9: Example of orthophoto application. Original model (a) and model with an orthophoto applied on its surface (b).

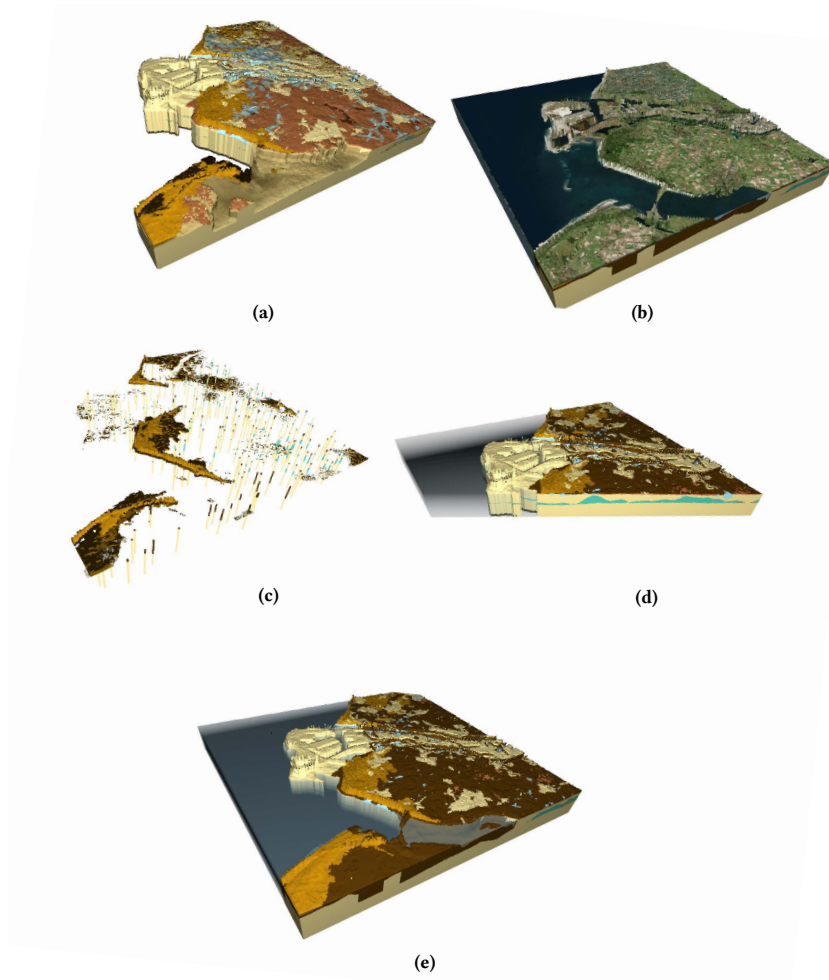


Figure 7.10: An original model (e) and different examples of visual operations applied: hiding of many layers (a), application of an orthophoto on the surface (b), borehole visualization combined with layer hiding (c) and visualization of cross-section (d).

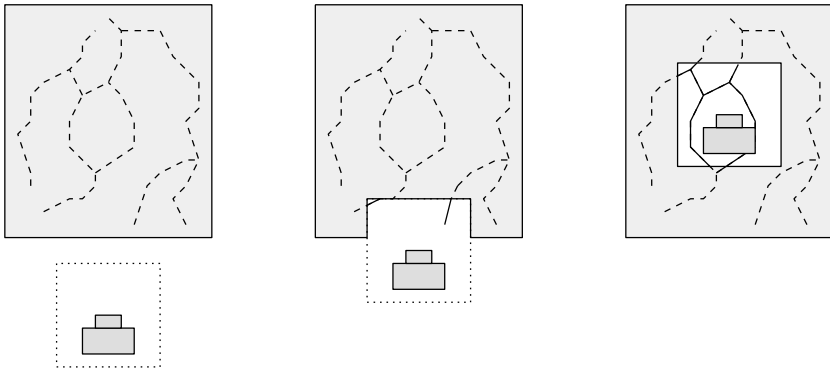


Figure 7.11: Internal description of the heterogeneous data layers using a camera enclosing bounding volume.

a specific order, so they can be hidden by others placed above of them. In a two-dimensional context, these issues can be trivially solved by raising the particular layer at the top of the stack layer. However, this solution cannot be applied to 3D GIS or GSIS applications, since the order of visualization is given by a depth buffer. For example, a layer representing subsoil structures such as sewage or subway networks may be hidden if a field layer that represent the terrain surface and subsurface is loaded into the system. To solve this situation, we can make use of some of the visual operations defined in the previous section such as layer attenuation or cross-sections.

In addition, we define a simple solution based on the camera position. For this purpose, the camera is enclosed in a bounding volume, namely an Axis-Aligned Bounding Box (AABB), locating the camera at its center. Then, the portion of the layer that collides with the AABB is excluded from the visualization. This feature allows moving the camera into the volumetric dataset in a very natural way, showing hidden layers of materials and internal structures of the vector layers. Figure 7.11 illustrates this feature.

Figures 7.12, 7.13 and 7.14 show some visualization examples.

## 7.2.5 Performance analysis

### 7.2.5.1 Hardware setup

The hardware configuration for the experiments carried out in this thesis consisted of a PC equipped with an Intel Core i7-4790 CPU running at 3.60 GHz with 16 GB of RAM and an NVIDIA GeForce GTX 970 graphics card.

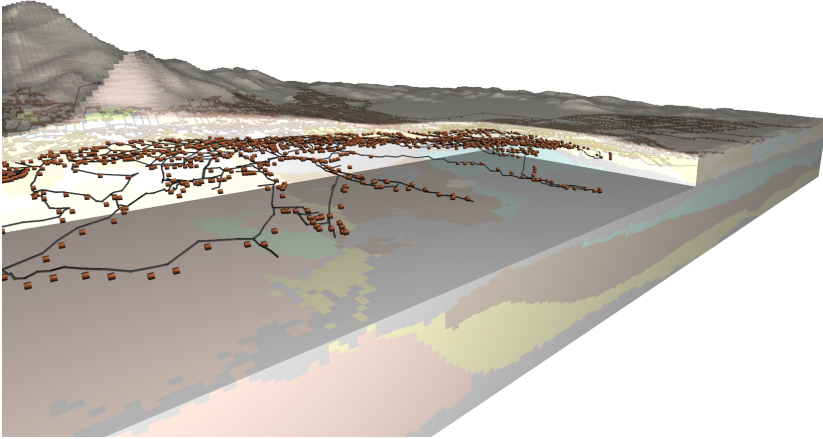


Figure 7.12: Example of attenuation of a field layer. A subsurface vector layer can be depicted.

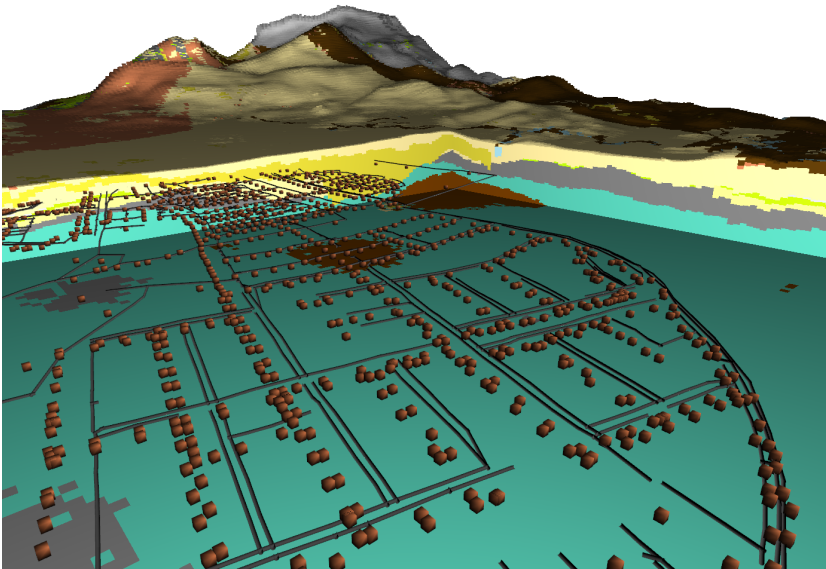


Figure 7.13: A close-up viewing of three different layers. Two subsurface vector layers are combined with a SBRT layer. The vector layers represent a sewage network (segment layer) and a set of manholes (point layer).

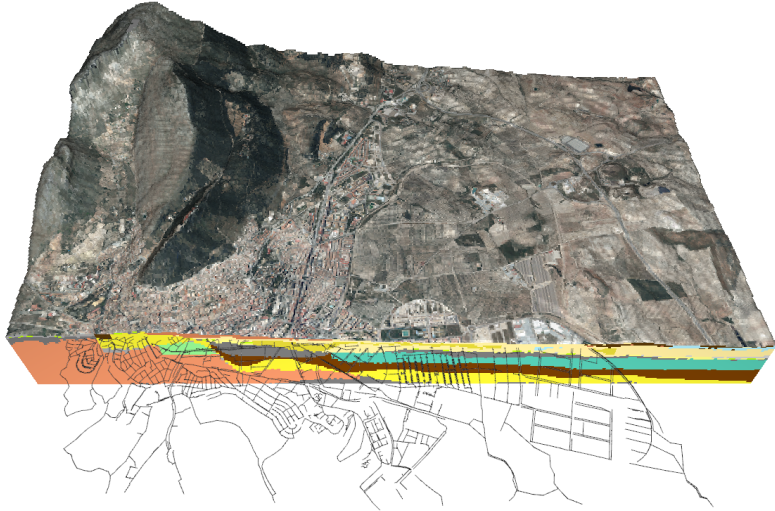


Figure 7.14: Cross section of a field layer. The cut allows the visualization of a subsurface vector layer. In addition, an orthophoto is applied on the field layer surface.

The framework proposed has been implemented in C++ and OpenGL 4.5 as 3D graphic library.

#### 7.2.5.2 Comparison of memory layouts

To test the approaches described in Section 7.2.2), we performed a set of experiments. We measure and compare the millions of rays per second (MRPS) by rendering the different layouts and their memory usage. We therefore tested four datasets with different grid dimension and stack sizes;  $200 \times 250$  with a maximum number of 10 intervals per stack, which corresponds to a model of  $200 \times 250 \times 320$  voxels (Dataset A);  $400 \times 500$  with 15 intervals, which corresponds to a model of  $400 \times 500 \times 400$  voxels (Dataset B) and  $800 \times 1000$  with 16 intervals, which corresponds to a model of  $800 \times 1000 \times 800$  (Dataset C). Finally, Dataset D is a modification of Dataset C by deleting several layers and visualizing some stacks (see Figure 7.15). These datasets were obtained from the DINOloket database (Gunnink et al., 2013). To measure the impact of raycasting direction we rendered the dataset from different virtual camera positions.

Regarding the implementation of the textures and buffers involved in the experiments in Figure 7.5, we declared a 2D texture of indices with

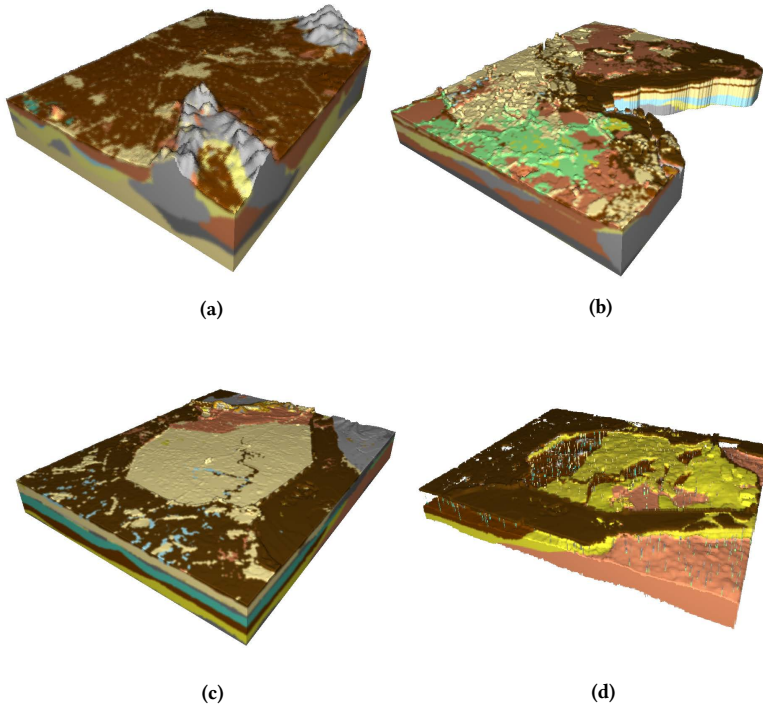


Figure 7.15: Overview of the datasets used in the experiments. Dataset A is shown in the left-upper corner (a), Dataset B is in the right-upper corner (b), Dataset C is in the left-lower corner (c) and Dataset D is in the right-lower corner (d).

GL\_RG32UI internal format and a 2D texture of intervals with GL\_RG16F internal format for approach 1-Textures. The approaches 2-R-Supertexels and 3-L-Supertexels used a single 2D texture with GL\_RG16F as the internal format. Also, we compared the layouts proposed with the 3D texture strategy presented in (Natali, Klausen, and Patel, 2014) (approach 4-3D-Texture). Likewise, the internal format of the 3D texture was GL\_RG16F. For approach 5-SSBO, the indices SSBO is filled by an array of structures formed by two 32-bits unsigned integers, whilst the Intervals SSBO by two 16-bits floating-point numbers. The rendering has been performed on a  $1056 \times 884$  viewport.

Figure 7.16 reports the minimum value of MRPS reached in the experiments for each dataset and approach (worst case). This case usually occurs when the ray has to traverse more empty space. On the other hand, Figure 7.17 shows the maximum value of MRPS reached (best case).

The plot showing the worst case follows a similar pattern for each dataset. Approaches 1-Textures and 5-SSBO perform better than others. Due to the fact that these approaches are linear, the prefetching and the cache work better at stack level. Furthermore, the layout provided by OpenGL for the SSBO seems better than approach 1-Textures. In Figure 7.17, results are less conclusive, obtaining in all datasets more than 200 MRPS for the chosen viewport dimensions.

The layout of approach 3-L-Supertexels presents some restrictions. The approach cannot be used for very large models. The graphics cards support textures up to a maximum size. For the card used in our experiments, this maximum size is  $16,384 \times 16,384$  for 2D textures. Therefore, in order to encode a dataset with a dimension of  $1,024 \times 1,024$  and more than 16 maximum intervals per stack, this layout needs at least a  $17,408 \times 1,024$  2D texture, exceeding the GPU limits.

The memory requirements of each layout as well as the wasted memory have also been summarized in Table 7.2, being approaches 1-Textures and 5-SSBO the most efficient packing the data. In these approaches, the intervals structure (in texture or SSBO) does not need a fixed interval size in order to identify a stack due to the indices. However, as can be noted, approach 1-Textures requires some extra memory for Dataset A that can be neglected. This is because the total number of intervals, 309,433 in this case, is a prime value. In order to decompose the set of intervals in a rectangular texture, we summed one to this value to be able to use Equation 7.3 (obtaining a  $479 \times 646$  texture). The outcomes for the remaining approaches show a noticeable wasting of memory storage.

In general, the approach 5-SSBO can be considered the best one in terms of performance since it reached the higher MRPS values on average. Moreover, the approaches 5-SSBO and 1-Textures provide the best results in memory usage and in the ratio consumed/wasted. SSBOs can use all the available GPU memory, which is considerably higher than the maximum size allowed for textures (as stated above,  $16,384 \times 16,384$  dimensions). Therefore, the approach 5-SSBO is able to manage datasets larger than those who can be held in 1-Textures, without the use of an out-of-core strategy. However, SSBOs are only available in modern GPUs supporting OpenGL 4.3 or higher. Consequently, approach 1-Texture provides a more general solution for older hardware.

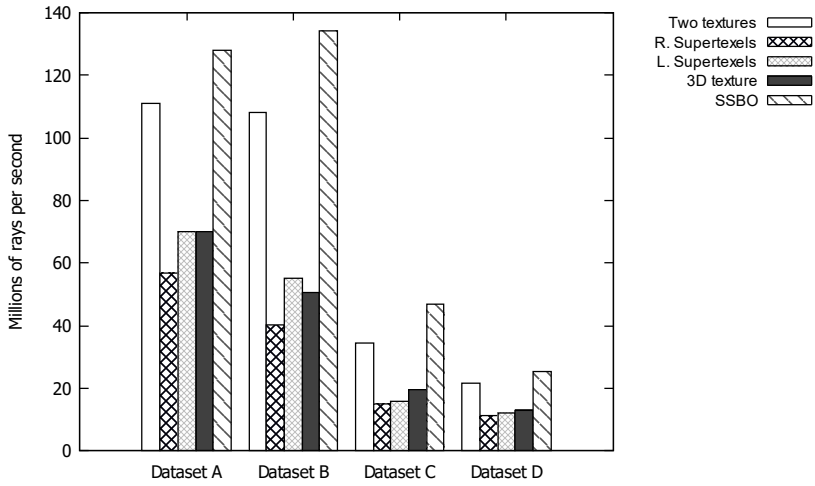


Figure 7.16: Minimum MRPS reached.

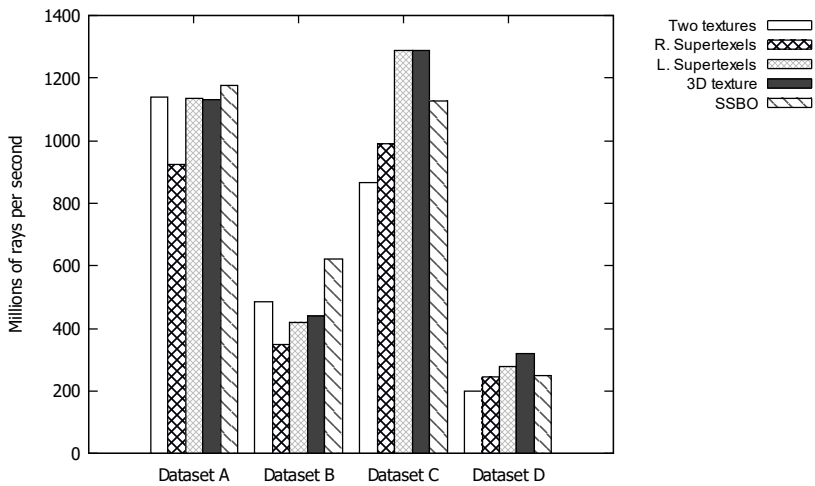


Figure 7.17: Maximum MRPS reached.



Table 7.2: Storage requirements of memory layouts.

	Dataset A			Dataset B			Dataset C		
	Wasted memory (MB)	Consumed memory (MB)	Percentage of wasting (%)	Wasted memory (MB)	Consumed memory (MB)	Percentage of wasting (%)	Wasted memory (MB)	Consumed memory (MB)	Percentage of wasting (%)
1-Textures	$3.81 \times 10^{-6}$	1.56	$2.4 \times 10^{-4}$	0.00	6.20	0.00	0.00	29.06	0.00
2-R-Supertexels	0.73	1.91	38.10	6.77	11.44	59.20	25.87	48.83	53.00
3-L-Supertexels	0.73	1.91	38.10	6.77	11.44	59.20	25.87	48.83	53.00
4-3D-Texture	0.73	1.91	38.10	6.77	11.44	59.20	25.87	48.83	53.00
5-SSBO	0.00	1.56	0.00	0.00	6.20	0.00	0.00	29.06	0.00

Table 7.3: Comparison of results for Dataset A. FPS means frames per seconds and MRPS, millions of rays per second.

Approach	FPS (worst)	FPS (best)	MRPS (worst)	MRPS (best)	Preprocessing time (s)	GPU memory usage (MB)	Main memory usage (MB)
SBRT	137	1259	128	1175	0	1.56	0.00
Gigavoxels	158	560	174	589	115	459.33	286.86

Table 7.4: Comparison of results for Dataset B. FPS means frames per seconds and MRPS, millions of rays per second.

Approach	FPS (worst)	FPS (best)	MRPS (worst)	MRPS (best)	Preprocessing time (s)	GPU memory usage (MB)	Main memory usage (MB)
SBRT	144	664	134	620	0	6.19	0.00
Gigavoxels	194	654	181	611	115	459.33	286.86

Table 7.5: Comparison of results for Dataset C. FPS means frames per seconds and MRPS, millions of rays per second.

Approach	FPS (worst)	FPS (best)	MRPS (worst)	MRPS (best)	Preprocessing time (s)	GPU memory usage (MB)	Main memory usage (MB)
SBRT	50	1208	47	1128	0	29.06	0.00
Gigavoxels	241	690	225	644	920	459.33	286.86

Table 7.6: Comparison of results for Dataset D. FPS means frames per seconds and MRPS, millions of rays per second.

Approach	FPS (worst)	FPS (best)	MRPS (worst)	MRPS (best)	Preprocessing time (s)	GPU memory usage (MB)	Main memory usage (MB)
SBRT	27	265	25	247	0	29.06	0.00
Gigavoxels	24	46	22	43	920	459.33	286.86

## 7.3 RAYCASTING THE QUADSTACK

Models represented by QuadStack can be rendered by ray casting without using an intermediate representation (e.g., a SBR or a 3D grid of voxels). Similar to other hierarchical data structures QuadStack allows an efficient implementation of ray casting, which is solved at the gstack level first, then at the interval level, and finally at the heightfield level, as depicted in Figure 7.18.

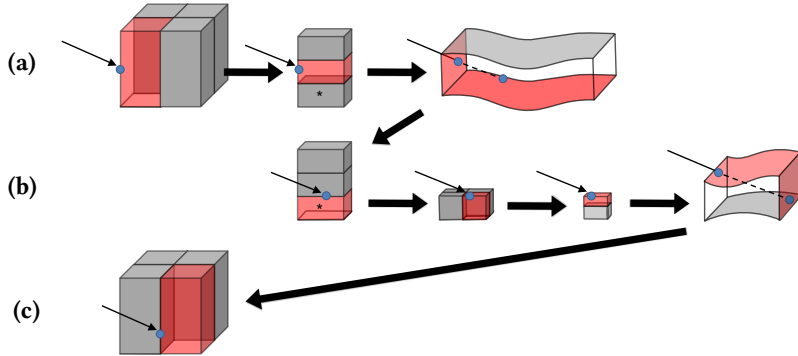


Figure 7.18: Raycasting a QuadStack, first resolved at the QuadStack level, then at the interval level and finally, at the heightfield level (a). When an \*-interval is found, a recursive call to traverse the gstacks in children nodes is required (b). After processing a gstack, the traverse continues with the next one, until the ray exits the volumetric model.

The rendering procedure starts by computing the intersection between the ray casted into the scene and the gstack at the root node. This can be computed efficiently, since a gstack defines a cuboid that spans the entire  $z$  dimension of the volumetric space. Then, the first intersection with an interval  $i_k$  of the gstack is calculated. This involves computing the intersection of the ray with its four lateral faces and two bounding heightfields:  $H_k$  (top) and  $H_{k-1}$  (bottom). If the interval  $i_k$  is terminal, its contribution to the accumulated color and opacity is computed as the integral of the transfer function for the attribute  $\alpha_k$  between the entry and exit points of the ray, together with an opacity correction due to adaptive sampling (Hadwiger et al., 2006). The ray processing finishes immediately if the opacity of the color is close to one. If  $i_k$  is a \*-interval, a recursive call is made to compute the contribution by the ray traversal of the gstacks in the four descendant nodes. After  $i_k$  has been processed, the traversal continues with a new interval until the ray exits the gstack. For the gstack

at the root node this finishes the sampling of the QuadStack, but in the general case, it implies the return of a recursive call and further processing in the gstack at the parent node.

The most time-consuming step in the QuadStack raycasting is the ray-heightfield intersection computation. In order to accelerate this step, each heightfield is mipmapped storing the min-max instead of averaging values (Tevs, Ihrke, and Seidel, 2008). Each mipmap level defines a bounding geometry for the heightfield with the shape of a set of cuboids with the same dimensions in the XY plane. The highest mipmap level represents the coarsest approximation (i.e., a bounding box) and the level zero represents the finest (i.e., the heightfield itself). To test if a ray intersects the heightfield associated to a given interval of a gstack, first the intersection with the bounding cuboid defined by the highest mipmap level is computed. If the cuboid is hit, the intersection computation continues with the four cuboids contained in it in the preceding mipmap level, until the ray passes by or hits the heightfield at level zero. The extra memory required (66% for each heightfield) can be reduced by using the heightfield compression explained in Chapter 5, resulting in a good trade-off between rendering time and memory footprint.

### 7.3.1 *QuadStack encoding in the GPU memory*

The key for an efficient raycasting implementation is a careful encoding of the model representation in the GPU memory. Our memory layout for QuadStack consists of three buffers: a tree buffer that encodes the QuadStack structure, a lookup table (LUT) for the set of gstacks, and a heightfield buffer that packs the heightfields associated to each gstack (Figure 7.19).

The structure of the tree buffer is inspired by the proposal of Lefebvre, Hornus, and Neyret (2005), where each tree node keeps either the data itself (leaf), or an index to its descendants (otherwise). Contrary to the cited work, in our structure an inner node also contains the corresponding gstack. Therefore, a node in the tree buffer holds indices of its children, and an extra pointer to the gstack LUT indicating the beginning of the sequence of intervals and its size.

The gstack LUT comprises every gstack in a consecutive manner. Each element of this buffer defines a gstack interval formed by its attribute and a pointer to the beginning of its corresponding heightfield in the heightfields buffer.

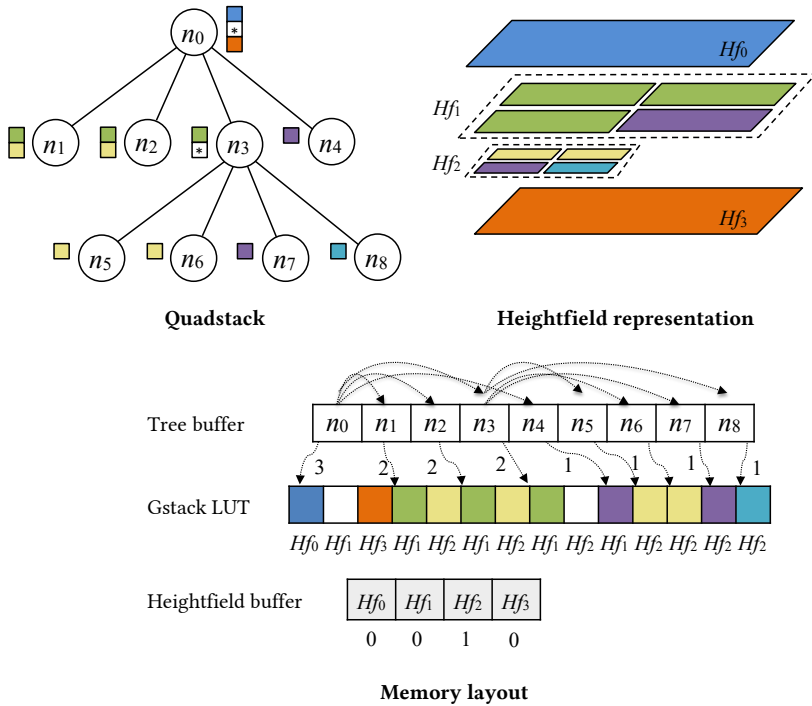


Figure 7.19: Memory layout of a given QuadStack after performing its heightfield arrangement.

Heightfields are compressed by using the approach described in Chapter 5. The detailed structure of the heightfield buffer is shown in Figure 7.20. A header contains a field with the number of blocks into which the heightfield is divided, followed by a sequence of block descriptors that comprises the base value, the number of bits required for encoding the height differences, and the address of the height data. Next, the encoded height data for each block is stored consecutively. A Morton encoding layout provides the required spatial coherence when accessing data both for block metadata and height differences. As shown in Figure 7.19, an index indicating the level of the QuadStack to which the heightfield is associated (base level) has been added. When a gstack in a descendant node references a specific quadrant of this heightfield, the use of this index avoids adding extra information at the LUT buffer: the actual quadrant can be quickly determined from the base level of the heightfield and the level queried.

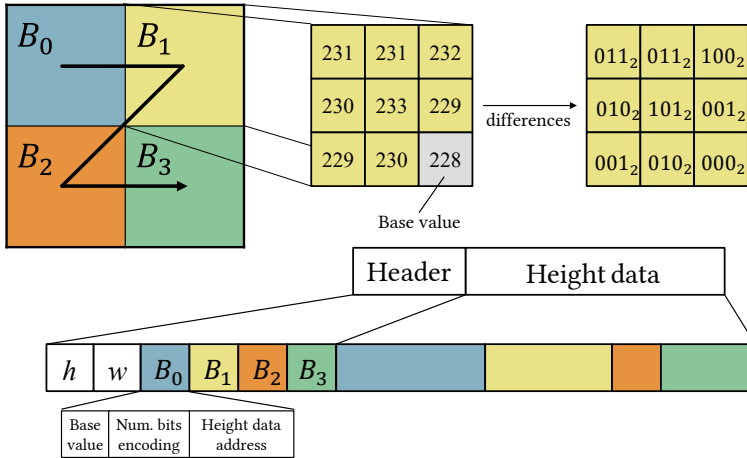


Figure 7.20: Heightfield compression scheme.

### 7.3.2 Performance analysis

We used five datasets for evaluation that exhibit strong to medium layered structure: two terrain models with several layers of different geological content (Terrain 1, Terrain 2) from DINOloket database (Gunnink et al., 2013), microstructure of Li-Pol battery (Battery) (Ebner et al., 2013), a part of an industrial model of a wing of a plane (Wing) (Aage et al., 2017), and a magnetic reconnection simulation (Magnetic) (Guo et al., 2014). We wanted to cover a wide spectrum of applications and wide variety of layers and structures. Figure 7.21 shows these datasets.

#### 7.3.2.1 Volume data compression

We have evaluated the ability of QuadStack to perform lossless compression of the input volumetric data. The amount of memory here examined is the total memory needed for the rendering.

The measured results are shown in Table 7.8. The memory for the input volume data ranges from 105 MB to 326 MB. The voxel representation uses 16 bits per voxel (bpv).

QuadStack requires from 3 MB to 16 MB of memory for our datasets, demanding less than 2 bpv for all the scenarios. The achieved compression ratio is between  $9\times$  and  $108\times$ .

A breakdown of the memory budget for QuadStack is show in Table 7.7. We can see that most of the memory is used by the quadtree and attribute

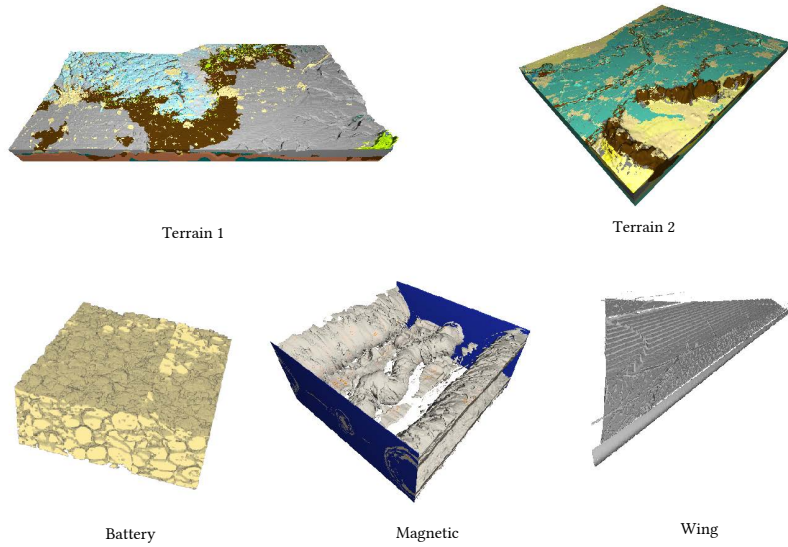


Figure 7.21: Datasets used in the experiments.

information. The great compression ratio provided by decoupling heightfields and attributes compensate the min-max mipmaps memory overhead. In Table 7.8 can be seen that even though including this memory increment, QuadStack requires least memory in almost every case. It should be noted that the memory required to store the min-max mipmaps is optional since it serves only as rendering acceleration data structure.

<b>Dataset</b>	<b>Attributes [MB (%)]</b>	<b>Heightfields [MB (%)]</b>	<b>Mipmaps [MB (%)]</b>	<b>Total [MB (%)]</b>
Terrain 1	12.2 (52%)	3.2 (14%)	8.0 (34%)	23.5 (100%)
Terrain 2	3.3 (54%)	1.0 (16%)	1.8 (30%)	6.2 (100%)
Battery	7.3 (39%)	4.8 (26%)	6.5 (35%)	18.6 (100%)
Wing	1.5 (37%)	1.2 (30%)	1.3 (33%)	4.0 (100%)
Magnetic	3.9 (49%)	1.5 (19%)	2.5 (32%)	7.9 (100%)

Table 7.7: Breakdown of the QuadStack memory requirements. The columns represent memory needed for representing the quadtree and attributes, compressed heightfields, and min-max mipmaps.

### 7.3.2.2 *Direct volume visualization*

We evaluated the GPU-based direct visualization of QuadStack by using the five datasets proposed above. As reference we used two visualization backends implemented in the Paraview software: the GPU accelerated rendering using VTK, and CPU based rendering using OSPRay. For all scenes we used a set of representative views for which we measured the rendering times that were converted to performance numbers expressed in millions of rays per second (MRays/s). These measures were averaged to report an overall performance for each scene and method at the bottom of the Table 7.8.

VTK uses a highly optimized GPU ray caster of uncompressed volume data and we can observe that it achieves the highest rendering performance for most cases. The QuadStack rendering is between  $1.38$ - $17.63\times$  slower, with the exception of the Wing scene where clearly outperforms VTK ( $2.25\times$  faster). The OSPRay renderer is CPU based and it generally achieves the lowest performance on the tested scenes. It is  $1.3$ - $4\times$  slower than the direct QuadStack visualization. The direct SBR rendering uses an equidistant sampling to satisfy the Nyquist-Shannon sampling theorem. The direct SBR rendering is slower than the QuadStack (between  $1.13$ - $7.9\times$ ) for all datasets except for the Battery model ( $4.33\times$  faster).

QuadStack performs better than the rest of renderers/techniques when a ray has to skip a large amount of empty space encoded in high levels of the tree structure (Terrain 1-2 and the Wing). However, the VTK renderer provided a better overall performance. This is primarily due to the heightfield decompression overhead implicit in each sampling step of the QuadStack traversal. As QuadStack uses just a fraction of memory required by the VTK renderer, it provides a good tradeoff between memory usage and rendering time.



	Terrain 1	Terrain 2	Battery	Wing	Magnetic
Resolution	$1,024 \times 512 \times 250$	$512 \times 512 \times 400$	$512 \times 512 \times 210$	$1,024 \times 1,024 \times 163$	$512 \times 512 \times 512$
Max. Layers	14	12	34	19	32
<b>Construction time [sec]</b>					
Voxel grid	-	-	-	-	-
SBR	8.7	5.6	3.3	14.0	9.5
QuadStack	12.7	6.0	10.5	14.5	10.7
<b>Memory size [MB / bpv (ratio)]</b>					
Voxel grid	250 / 16 (100%)	200 / 16 (100%)	105 / 16 (100%)	326 / 16 (100%)	256 / 16 (100%)
SBR	<b>22 / 1.41 (8.8%)</b>	9 / 0.73 (4.5%)	27 / 4.08 (25.6%)	22 / 1.07 (6.7%)	11 / 0.7 (4.3%)
QuadStack	24 / 16 (100%)	<b>6 / 0.48 (3.0%)</b>	<b>19 / 2.90 (18.1%)</b>	<b>4 / 0.20 (1.2%)</b>	<b>8 / 0.50 (3.1%)</b>
<b>Rendering performance [MRays/sec (ratio)]</b>					
VTK (Voxel grid)	<b>126 (100%)</b>	<b>72 (100%)</b>	<b>423 (100%)</b>	8 (100%)	<b>180 (100%)</b>
OSPRay (Voxel grid)	18 (14.3%)	23 (31.2%)	18 (4.3%)	6 (75%)	28 (15.5%)
SBR	37 (29.3%)	46 (63.9%)	104 (24.6%)	13 (162%)	10 (5.5%)
QuadStack	72 (57.1%)	52 (72.2%)	24 (5.7%)	<b>18 (225%)</b>	79 (43.8%)

Table 7.8: Results measured on five test datasets. The table shows basic dataset properties, construction times and memory requirements for evaluated representations (Voxel grid, SBR, QuadStack). The bottom part of the table compares rendering performance for different methods (VTK, OSPRay, QuadStack). The results for the method with the lowest memory consumption and best rendering performance are highlighted in bold.

## 7.4 CONCLUSION

This chapter has brought together the research in terms of visualization carried out in this thesis. In Section 7.2 we have suggested by first time the use of the stack-based representation as the main rendering data structure. Also, we have tested several approaches based on typical textures and the new SSBOs, to store this representation on GPU reaching an outcome quite acceptable for interactive applications. It should also be pointed out that, as far as we know, the use of SSBOs to store volumetric data has not been explored in the literature yet. Additionally, this section presents an accurate, simple and efficient method to calculate normal surface vectors in image space. Finally, we have shown how several common visual operations of interest in geoscientific applications can be implemented in a straightforward way using the SBRT and the described visualization method.

Finally, Section 7.3 introduces a novel algorithm for direct rendering of the compressed data by a QuadStack. We show its GPU implementation that performs comparable with the state-of-the art algorithms for direct volume rendering, but instead of using full data it works directly with the compressed volumes.



Part IV

CONCLUDING REMARKS



## CONCLUSIONS

---

In this dissertation, we have proposed the use of the stack-based representation as both SBRT and QuadStack as a basis for geomodeling applications. Stack-based representation has proved to be a very compact representation with a good trade-off between memory requirement and access speed. We have put in context this representation with those existing in the literature, highlighting its advantages and drawbacks. Throughout the chapters of this document, its different aspects have been developed such as a formal definition of the representation and its operations, construction procedures, access methods and visualization techniques.

On a personal and non-technical level, these last four years have allowed me to improve myself as researcher and person. The work carried out has generated important results which have been published or are currently under revision in top journals and conferences on different fields, proving the multidisciplinary research made in this thesis.

### 8.1 SUMMARY OF CONTRIBUTIONS

In the first part of this dissertation, we highlighted that the stack-based representation has been used only for a few papers focused on visualization and just as an auxiliary data structure, despite its interesting features. Therefore, we suggested the stack-based representation as a first-class representation for managing geological data at surface and subsurface levels. Here, we have highlighted its low memory requirements in comparison with other commonly used structures such as voxel grids or octrees.

Even though we provide a clear and intuitive definition of the SBRT (see Section 3), in Section 4 we present a formal framework to validate this representation and define properties and algorithms precisely. We based this formalization on the geo-atom theory, a framework that unifies field and discrete object data models in a general spatial representation. We began by developing a framework in which the geo-atom theory is used for the representation of 3D voxel-based terrains that supports our description of the final formalization of the SBRT representation. In addition, we have defined a set of common spatial operations on this representation using the

tools provided by map algebra. More complex geoprocessing operations or geophysical simulations using the SBRT as underlying representation can be implemented as a composition of these fundamental operations. Finally, a data model and an implementation extending the coverage concept provided by the Geography Markup Language standard are suggested.

Once this core representation is defined, we aimed to improve it to reduce its space requirements (Chapter 5). QuadStack is a novel data structure which not only exploits the layered structure of datasets such as the SBR, but also the horizontal spatial coherence within the layers. QuadStack first compresses the volumetric data into vertical stacks that are further compressed into a quadtree that identifies and represents the layered structures at the internal nodes. The main contribution of this chapter is the arrangement of apparently unstructured volumetric layered datasets in a set of heightfields, isolating them from the attribute values, and compressing them into a quadtree. This allows the compression of each part separately, providing excellent compression results. We described a construction algorithm in two phases, a fast access algorithm as well as a lossless random-access compression method for the decoupled heightfields.

The second part of this dissertation is focused on the definition and implementation of rendering algorithms using the data structures introduced previously. After a review of the main concepts and methods used in direct volume rendering, we proposed a real-time rendering method based on GPU raycasting for the visualization of volumetric terrains and geological structures, using again the stack-based representation of terrains as main data structure. To achieve this, different GPU memory layouts are proposed comparing their performance. Also, a comparison with a hierarchical data structure was made, showing similar results. In addition, we implemented useful visual features for geoscientific applications, such as the visualization of boreholes or geological cross sections, or the selective attenuation of strata to visualize internal structures or hidden vector layers. The last contribution regarding the stack-based representation visualization is an efficient and simple method to calculate normal surface vectors in image space which is used in a deferred shading stage.

Because of its fast access, the QuadStack is also suitable for direct rendering; therefore, we presented a compact GPU memory layout and a raycasting-based algorithm for visualization. We compared its performance in terms of storage requirements and rendering speed with datasets from different domains, besides the geological one. We showed that the speed performance of our GPU implementation is similar to the one of

state-of-the-art algorithms, while using only a portion of their memory requirements.

## 8.2 FUTURE WORK

Geomodeling is a very active area of research. Advances made by researchers are rapidly adopted by engineers and geoscience professionals, which results in a constantly evolving industry.

Regarding the content of this dissertation, some aspects that deserve further study are discussed below:

- So far, the stack-based representation has been extensively used for simulating natural processes like thermal or water erosion (Benes and Forsbach, 2001; Št'ava et al., 2008) and snow-covering of mountains (Cordonnier et al., 2018). However, other applications such as the computation of drainage networks (Ortega and Rueda, 2010), the simulation of sediment transport, erosion and deposition (Karszenberg and Bridge, 2008), the groundwater modeling (Carlotto, Silva, and Grzybowski, 2018) or the simulation of lava flows (Hérault et al., 2011) are worth exploring. The operations required for these applications can be fully implemented in GPU in an integrated pipeline that involves a computation stage and a visualization stage, avoiding expensive data transfers between GPU and CPU.
- The data structures and visualization methods presented in this dissertation can be implemented as a module for Open-Source GIS packages such as GRASS (GRASS Development Team, 2016), since these tools handle volumetric terrains and subsurface structures by means of voxel models. Moreover, our rendering algorithm would improve one of the main drawbacks of this kind of tools, i.e., their 3D visualization.
- We can enhance our framework by allowing the representation of heterogeneous materials (Conde-Rodríguez et al., 2015). Material distributions are in fact heterogeneous by its very nature, since a material is usually mixed with those around it. For instance, in the transition between water and clay materials, there are points which present different proportions of them. The representation of this type of materials is a complex task, since the proportions of the different material mixtures must be managed in a proper way. This



feature would provide a more accurate representation and modeling of the strata at surface and subsurface levels.

- We introduced an exchange data model based on GML, a standard file format in geospatial applications. However, other professionals working in fields like engineering or physics could be interested in developing our methods. Therefore, the use of other exchange formats such as NetCDF could be appropriate.
- QuadStack is a totally new data structure; hence, we are still far from having studied all its properties. The study of a *layerity* factor could be very interesting in order to analyze beforehand if a dataset is suitable to be encoded with this data structure. Also, we have not already studied the influence of dataset transformations, like rotations, in the compression ratio.
- QuadStack has been tested with datasets with an orthogonal spatial coherence. However, many other scientific data present this coherence in a cylindrical perspective (medical datasets, 3D CAD models, etc.). Therefore, an extension of the data structure using polar coordinates seems convenient.
- Although it has proven to be fast enough for the tested data, the QuadStack construction algorithm may be time-consuming if the datasets contain many layers. The use of data structures designed for this purpose such as suffix-trees or the use of heuristics to simplify the problem would be good contributions.
- In the same way as the SBRT, the QuadStack can be suitable for the implementation of simulations of geological processes. As other hierarchical data structures, it does not need a complete recalculation when only local zones are affected, but the implementation of these and other algorithms that modify the geodata represented by a QuadStack is an open problem that requires an in-depth study.

Part V

APPENDICES





## PROGRAMMING CUSTOM VISUALIZATION ALGORITHMS ON GPU

---

In this appendix, we show some of the concepts of the graphic GPU programming used in this thesis, as well as the technical details of the visualization algorithms presented. We will explain some features of OpenGL, the graphics API used during this thesis, and why we decided to work with it.

It should be noted that through this dissertation we have used the term *GPU implementation* when we referred to algorithms designed to be compiled and executed in graphics hardware.

### A.1 INTRODUCTION TO GPU PROGRAMMING

In 1963, Sutherland (1963) presented in his doctoral thesis *Sketchpad, A Man-Machine Graphical Communication System*, a pioneer display which launched the research in the new field of computer graphics and the development of graphics hardware. The first graphics workstations released in 1980s were able to overcome operations with 100,000 vertices per second, while the current graphics cards have increased their capabilities to more than 20 billion of vertices per second. Of course, these advances have been the result of changes in the hardware architecture and the way in how they are programmed. Over the years, these architectures have become more flexible and their coding easier by abstracting the programmer from the hardware details.

The evolution of graphics hardware usually has been historically divided into five or six generations, depending on the author. With the arrival of the first generation, some graphics APIs emerged. These APIs serve as an abstraction layer between the actual application and the graphics system, providing high-level interfaces to configure and program the graphics pipeline. The GPUs of the first and second generations barely provide the user with a programmable interface. Only a few configurable parameters and the raster operations could be customized for the particular application. Therefore, the graphics pipeline could be considered mostly fixed. In the third generation, it was included a programmable pipeline in which

*vertex shaders* were introduced. A shader is a program that is executed on the GPU instead on the CPU and modifies a stage of the rendering pipeline. A vertex shader performs operations on the input vertices, to compute a lighting model or to update their geometry. By using this shader it is not possible to remove or add new vertices to the pipeline. *Fragment/pixel shaders*, introduced in the fourth generation, act on the pixels of the final framebuffer and give more versatility than vertex shaders. In these programs, the final pixel color is computed as the result of a function that can be very complex, implementing advanced effects and combining the contribution of several textures. Other types of shaders have been added to the rendering pipeline such as the *tessellation shaders* or the *geometry shaders*. Contrary to the previous shaders explained, these new stages (introduced in the fifth generation) can create or delete geometry. In the case of the tessellation shaders, new primitives can be added to the input data by subdividing them into smallest ones. This operation is very useful to create new geometry on-the-fly in order to reduce the CPU-GPU data communication, considered as a bottleneck in real-time rendering. Geometry shaders receive the vertices returned by the vertex or tessellation stage as actual primitives (for example triangles), allowing the evaluation and transformation of them. The functionality of a tessellation shader can be implemented by a geometry shader, however, due to its flexibility, geometry shaders are less efficient. In modern programmable APIs, the implementation of vertex and fragment shaders is mandatory, while the geometry and tessellation stages are optional.

Further, it was introduced a new type of shader that is not related to the rendering process: the *compute shader*. They arise as an alternative to general GPU computing APIs such as CUDA or OpenCL in order to directly manage the data used in the graphics pipeline. This is a very efficient solution when the processing of the data is totally integrated with its visualization like in fluids or particle simulation applications. Since this stage is not executed within the rendering pipeline, it needs to be explicitly invoked for its execution.

Currently, there exist several APIs that provide access to the GPU programming capabilities described above. The cross-platform OpenGL launched by the Khronos Group Inc. and Direct 3D, included in the set of Microsoft's Direct X APIs, are the most widely used in computer graphics. However, other modern APIs like Metal, released by Apple Inc. or Vulkan, also owned by the Khronos Group are attracting programmers.

The visualization algorithms included in this dissertation have been implemented using the OpenGL API. The reasons for this choice are outlined below:

- This is the most used API in a research context. Therefore, the implementation of many algorithms of the state-of-the-art can be directly compared with the methods presented.
- It is cross-platform. The methods implemented in a platform can be ported almost effortlessly to other systems.
- It is open-source. In this dissertation, we have tried to use open-source technologies as far as possible. In addition, this is a standard created by many companies, so it is vendor-independent.
- The community behind it. Since OpenGL is open-source, there exist many support libraries with interesting functionalities.
- This API can be called from different programming languages. One of them is C++, the standard language for real-time graphics applications as in our case.

Figure A.1 shows the data stream among the graphics application, API and shaders.

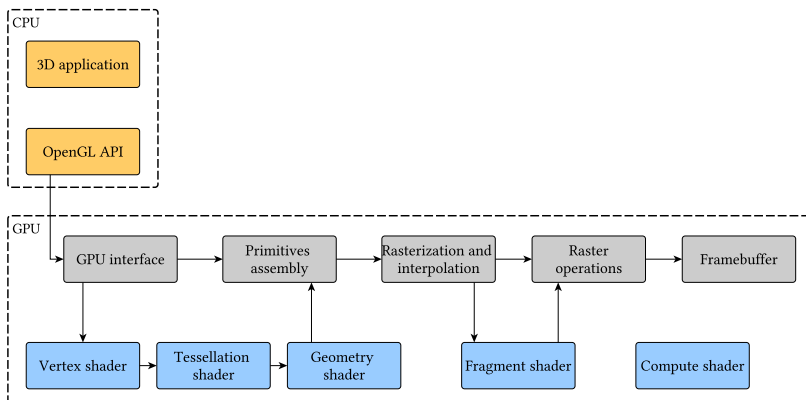


Figure A.1: Programmable pipeline organization. The data flow between the fixed pipeline stages and the programmable shaders are shown.

## A.2 OPENGL API

OpenGL (Open Graphics Library) was released in the beginning of 1990s by Silicon Graphics Inc. along with the OpenGL ARB (Architectural Review Board), composed of members of companies such as Compaq, IBM, Microsoft or Intel among others. This board is responsible for designing and producing the OpenGL specification that must be followed by the concrete vendors.

The OpenGL version 1.0 was published in 1992, and at the time of writing this dissertation the last available version is the 4.6, released in 2017. The functionalities of this API have been adapting to the GPUs generations and the rendering pipeline evolution. The 2.0 version brought the high level shading language used since that time: the OpenGL Shading Language or GLSL. At this point, just vertex and fragment shaders were supported. In 2008 in conjunction with the 3.0 version, the OpenGL ARB introduced the *core profile* and *compatibility modes*, as mechanisms to simplify the addition of new functionalities to OpenGL. The functionalities to be removed in future versions are included in this compatibility mode, while the current functions are kept in the core profile mode. The compatibility profile included, for instance, deprecated fixed functions of vertex and fragment stages.

### A.2.1 Data model

OpenGL allows two ways to store and access data from the graphics applications: by means of buffers and textures. The former are linear blocks of memory than can be seen as generic memory allocations. The latter are used to store multidimensional data like images.

Buffers were thought to provide to the API the greatest possible flexibility. They can be used in many situations. For example, we can send vertex coordinates and other attributes to a vertex shader by using the so-called Vertex Buffer Objects (VBOs) or by a set of unstructured elements to any shader by means of Uniform Buffer Objects (UBOs). Whether a buffer is going to be used as a VBO or a UBO, among others, it is determined by the *target* or *binding point*. For a long time, these targets did not allow a total freedom when using the buffers, since most of them only allow read-only access or had a strong size limitation. OpenGL 4.3 version provided a new type of buffer object: the Shader Storage Buffer Objects (SSBOs), which were introduced in this thesis in Chapter 7. Their main advantages

are the use of almost all available memory in the graphics card and their possibility to be directly updated from any arbitrary shader of the graphics pipeline. However, this flexibility makes it difficult for OpenGL to optimize its access, making other options such as UBOs much more efficient.

Textures are structured storage objects that can be used both for reading and for writing. Likewise buffer objects, textures need targets to declare the type of structure to be used as well as a set of binding points; in this case they are called *texture units*. The number of texture units are limited depending on the specific hardware. There is a wide variety of available targets. We can use 1D, 2D or 3D textures to store for example a transfer function, an image to be applied as a normal map or a medical volumetric image, respectively. Also, a set of textures can be accessed from the same unit texture by means of the array texture target. Interestingly, the content of a buffer object can be accessible from a texture unit by using texture buffer objects (TBOs), therefore technically, TBOs are buffers rather than textures. The main advantage of using buffer textures is the same as that of the previously described SSBOs, the amount of memory available. However, the largest advantage of a SSBO compared with a TBO is that the first can use interface blocks. A block is a kind of struct allowing the combination of many data types like `int`'s, `float`'s or `vec4`'s, that can be defined inside a shader. All of these features make SSBOs the most flexible way for storing data that needs to be accessible from a shader in OpenGL.

### A.3 RAYCASTING IMPLEMENTATION IN GLSL

In this section, we show a full GPU implementation of the DVR raycasting algorithm (Hadwiger et al., 2006) taking as example the visualization method implemented for the stack-based representation of terrains.

Probably, GPU-based raycasting is the most common algorithm for rendering volumetric data. This technique first appeared in 2003 in two different papers (Kruger and Westermann, 2003; Roettger et al., 2003). Previously, its CPU-based version was also extensively used since 1980s (Levoy, 1988). The basic structure of this algorithm was explained in Chapter 7 thereafter, we will see the implementation details, by using a specific shading language, GLSL.

This method is mostly implemented in the fragment shader. Vertex shader is used to receive and transform to view coordinates the proxy geometry, i.e., the AABB used in our algorithms. This shader can also compute the exit points of the ray marching, however, in our implementa-



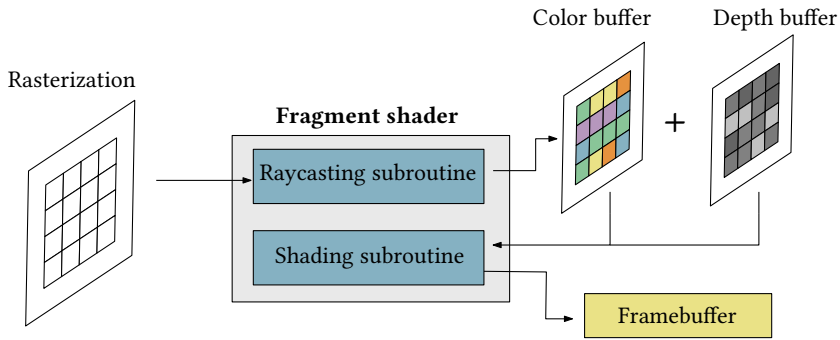


Figure A.2: Deferred shading strategy using GLSL subroutines.

tion the ray termination condition is checked by an inclusion test with the AABB or if the criterion of opacity threshold has been fulfilled.

In the fragment shader, the ray direction is calculated by subtracting the input point interpolated from the vertex shader from the camera position. Then, the ray traversal procedure is performed inside a loop compositing the obtained samples values with a front-to-back scheme. The GLSL code shown in Listing A.1 details this implementation. This is a simplified code with respect to the actual one used. Some optimizations are not included, for example the adaptive sampling when finding *empty space* or the support of the visual operations explained in Chapter 7. Furthermore, a code snippet representing the access of a SBRT encoded in a couple of SSBOs is shown in Listing A.2.

At the end of Listing A.1, the generation of the necessary depth and color buffers for the deferred shading (see Section 7.2.1) is performed. We implemented the deferred shading in OpenGL by means of *shader subroutines* (see Figure A.2). In this way, the functionality of the shader can be changed for a second pass without the need to link a new shader that can be a time-consuming operation if it is executed on every frame. The function responsible for providing the final pixel color is shown in Listing A.3.

The raycasting method implemented for the QuadStack has the same structure than the shown above. The most outstanding difference is the data structure sampling. This step is more complicated as explained in Section 7.3. The estimation of normal vectors does not penalize the algorithm in a significant way since it is solved using an image-space approach.

```

#version 430 core

// Fragment shader outputs
layout( location = 0 ) out vec4 fragColor;
layout( location = 1 ) out vec4 fragDepth;

// Vertex shader inputs
in vec3 entryPoint;

// Constant declaration
const int MAX_SAMPLES = 500;
const float OP_THRESHOLD = 0.9;

uniform vec3 minAABB;
uniform vec3 maxAABB;
uniform vec3 camera;
uniform int unknownIndex;
uniform float stepSize;

subroutine (renderPassType)
void raycastingPass(){
    vec3 dirStep = normalize(camera - entryPoint);
    vec3 position = entryPoint;
    vec4 accumColor = vec4(0.0);
    AABB bBox.min = minAABB;
    AABB bBox.max = maxAABB;
    int pass = 0;

    while(pass++ < MAX_SAMPLES){
        float height = position.y;
        ivec2 texIndices = getIndices(position.xz, bBox, spacing);
        int sample = sampleSbrt(texIndices, height);

        if (isRenderable(sample)) {
            vec4 color = evalTF(sample);
            accumColor += compositing(accumColor, color);
        }

        if( accumColor.a > OP_THRESHOLD ||
           !isInside(bBox, position))
            break;

        position += dirStep * stepSize;
    }
}

```

```

// fill the depth and color buffers for shading
double depth = computeDepthValue(position);

if (tempColor.a > 0.00)
    fragDepth = vec4(vec3(depth), 1.0);
else
    fragDepth = vec4(vec3(1.0,0.0,0.0), 0.5);

fragColor = tempColor;
}

```

Listing A.1: A simplified version of a raycasting function in GLSL

```

uniform int dataCols;
uniform int dataRows;

// SSBOs declaration

layout( binding=2 ) buffer ssboIndices {
    uvec2 indice[];
};

layout( binding=3 ) buffer ssboIntervals {
    float interval[];
};

int sampleSbrt(ivec2 indices, float height) {
    uint index1D = index2Dto1D(indices, dataCols, dataRows);
    uvec2 indiceSampled = indices[index1D]; // Indices SSBO
    uint intervalIndex = indiceSampled.x;
    uint intervalSize = indiceSampled.y;

    for (uint i = 0; i < intervalSize; ++i) {
        float sample = interval[intervalIndex + i];
        // Both elements are packed into a 32 bits float
        vec2 unpackedInterval = unpackHalf2x16(sample);
        float attribute = unpackedInterval.x;
        float currentHeight = unpackedInterval.y;

        if (height >= currentHeight)
            return attribute;
    }
}

```

```
return unknownIndex;
}
```

Listing A.2: Sampling method of a SBRT encoded using SSBOs in GLSL

```
// Texture samples
layout( binding=4 ) uniform sampler2D depthTex;
layout( binding=5 ) uniform sampler2D colorTex;

uniform uint kernelSize;

subroutine (renderPassType)
void shadingPass() {
    vec3 dirStep = camera - entryPoint;
    ivec2 pixel = ivec2(gl_FragCoord.xy);
    vec4 color = texelFetch(colorTex, pixel, 0).rgba;
    vec4 depth = texelFetch(depthTex, pixel, 0).rgba;

    if (depthTexel.a < 0.9)
        discard;

    vec3 shadedColor = vec3(0.0);
    vec3 point3D = getWorldPoint(gl_FragCoord.xy, depthTexel.r);
    normal = surfaceNormal(point3D, kernelSize);

    shadedColor += getDiffuseColor(color.rgb, normal, dirStep);
    shadedColor += getAmbientColor(color.rgb);

    fragColor = vec4(shadedColor, color.a);
}
```

Listing A.3: Implementation of the deferred shading approach in GLSL



DOCUMENTACIÓN EN CASTELLANO

---

La actual normativa de la Universidad de Jaén para la defensa de tesis doctorales, adaptada a las directrices del R.D. 99/2011, y aprobada en Consejo de Gobierno el 6 de febrero de 2012 establece que todas las tesis doctorales escritas en un idioma distinto al castellano deberán incluir los siguientes apartados en dicho idioma: título, índice, introducción, resumen y conclusiones. En base a esta normativa se ha incluido el siguiente anexo.



# REPRESENTACIÓN MULTICAPA PARA SISTEMAS DE INFORMACIÓN GEOLÓGICA

ALEJANDRO GRACIANO

Memoria de tesis presentada para optar  
al grado de Doctor en Informática

DIRECTORES:

Dr. Antonio J. Rueda

Dr. Francisco R. Feito



Universidad de Jaén

Departamento de Informática  
Escuela Politécnica Superior de Jaén  
Universidad de Jaén

Julio 2019





## CONTENIDOS

---

### I INTRODUCCIÓN Y PERSPECTIVA

1	INTRODUCCIÓN	3
1.1	Propósitos y objetivos . . . . .	6
1.2	Organización de esta tesis . . . . .	6
2	FUNDAMENTOS DEL MODELADO GEOLÓGICO	9
2.1	Modelos de campos continuos . . . . .	10
2.1.1	Modelos y estructuras de datos basados en rejillas en el modelado geológico . . . . .	11
2.1.2	Modelos de rejillas para el modelado numérico . . .	13
2.1.3	Modelos de rejillas jerárquicas para el modelado geológico . . . . .	14
2.2	Modelos de objetos discretos . . . . .	15
2.2.1	Modelado geológico con complejos simpliciales . .	16
2.2.2	Modelado geológico con complejos celulares . . .	17
2.2.3	Modelos geológicos ordenados topológicamente .	18
2.3	Estrategias generalizadas para el modelado geológico . . .	19
2.4	Comparación de estructuras de datos para el geomodelado	21
2.4.1	Topología . . . . .	21
2.4.2	Requisitos de memoria . . . . .	23
2.4.3	Consultas y actualización . . . . .	23
2.4.4	Facilidad de construcción . . . . .	23
2.5	Visualización en GIS . . . . .	24

### II ESTRUCTURAS DE DATOS PARA PARA DATOS GEOLÓGICOS POR CAPAS

3	LA REPRESENTACIÓN DE TERRENOS BASADA EN STACKS	31
3.1	Eficiencia en el modelado geológico de campos . . . . .	31
3.2	Trabajos relacionados . . . . .	32
3.3	Definiendo la representación de terrenos basada en stacks	33
3.3.1	Definición matemática . . . . .	33
3.4	Comparativa de requisitos de memoria . . . . .	35
4	UN ESQUEMA FORMAL PARA LA REPRESENTACIÓN DE TER- RENOS BASADA EN STACKS	37
4.1	Introducción . . . . .	37

4.2	Fundamentos de la teoría de geo-átomos . . . . .	38
4.3	Terrenos 3D como geo-campos . . . . .	40
4.4	Terrenos basados en stacks como geo-campos . . . . .	41
4.5	Operaciones con terrenos 3D . . . . .	42
4.5.1	Fundamentos del álgebra de mapas . . . . .	43
4.5.2	Definición de operaciones . . . . .	44
4.6	Un modelo de implementación . . . . .	51
4.6.1	Extendiendo el modelo con clases de SBRT . . . . .	54
4.7	Conclusiones y trabajo futuro . . . . .	55
5	REPRESENTACIÓN EFICIENTE DE INFORMACIÓN BASADA EN STACKS . . . . .	59
5.1	Introducción . . . . .	59
5.2	La estructura de datos QuadStack . . . . .	61
5.2.1	Grupos de stacks . . . . .	61
5.2.2	Grupos de jerarquías de stacks . . . . .	63
5.2.3	Construcción del QuadStack . . . . .	65
5.2.4	Compresión de campos de altura . . . . .	67
5.2.5	Optimizaciones . . . . .	68
5.3	Accediendo al QuadStack . . . . .	69
5.4	Conclusiones . . . . .	71
 <b>III VISUALIZACIÓN DE DATOS GEOLÓGICOS</b>		
6	FUNDAMENTOS DE LA VISUALIZACIÓN DE VOLÚMENES . . . . .	75
6.1	Introducción . . . . .	75
6.2	Revisión de técnicas de visualización directa de volúmenes . . . . .	75
6.2.1	La ecuación del renderizado de volúmenes . . . . .	75
6.2.2	Funciones de transferencia . . . . .	77
6.2.3	Enfoques para el renderizado directo de volúmenes . . . . .	77
6.3	Raycasting . . . . .	79
7	VISUALIZACIÓN EN TIEMPO REAL DATOS BASADOS EN STACKS . . . . .	81
7.1	Introducción . . . . .	81
7.2	Usando raycasting en la representación basada en stacks . . . . .	82
7.2.1	Cálculo de los vectores normales a la superficie . . . . .	84
7.2.2	Codificación de la SBRT en la memoria de la GPU . . . . .	87
7.2.3	Operaciones visuales . . . . .	91
7.2.4	Visualización de capas basadas en SIG . . . . .	92
7.2.5	Análisis del rendimiento . . . . .	96
7.3	Usando raycasting en un QuadStack . . . . .	104

7.3.1	Codificación de un QuadStack en la memoria de la GPU . . . . .	105
7.3.2	Análisis del rendimiento . . . . .	107
7.4	Conclusiones . . . . .	111
<b>IV CONSIDERACIONES FINALES</b>		
8	CONCLUSIONES . . . . .	115
8.1	Resumen de las contribuciones . . . . .	115
8.2	Trabajo futuro . . . . .	117
<b>V ANEXOS</b>		
A	PROGRAMACIÓN DE ALGORITMOS DE VISUALIZACIÓN EN LA GPU . . . . .	121
A.1	Introducción a la programación de la GPU . . . . .	121
A.2	La API OpenGL . . . . .	124
A.2.1	Modelo de datos . . . . .	124
A.3	Implementación del raycasting en la GPU . . . . .	125
B	DOCUMENTACIÓN EN CASTELLANO . . . . .	131
	BIBLIOGRAFÍA . . . . .	151



## INTRODUCCIÓN

---

Tener un conocimiento general de las estructuras geológicas, su organización, sus propiedades y su evolución a lo largo del tiempo es de especial interés para los geocientíficos. Este conocimiento les habilita para poder proporcionar evaluaciones precisas de los recursos o predicciones adecuadas de las amenazas geológicas. El acceso a esta información ha aumentado en gran medida gracias a los avances en las tecnologías de adquisición de datos, que han dado lugar a un gran flujo de valiosa información espacial para los campos relacionados con las geociencias y la geoinformación. La ciencia que busca la integración de estos datos heterogéneos y generar representaciones asistidas por ordenador de las estructuras geológicas, tanto a nivel de la superficie como a nivel subterráneo se denomina **Modelado geológico o Geomodelado**.

Como muchas otras disciplinas, el modelado geológico tiene sus raíces en la Revolución Científica del siglo XV que tomaron muchas ideas árabes y griegas y las renovaron no sólo desde un punto de vista puramente científico. El geomodelado atrajo a muchos eruditos renacentistas, como Leonardo Da Vinci, el cual realizó estudios sobre las formaciones geológicas que fueron utilizados posteriormente para aumentar la precisión de sus creaciones artísticas. La Figura Bo.1.a muestra un boceto datado en 1473 en el que se pueden ver con precisión capas geológicas en el paisaje. Otros artistas como Albrecht Dürer, mejoraron los resultados de Da Vinci añadiendo un mayor realismo y aumentando los detalles (Figure Bo.1.b). Estas observaciones y principios son la base del modelado geológico tal como lo conocemos e inspiraron la cartografía geológica actual.

La fiabilidad con la que se represente un modelo geológico en un ordenador es crucial para su eficiente gestión, procesamiento y visualización. Por lo tanto, el estudio de representaciones de datos espaciales ha sido un punto clave en la investigación geocientífica en las últimas décadas.

Sin embargo, sigue siendo de gran importancia el estudio de nuevas herramientas y estructuras de datos eficientes que cumplan con las llamadas cinco Ms de los Sistemas de Información Geográfica (SIG). Estas estructuras de datos deben ser capaces de **Manejar** los datos espaciales de tal manera que permitan una disposición eficiente y la mantengan actualizada. Estas posibles actualizaciones deben ser **Monitorizadas** para que se



(a) Leonardo Da Vinci - Paisaje (1473)



(b) Albrecht Dürer - Cantera (1506)

Figura Bo.1: Ejemplos de observaciones geológicas en el Renacimiento.

puedan consultar de forma rápida y sencilla. La organización de estos datos es crucial no sólo para una gestión adecuada, sino también para realizar cálculos y **Medidas** sobre éstos de forma eficiente. También es deseable que las estructuras de datos **Modelen** con precisión la complejidad de los datos y que los integren de forma compacta, por ejemplo, combinando la información de la superficie y la del subsuelo en una misma representación. Y, por último, pero no por ello menos importante, estas estructuras de datos deben facilitar la extracción de información para crear **Mapas** y modelos visuales con el fin de tomar decisiones a simple vista. Esta es una característica muy interesante para todas las herramientas y estructuras de datos que manejan datos espaciales.

A pesar de que los requisitos anteriores han sido definidos para herramientas SIG, también deben ser cumplidos por cualquier aplicación geoespacial. El modelado geológico trata fundamentalmente con representaciones volumétricas que raramente son soportadas por los SIG tradicionales. Por ello, a finales de los años ochenta, surgió el concepto de Sistemas de Información Geocientífica (GSIS, por sus siglas en inglés) que define aquellos sistemas dedicados a la gestión y visualización de geodatos 3D. Típicamente, los sistemas GSIS siguen la arquitectura representada en la Figura Bo.2. La arquitectura se divide en tres partes o módulos distintos: el relativo a la adquisición de datos (1), en el que se recoge la información de entrada procedente del trabajo de campo o de los procedimientos de monitorización. El modelo geológico (2) que actúa como núcleo de la arquitectura, establece las estructuras de datos que representan las características geológicas. Este modelo puede separarse en dos submódulos. Por un lado, en el submódulo de modelado geométrico se utilizan métodos de

interpolación para generar la información geométrica. Por otro lado, el submódulo de modelado numérico utiliza modelos matemáticos y predictivos para extrapolar y simular propiedades físicas y fenómenos naturales. A menudo, ambos submódulos están estrechamente conectados. Como último fin (3), estos modelos son usados por los geocientíficos para aplicar métodos de análisis y visualización con la idea de extraer nuevo conocimiento sobre los datos, así como para procesos de toma de decisiones.

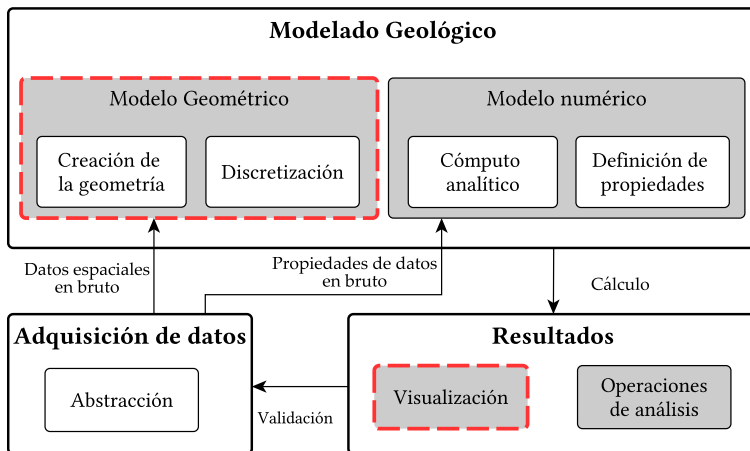


Figura Bo.2: Arquitectura del proceso de modelado geológico adaptado de (Brobrowsky y Marker, 2018) y (Turner, 2006). Las etapas que se tratan en esta tesis se indican con recuadros de color rojo.

La visualización de datos geológicos es una herramienta muy interesante para los profesionales de GIS. Al igual que en otros campos científicos como la radiología o la dinámica de fluidos computacional, disponer de una visión global y en tiempo real de los datos puede proporcionar una primera visión de forma rápida y eficaz. Desde el punto de vista de la informática gráfica, los objetos tridimensionales se obtienen principalmente mediante técnicas de visualización de superficies o volumétricas. Por lo tanto, este último enfoque parece el apropiado para la visualización de datos geológicos, siendo ésta una estrategia muy utilizada en aplicaciones GIS. Existen dos alternativas principales para visualizar datos volumétricos: la renderización indirecta y la directa. El primer método utiliza algoritmos para obtener y visualizar la información de la superficie a partir de los datos volumétricos, ignorando la información interna. En cambio, la segunda define técnicas para visualizar los datos sin aplicar ningún procedimiento



para extraer su superficie. Esta estrategia se basa en la simulación de la interacción entre modelos de transporte de luz y un campo escalar 3D.

Que la visualización se realice en tiempo real es también un aspecto muy apreciado a la hora de realizar operaciones sobre los geodatos. Además, para algunos tipos de operaciones como las simulaciones sería muy beneficioso que tanto los procedimientos de cálculo como los de visualización pudieran sincronizarse, mostrando la operación paso a paso y de forma fluida. Por ejemplo, puede ser muy interesante ver cómo un fenómeno natural como los flujos de lava o la deposición de sedimentos evolucionan en una simulación. El continuo desarrollo del hardware y la aparición de tecnologías como la computación en GPU están contribuyendo a que esto sea posible.

#### PROPÓSITOS Y OBJETIVOS

Teniendo en cuenta la arquitectura de los sistemas GIS descrita en la sección anterior y los requisitos expuestos, el objetivo general de esta tesis es contribuir a la mejora de algunas etapas mediante la investigación y el desarrollo de estructuras de datos y métodos de interés para los geocientíficos. Como se puede ver en la figura [Bo.2](#), esta tesis intentará cubrir los procedimientos de representación geométrica y de discretización situados dentro del submódulo modelado geométrico, y su visualización en tiempos de respuesta interactivos. Los objetivos de este trabajo se enumeran a continuación:

- El estudio y desarrollo de representaciones y estructuras de datos para ser utilizadas en el proceso de modelado geológico. Las estructuras de datos propuestas deben cumplir una serie de especificaciones, en particular una definición simple y una fácil implementación, requisitos razonables de espacio y tiempo de acceso y capacidad para gestionar información geológica tridimensional compleja.
- La definición de un marco formal para validar las representaciones propuestas y definir propiedades y algoritmos con precisión. Esta formalización debe basarse en métodos y teorías estándar de la literatura geoespacial.
- La descripción de un método de renderizado en tiempo real para la visualización de terrenos volumétricos y estructuras geológicas, utilizando para ello las estructuras de datos definidas previamente.

Además, se discutirá la implementación de operaciones visuales útiles en aplicaciones geocientíficas con el fin de validar el uso de estas representaciones.

#### ORGANIZACIÓN DE ESTA TESIS

Esta tesis consta de cuatro partes organizadas en un conjunto de capítulos:

**PARTE I** Esta parte incluye el capítulo actual e introduce al lector en el tema de esta tesis. Además, en la Sección 2 se ofrece una revisión del estado del arte del modelado y la visualización geológica.

**PARTE II** En esta parte se presenta la representación central en la que se sustenta esta tesis: la representación de terrenos basada en stacks, proponiendo su uso como una representación global tanto para la información de superficie como para la del subsuelo. La Sección 3 realiza una descripción de la esta estructura de datos, destacando sus ventajas e inconvenientes en comparación con otras representaciones. En la Sección 4 se presenta un marco formal para la representación de terrenos en 3D basados en stacks. Este esquema se ha derivado de la teoría de geo-átomos, una representación espacial que generaliza los tipos de datos vectoriales y raster. La formalización propuesta se completa con un conjunto de operaciones inspiradas en el conocido álgebra de mapas. Además, se sugiere un modelo de intercambio de datos como ejemplo de implementación, utilizando un estándar como Geography Markup Language. Finalmente, la Sección 5 describe una nueva estructura de datos que mejora las ventajas de la representación basada en stacks en términos de requisitos de almacenamiento y organización de datos: el QuadStack.

**PARTE III** Esta parte explica cómo las estructuras de datos introducidas en la Parte II pueden ser utilizadas para renderizar información 3D de una manera eficiente, haciendo uso de técnicas de visualización directa de volúmenes. En la Sección 6 se realiza una introducción de algunos conceptos y técnicas usados en la visualización de volúmenes. La Sección 7 describe los métodos de visualización para las representaciones introducidas en esta tesis, incluyendo operaciones visuales tales como la visualización de secciones transversales o la inspección de estructuras internas, así como capas de datos vectoriales. Asimismo, esta sección finaliza con la descripción del

método usado para la visualización de datos volumétricos codificados en QuadStacks.

PARTE IV La Sección 8 concluye este trabajo destacando los principales logros y señalando las líneas abiertas que deja esta tesis como trabajo futuro.

## CONCLUSIONES

---

En esta tesis, hemos propuesto el uso de la SBRT y de los QuadStacks como base en aplicaciones de geomodelado. La representación basada en stacks ha sido probada como una representación muy compacta, con un buen compromiso entre sus requisitos de memoria y su velocidad de acceso. Hemos puesto en contexto esta representación con las existentes en el estado del arte, destacando sus ventajas e inconvenientes. A lo largo de los capítulos de esta tesis se han desarrollado diferentes aspectos como la definición formal de la representación y su funcionamiento, sus procedimientos de construcción, sus métodos de acceso y técnicas para su visualización.

A nivel personal, estos últimos cuatro años me han permitido mejorar como investigador y como persona. El trabajo realizado ha generado importantes resultados que han sido publicados o están siendo revisados en revistas y congresos de primera línea en diferentes campos, demostrando la investigación multidisciplinar realizada en esta tesis.

## RESUMEN DE LAS CONTRIBUCIONES

En la primera parte de esta tesis, se expone que la representación basada en stacks ha sido utilizada sólo en unos pocos trabajos enfocados a la visualización, usándose únicamente como una estructura de datos auxiliar, a pesar de sus interesantes características. Por lo tanto, hemos sugerido esta representación como la principal para la gestión de datos geológicos tanto a nivel de superficie como subterráneo. También hemos destacado sus bajos requisitos de almacenamiento en comparación con otras estructuras de datos usadas comúnmente como los modelos de vóxeles o los octrees.

A pesar de que se ha proporcionado una definición clara e intuitiva de la SBRT (Capítulo 3), en el Capítulo 4 hemos presentado un marco formal para validar esta representación y definir sus propiedades y operaciones principales de forma precisa. Para esta formalización nos hemos basado en la teoría de geo-átomos; un marco que unifica los modelos de datos de campos y objetos discretos en una representación espacial general. Para llevar a cabo esto, hemos comenzado por desarrollar un esquema en el que se utiliza la teoría de geo-átomos para representar modelos de

vóxeles. Este paso previo sirve de apoyo para describir la formalización final de la SBRT. Además, hemos definido un conjunto de operaciones espaciales habituales para esta representación, utilizando el marco teórico proporcionado por el álgebra de mapas. A partir de estas operaciones fundamentales, se pueden construir otras operaciones geoespaciales o simulaciones geofísicas más complejas. Para finalizar, hemos sugerido un modelo de datos y una implementación que amplía el concepto de cobertura proporcionado por el estándar Geography Markup Language.

Tras definir esta representación, nos propusimos mejorarla para reducir sus requisitos de almacenamiento (Capítulo 5). Un QuadStack es una nueva estructura de datos que no sólo saca partido de la estructura de capas de los datos como la SBR, sino también la coherencia espacial horizontal dentro de las propias capas. Para ello se hace uso de un quadtree en el que se representa la estructura de capas en sus nodos internos.

La principal contribución de este capítulo es la separación de la información de altura de los valores categóricos como el material, incluso en datos aparentemente no estructurados en una secuencia de capas. De esta forma conseguimos la compresión de cada parte por separado, proporcionando excelentes resultados de compresión. A continuación, se ha descrito un algoritmo de construcción en dos fases, un algoritmo de acceso rápido y un método de compresión de acceso aleatorio para la información de altura.

La segunda parte de esta tesis se centra en la definición e implementación de algoritmos de visualización utilizando las estructuras de datos introducidas anteriormente. Tras realizar una revisión de los principales términos y métodos utilizados en el campo de la visualización directa de volúmenes, se ha propuesto un método de renderizado en tiempo real acelerado por GPU para la visualización de terrenos volumétricos y estructuras geológicas, utilizando nuevamente la representación de terrenos basada en stacks como estructura de datos principal. Para llevar a cabo esto, hemos propuesto y comparado diferentes esquemas de memoria en la GPU para su codificación. Además, también se ha realizado una comparación con una estructura de datos jerárquica, obteniendo resultados similares en velocidad de visualización. Complementariamente, hemos implementado un conjunto de operaciones visuales útiles para aplicaciones geocientíficas, como la visualización de catas o secciones transversales geológicas, o la atenuación selectiva de capas de materiales. La última contribución de esta tesis en cuanto a la visualización de la representación basada en stacks, es la implementación de un método simple y eficiente para el cálculo de

vectores normales a la superficie. Este método utiliza un enfoque basado en el espacio de imagen y es usado en una etapa diferida de sombreado.

Debido a su eficiente método de acceso, los QuadStacks también son adecuados para su visualización directa, por lo que también se ha presentado un algoritmo de visualización para esta estructura de datos. Tras evaluar su rendimiento en términos de requisitos de memoria y velocidad de visualización con datos procedentes de diferentes dominios, hemos demostrado que nuestra implementación es comparable a la de otros algoritmos avanzados, usando únicamente una porción de la memoria necesitada por éstos.

#### TRABAJO FUTURO

El modelado geológico es un área de investigación muy activa. Los avances realizados por investigadores son adoptados rápidamente por ingenieros y profesionales geoespaciales, lo que resulta en una industria en constante evolución.

Una vez resumido el contenido de esta tesis, a continuación discutimos los aspectos que merecen ser estudiados más a fondo:

- Hasta ahora la representación basada en stacks se ha utilizado para simular procesos como la erosión térmica o hidráulica (Benes y Forsbach, 2001; Št'ava y col., 2008) o la cobertura de nieve en las montañas (Cordonnier y col., 2018). Sin embargo, vale la pena explorar otras aplicaciones como el cálculo de redes de drenaje (Ortega y Rueda, 2010), la simulación del transporte, erosión y deposición de sedimentos (Karssenbergy Bridge, 2008), el modelado de aguas subterráneas (Carlotto, Silva y Grzybowski, 2018) o la simulación de flujos de lava (Hérault y col., 2011). Estas operaciones pueden implementarse completamente en la GPU en un sistema integrado que incluya una fase de cálculo en la GPU y una fase de visualización, lo que evitaría costosas transferencias de datos entre la CPU y la GPU.
- Las estructuras de datos y métodos de visualización que han sido presentados en esta tesis se pueden añadir como un módulo a aplicaciones informáticas GIS de código abierto como GRASS (GRASS Development Team, 2016), ya que estas herramientas únicamente manejan terrenos volumétricos y estructuras subterráneas por medio de modelos de vóxeles. Además, nuestro algoritmo de visuali-

zación mejoraría uno de los principales inconvenientes de este tipo de herramientas, es decir, su visualización en 3D.

- También podemos mejorar la representación basada en stacks permitiendo el modelado de materiales heterogéneos (Conde-Rodríguez y col., 2015). De hecho, la distribución de materiales geológicos es heterogénea por su propia naturaleza, ya que un material suele mezclarse con los que lo rodean. Por ejemplo, en la transición entre el agua y materiales arcillosos, hay puntos que presentan diferentes proporciones de ellos. La representación de este tipo de materiales es una tarea compleja, ya que las proporciones de las diferentes mezclas de materiales deben ser modeladas de forma adecuada. Esta característica proporcionaría una representación más precisa de los estratos tanto a nivel de superficie como subterráneo.
- En esta tesis hemos introducido un modelo de intercambio de datos basado en GML, un formato estándar en aplicaciones geoespaciales. Sin embargo, otros profesionales que trabajan en campos como la ingeniería o la física podrían estar interesados en desarrollar nuestros métodos. Por lo tanto, puede ser apropiada su implementación en otros formatos de intercambio como NetCDF.
- El QuadStack es una estructura de datos totalmente nueva, por lo que todavía estamos lejos de haber estudiado todas sus propiedades. El estudio de un factor de *estratificación* puede ser muy interesante para analizar de antemano si un dataset es adecuado para ser codificado con esta estructura de datos. Otros aspectos como la influencia de las rotaciones de los datos en la relación de compresión tampoco han sido estudiados.
- Los QuadStacks han sido probados con conjuntos de datos que presentan una coherencia espacial ortogonal, sin embargo muchos otros datos científicos presentan esta coherencia en una perspectiva cilíndrica (imágenes médicas, modelos CAD, etc.). Por lo tanto, sería interesante ampliar la estructura de datos para que utilice coordenadas polares en lugar de cartesianas.
- Aunque se ha demostrado que es lo suficientemente rápido para los datos probados, el algoritmo de construcción de QuadStacks puede consumir mucho tiempo si los datos contienen muchas capas. El uso de estructuras de datos diseñadas para este fin, como los árboles de

sufijos o el uso de heurísticas para simplificar el problema serían contribuciones interesantes.

- Al igual que la SBRT, un QuadStack puede ser adecuado para simular procesos geológicos. Al igual que otras estructuras de datos jerárquicas, no necesita un recálculo completo al realizar cambios en zonas locales. Sin embargo, la implementación de este tipo de operaciones usando QuadStacks es un problema abierto que requiere de un estudio en profundidad.





## BIBLIOGRAPHY

---

- Aage, N., E. Andreassen, B. S. Lazarov, and O. Sigmund (2017). «Giga-voxel computational morphogenesis for structural design.» In: *Nature* 550, pp. 84–86.
- Agterberg, F. (2018). *Handbook of Mathematical Geosciences*. ISBN: 978-3-319-78998-9.
- Akenine-Möller, T., E. Haines, and N. Hoffman (2009). *Real-time rendering*. A. K. Peters, Ltd., p. 1045. ISBN: 9871568814247.
- Ammann, L., O. Génevaux, and J. J.-M. Dischler (2010). «Hybrid rendering of dynamic heightfields using ray-casting and mesh rasterization.» In: *Proceedings of Graphics Interface 2010*, pp. 161–168. ISSN: 07135424.
- Andújar, C. (2010). «Topographic Map Visualization from Adaptively Compressed Textures.» In: *Computer Graphics Forum* 29.3, pp. 1083–1092.
- Antoniou, P., J. Holub, C. S. Iliopoulos, B. Melichar, and P. Peterlongo (2006). «Finding Common Motifs with Gaps Using Finite Automata.» In: *Implementation and Application of Automata*. Ed. by O. H. Ibarra and H.-C. Yen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 69–77.
- Antoniou, P., M. Crochemore, C. S. Iliopoulos, and P. Peterlongo (2007). «Application of suffix trees for the acquisition of common motifs with gaps in a set of strings.» In: *LATA 2007. Proceedings of the 1st International Conference on Language and Automata Theory and Applications*. Pp. 57–66.
- Arnone, E., A. Francipane, A. Scarbaci, C. Puglisi, and L. V. Noto (2016). «Effect of raster resolution and polygon-conversion algorithm on landslide susceptibility mapping.» In: *Environmental Modelling & Software* 84, pp. 467–481. ISSN: 13648152.
- Arroyo Ogori, K. (2016). «Higher-dimensional modelling of geographic information.» PhD thesis. Delft University of Technology.
- Arroyo Ogori, K., H. Ledoux, and J. Stoter (2015). «An evaluation and classification of n D topological data structures for the representation of objects in a higher-dimensional GIS.» en. In: *International Journal of Geographical Information Science* 29.5, pp. 825–849. ISSN: 1365-8816.
- Baert, J., A. Lagae, and P. Dutré (2012). «Out-of-Core Construction of Sparse Voxel Octrees.» In: *Computer Graphics Forum*. ISSN: 1467-8659.

- Becher, M., M. Krone, G. Reina, and T. Ertl (2019). «Feature-based volumetric terrain generation and decoration.» In: *IEEE Transactions on Visualization and Computer Graphics* 25.2, pp. 1283–1296. ISSN: 19410506.
- Benes, B. and R. Forsbach (2001). «Layered data representation for visual simulation of terrain erosion.» In: *Proceedings Spring Conference on Computer Graphics*. ISBN: 0-7695-1215-1.
- Berry, P., S. Bonduá, V. Bortolotti, C. Cormio, and E. Vasini (2014). «A GIS-based open source pre-processor for georesources numerical modeling.» In: *Environmental Modelling & Software* 62, pp. 52–64. ISSN: 13648152.
- Bieri, H. (1995). «Nef Polyhedra: A Brief Introduction.» In: *Geometric Modelling*. Ed. by H. Hagen, G. Farin, and H. Noltemeier. Vienna: Springer Vienna, pp. 43–60. ISBN: 978-3-7091-7584-2.
- Bobrowsky, P. T. and B. Marker (2018). *Encyclopedia of Engineering Geology*. Springer, p. 979. ISBN: 978-3-319-73566-5.
- Bonomi, T. (2009). «Database development and 3D modeling of textural variations in heterogeneous, unconsolidated aquifer media: Application to the Milan plain.» In: *Computers & Geosciences* 35.1, pp. 134–145. ISSN: 00983004.
- Breunig, M., P. V. Kuper, E. Butwilowski, A. Thomsen, M. Jahn, A. Ditrach, M. Al-Doori, D. Golovko, and M. Menninghaus (2016). «The story of DB4Geo – A service-based geo-database architecture to support multi-dimensional data analysis and visualization.» In: *ISPRS Journal of Photogrammetry and Remote Sensing* 117, pp. 187–205. ISSN: 09242716.
- Brink, L. van den, J. Stoter, and S. Zlatanova (2013). «Establishing a national standard for 3D topographic data compliant to CityGML.» en. In: *International Journal of Geographical Information Science* 27.1, pp. 92–113. ISSN: 1365-8816.
- Brisson, E. (1993). «Representing geometric structures in d dimensions: Topology and order.» In: *Discrete & Computational Geometry* 9.1, pp. 387–426. ISSN: 01795376.
- Brooks, S. and J. L. Whalley (2008). «Multilayer hybrid visualizations to support 3D GIS.» In: *Computers, Environment and Urban Systems* 32.4, pp. 278–292. ISSN: 01989715.
- Cacciari, P. P. and M. M. Futai (2017). «Modeling a Shallow Rock Tunnel Using Terrestrial Laser Scanning and Discrete Fracture Networks.» In: *Rock Mechanics and Rock Engineering*. ISSN: 0723-2632.
- Camara, G., M. J. Egenhofer, K. Ferreira, P. Andrade, G. Queiroz, A. Sanchez, J. Jones, and L. Vinhas (2014). «Fields as a Generic Data Type for Big Spatial Data.» In: *Geographic Information Science*, in press. ISSN: 16113349.

- Carlotto, T., R. da Silva, and J. Grzybowski (2018). «A GPGPU-accelerated implementation of groundwater flow model in unconfined aquifers for heterogeneous and anisotropic media.» In: *Environmental Modelling & Software* 101, pp. 64–72. ISSN: 1364-8152.
- Carré, F. and M. C. Girard (2002). «Quantitative mapping of soil types based on regression kriging of taxonomic distances with landform and land cover attributes.» In: *Geoderma* 110.3-4, pp. 241–263. ISSN: 00167061.
- Caumon, G., P. Collon-Drouaillet, C. Le Carlier De Veslud, S. Viseur, and J. Sausse (2009). «Surface-based 3D modeling of geological structures.» In: *Mathematical Geosciences* 41.8, pp. 927–945. ISSN: 18748961.
- Caumon, G., B. Lévy, L. Castanié, and J. C. Paul (2005). «Visualization of grids conforming to geological structures: A topological approach.» In: *Computers and Geosciences* 31.6, pp. 671–680. ISSN: 00983004.
- Caumon, G., G. G. Gray, C. Antoine, and M.-O. Titeux (2012). «3D implicit stratigraphic model building from remote sensing data on tetrahedral meshes: theory and application to a regional model of La Popa Basin, NE Mexico.» In: *IEEE Transactions on Geoscience and Remote Sensing* 51.3, pp. 1613–1621. ISSN: 0196-2892.
- Čomić, L., L. De Floriani, F. Iuricich, and U. Fugacci (2014). «Topological modifications and hierarchical representation of cell complexes in arbitrary dimensions.» In: *Computer Vision and Image Understanding* 121, pp. 2–12. ISSN: 10773142.
- Conde-Rodríguez, F., J. C. Torres-Cantero, A. L. García-Fernández, and F. R. Feito-Higuera (2015). «A comprehensive framework for modeling heterogeneous objects.» In: *The visual Computer*. ISSN: 0178-2789.
- Coors, V. (2003). «3D-GIS in networking environments.» In: *Computers, Environment and Urban Systems* 27.4, pp. 345–357. ISSN: 01989715.
- Cordeiro, J. P. C., G. Camara, U. Moura de Freitas, and F. Almeida (2009). «Yet another map algebra.» In: *GeoInformatica* 13.2, pp. 183–202. ISSN: 13846175.
- Cordonnier, G., M. P. Cani, B. Benes, J. Braun, and E. Galin (2017). «Sculpting Mountains: Interactive Terrain Modeling Based on Subsurface Geology.» In: *IEEE Transactions on Visualization and Computer Graphics* PP.99, pp. 1–1. ISSN: 1077-2626.
- Cordonnier, G., P. Ecomier, E. Gali, J. Gain, B. Benes, and M.-P. Cani (2018). «Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches.» In: *Computer Graphics Forum (To appear)*. Wiley Online Library.

- Cova, T. J. and M. F. Goodchild (2002). «Extending geographical representation to include fields of spatial objects.» In: *International Journal of Geographical Information Science* 16.6, pp. 509–532. ISSN: 1365-8816.
- Crespin, B., R. Bézin, X. Skapin, O. Terraz, and P. Meseure (2014). «Generalized maps for erosion and sedimentation simulation.» In: *Computers & Graphics* 45, pp. 1–16. ISSN: 00978493.
- Cui, Y., Q. Li, Q. Li, J. Zhu, C. Wang, K. Ding, D. Wang, and B. Yang (2017). «A Triangular Prism Spatial Interpolation Method for Mapping Geological Property Fields.» In: *ISPRS International Journal of Geo-Information* 6.8, p. 241.
- Dado, B., T. R. Kol, P. Bauszat, J. M. Thiery, and E. Eisemann (2016). «Geometry and attribute compression for voxel scenes.» In: *Computer Graphics Forum* 35.2, pp. 397–407. ISSN: 14678659.
- De Cola, L. and N. Montagne (1993). «The pyramid system for multiscale raster analysis.» In: *Computers and Geosciences* 19.10, pp. 1393–1404. ISSN: 00983004.
- De Kemp, E. A. (1999). «Visualization of complex geological structures using 3-D Bézier construction tools.» In: *Computers and Geosciences* 25.5, pp. 581–597. ISSN: 00983004.
- De Oliveira Miranda, A. C., W. W. M. Lira, R. C. Marques, A. M. B. Pereira, J. B. Cavalcante-Neto, and L. F. Martha (2014). «Finite element mesh generation for subsurface simulation models.» In: *Engineering with Computers*, pp. 1–20. ISSN: 01770667.
- De Toledo, R., B. Wang, and B. Levy (2008). «Geometry Textures and Applications.» In: *Computer Graphics Forum*. ISSN: 1467-8659.
- DeMers, M. (2002). *GIS Modeling in Raster*. Wiley. ISBN: 9780471319658.
- Denver, L. and D. Phillips (1990). «Stratigraphic geocellular modeling.» In: *Geobyte; (USA)* 5.1. ISSN: 0885-6362.
- Dunstan, S. P. and A. J. B. Mill (1989). «Spatial indexing of geological models using linear octrees.» In: *Computers and Geosciences* 15.8, pp. 1291–1301. ISSN: 00983004.
- Ebner, M., F. Geldmacher, F. Marone, M. Stampanoni, and V. Wood (2013). «X-Ray tomography of porous, transition metal oxide based lithium ion battery electrodes.» In: *Advanced Energy Materials* 3.7, pp. 845–850. ISSN: 614-6832.
- Egenhofer, M. J. (1989). «A formal definition of binary topological relationships.» In: *Foundations of Data Organization and Algorithms: 3rd International Conference, FODO 1989 Paris, France, June 21–23, 1989 Pro-*

- ceedings. Ed. by W. Litwin and H.-J. Schek. Springer Berlin Heidelberg, pp. 457–472. ISBN: 978-3-540-46186-9.
- Egenhofer, M. J. and R. D. Franzosa (1991). «Point-set topological spatial relations.» en. In: *International journal of geographical information systems* 5.2, pp. 161–174. ISSN: 0269-3798.
- Fisher, T. R. and W. R. Q. (1992). «Three-dimensional solid modeling of geo-objects using Non-Unifrom Rational B-Splines (NURBS).» In: *Three-Dimensional Modeling with Geoscientific Information Systems*. Ed. by A. K. Turner. Dordrecht: Springer Netherlands, pp. 123–141. ISBN: 978-94-011-2556-7.
- Florian Wellmann, J., A. Croucher, and K. Regenauer-Lieb (2012). «Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT.» In: *Computers and Geosciences* 43, pp. 197–206. ISSN: 00983004.
- Foged, N., P. a. Marker, a. V. Christansen, P. Bauer-Gottwein, F. Jørgensen, a. S. Høyer, and E. Auken (2014). «Large-scale 3-D modeling by integration of resistivity models and borehole data through inversion.» In: *Hydrology and Earth System Sciences* 18.11, pp. 4349–4362. ISSN: 1607-7938.
- Franklin, W. R. (1995). «Compressing Elevation Data.» In: *Advances in Spatial Databases*. Ed. by M. J. Egenhofer and J. R. Herring. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 385–404.
- GRASS Development Team (2016). *Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.0*. Open Source Geospatial Foundation.
- Galera, C., C. Bennis, I. Moretti, and J. L. Mallet (2003). «Construction of coherent 3D geological blocks.» In: *Computers and Geosciences* 29, pp. 971–984. ISSN: 00983004.
- Galin, E., E. Guérin, A. Peytavie, G. Cordonnier, M.-P. Cani, B. Benes, and J. Gain (2019). «A Review of Digital Terrain Modeling.» In: *Computer Graphics Forum*. ISSN: 1467-8659.
- Gold, C. and M. A. Mostafavi (2000). «Towards the global GIS.» In: *ISPRS Journal of Photogrammetry and Remote Sensing* 55.3, pp. 150–163. ISSN: 09242716.
- Gong, J., P. Cheng, R. Liu, D. Li, and S. Liu (2002). «Study on 3D modeling and visualization in geological exploration engineering.» In: *Proceeding ISPRS Commission II Symposium on Integrated Systems for Spatial Data Production, Custodian and Decision Support*, pp. 133–138.

- Gong, J., P. Cheng, and Y. Wang (2004). «Three-dimensional modeling and application in geological exploration engineering.» In: *Computers and Geosciences* 30.4, pp. 391–404. ISSN: 00983004.
- Goodchild, M. and P. Kyriakidis (2005). «Uncertainty and interoperability: the areal interpolation problem.» In: pp. 1–9.
- Goodchild, M. F. (1992). «Geographical data modeling.» In: *Computers and Geosciences* 18.4, pp. 401–408. ISSN: 00983004.
- Goodchild, M. F., M. Yuan, and T. J. Cova (2007). «Towards a general theory of geographic representation in GIS.» In: *International Journal of Geographical Information Science* 21.3, pp. 239–260. ISSN: 1365-8816.
- Goodchild, M. F., M. J. Egenhofer, K. K. Kemp, D. M. Mark, and E. Sheppard (1999). «Introduction to the Varenus project.» In: *International Journal of Geographical Information Science* 13.8, pp. 731–745. ISSN: 13623087.
- Gröger, G. and L. Plümer (2012). «CityGML – Interoperable semantic 3D city models.» In: *ISPRS Journal of Photogrammetry and Remote Sensing* 71, pp. 12–33. ISSN: 09242716.
- Guillen, A., P. Calcagno, G. Courrioux, A. Joly, and P. Ledru (2008). «Geological modelling from field data and geological knowledge. Part II. Modelling validation using gravity and magnetic data inversion.» In: *Physics of the Earth and Planetary Interiors* 171.1-4, pp. 158–169. ISSN: 00319201.
- Gunnink, J. L., D. Maljers, S. F. Van Gessel, A. Menkovic, and H. J. Hummelman (2013). «Digital Geological Model (DGM): A 3D raster model of the subsurface of the Netherlands.» In: *Geologie en Mijnbouw/Netherlands Journal of Geosciences* 92.1, pp. 33–46. ISSN: 00167746.
- Guo, F., H. Li, W. Daughton, and Y.-H. Liu (2014). «Formation of Hard Power Laws in the Energetic Particle Spectra Resulting from Relativistic Magnetic Reconnection.» In: *Phys. Rev. Lett.* 113 (15), p. 155005.
- Guo, J., L. Wu, W. Zhou, J. Jiang, and C. Li (2016). «Towards Automatic and Topologically Consistent 3D Regional Geological Modeling from Boundaries and Attitudes.» In: *ISPRS International Journal of Geo-Information* 5.2, p. 17. ISSN: 2220-9964.
- Hachenberger, P., L. Kettner, and K. Mehlhorn (2007). «Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments.» In: *Computational Geometry: Theory and Applications* 38.1-2, pp. 64–99. ISSN: 09257721.
- Hadwiger, M., J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel (2006). *Real-time Volume Graphics*. A. K. Peters, Ltd., p. 497. ISBN: 1568812663.

- Hillier, M. J., E. M. Schetselaar, E. A. de Kemp, and G. Perron (2014). «Three-Dimensional Modelling of Geological Surfaces Using Generalized Interpolation with Radial Basis Functions.» In: *Mathematical Geosciences* 46.8, pp. 931–953. ISSN: 18748953.
- Hoffmann, J., C. Scheidt, A. Barfod, and J. Caers (2017). «Stochastic simulation by image quilting of process-based geological models.» In: *Computers and Geosciences* 106.May, pp. 18–32. ISSN: 00983004.
- Holt, T., W. Freiler, F. Gschwantner, H. Doleisch, G. Heinemann, and M. Hadwiger (2012). «SeiVis: An Interactive Visual Subsurface Modeling Application.» In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2226–2235. ISSN: 1077-2626.
- Hérault, A., G. Bilotta, A. Vicari, E. Rustico, and C. D. Negro (2011). «Numerical simulation of lava flow using a GPU SPH model.» In: *Annals of Geophysics* 54.5. ISSN: 2037-416X.
- Hussain, A. J., A. Al-Fayadh, and N. Radi (2018). «Image compression techniques: A survey in lossless and lossy algorithms.» In: *Neurocomputing* 300, pp. 44–69. ISSN: 18728286.
- Iliopoulos, C. S., J. Mchugh, P. Peterlongo, N. Pisanti, W. Rytter, and M.-F. Sagot (2005). «A first approach to finding common motifs with gaps.» In: *International Journal of Foundations of Computer Science* 16.06, pp. 1145–1154.
- Jansen, G., R. Sohrabi, and S. A. Miller (2017). «HULK – Simple and fast generation of structured hexahedral meshes for improved subsurface simulations.» In: *Computers & Geosciences* 99.July 2016, pp. 159–170. ISSN: 00983004.
- Jjumba, A. and S. Dragičević (2015). «Integrating GIS-Based Geo-Atom Theory and Voxel Automata to Simulate the Dispersal of Airborne Pollutants.» In: *Transactions in GIS* 19.4, pp. 582–603. ISSN: 14679671.
- (2016). «Spatial indices for measuring three-dimensional patterns in a voxel-based space.» In: *Journal of Geographical Systems*. ISSN: 1435-5930.
- Jones, M. D., M. Farley, J. Butler, and M. Beardall (2010). «Directable weathering of concave rock using curvature estimation.» In: *IEEE Transactions on Visualization and Computer Graphics* 16.1, pp. 81–94. ISSN: 10772626.
- Jönsson, D., P. Ganestam, A. Ynnerman, M. Doggett, and T. Ropinski (2012). «Explicit Cache Management for Volume Ray-Casting on Parallel Architectures.» In: *EG Symposium on Parallel Graphics and Visualization (EGPGV)*. Eurographics, pp. 31–40.
- Jørgensen, F., R. R. Møller, L. Nebel, N.-P. Jensen, A. V. Christiansen, and P. B. E. Sandersen (2013). «A method for cognitive 3D geological voxel



- modelling of AEM data.» In: *Bulletin of Engineering Geology and the Environment* 72.3-4, pp. 421–432. ISSN: 1435-9529.
- Jørgensen, F., A. S. Høyer, P. B. Sandersen, X. He, and N. Foged (2015). «Combining 3D geological modelling techniques to address variations in geology, data type and density - An example from Southern Denmark.» In: *Computers and Geosciences* 81, pp. 53–63. ISSN: 00983004.
- Kadosh, A., D. Cohen-Or, and R. Yagel (2003). «Tricubic Interpolation of Discrete Surfaces for Binary Volumes.» In: *IEEE Transactions on Visualization and Computer Graphics* 9.4, pp. 580–586. ISSN: 10772626.
- Karimipour, F., M. R. Delavar, and A. U. Frank (2010). «A simplex-based approach to implement dimension independent spatial analyses.» In: *Computers & Geosciences* 36.9, pp. 1123–1134. ISSN: 00983004.
- Karssenberg, D. and J. S. Bridge (2008). «A three-dimensional numerical model of sediment transport, erosion and deposition within a network of channel belts, floodplain and hill slope: Extrinsic and intrinsic controls on floodplain dynamics and alluvial architecture.» In: *Sedimentology* 55.6, pp. 1717–1745. ISSN: 13653091.
- Kaufmann, O. and T. Martin (2009). «3D geological modelling from boreholes, cross-sections and geological maps, application over former natural gas storages in coal mines.» In: *Computers and Geosciences* 35.1, pp. 70–82. ISSN: 00983004.
- Kessler, H., S. Mathers, and H.-G. Sobisch (2009). «The capture and dissemination of integrated 3D geospatial knowledge at the British Geological Survey using GSI3D software and methodology.» In: *Computers & Geosciences* 35.6, pp. 1311–1321. ISSN: 00983004.
- Kjenstad, K. (2013). «On the hyperfield or field-of-field concept.» In: *International Journal of Geographical Information Science* 27.5, pp. 963–985. ISSN: 1365-8816.
- Knoll, A., S. Thelen, I. Wald, C. D. Hansen, H. Hagen, and M. E. Papka (2011). «Full-resolution interactive CPU volume rendering with coherent BVH traversal.» In: *2011 IEEE Pacific Visualization Symposium*, pp. 3–10.
- Koca, Ç. and U. Güdükbay (2014). «A hybrid representation for modeling, interactive editing, and real-time visualization of terrains with volumetric features.» en. In: *International Journal of Geographical Information Science* 28.9, pp. 1821–1847. ISSN: 1365-8816.
- Kruger, J. and R. Westermann (2003). «Acceleration Techniques for GPU-based Volume Rendering.» In: *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. VIS '03. Washington, DC, USA: IEEE Computer Society, pp. 38–. ISBN: 0-7695-2030-8.

- Lan, H., C. Derek Martin, and C. Lim (2007). «RockFall analyst: A GIS extension for three-dimensional and spatially distributed rockfall hazard modeling.» In: *Computers & Geosciences* 33.2, pp. 262–279. ISSN: 00983004.
- Le, H. H., P. Gabriel, J. Gietzel, and H. Schaeben (2013). «An object-relational spatio-temporal geoscience data model.» In: *Computers & Geosciences* 57, pp. 104–115. ISSN: 00983004.
- Ledoux, H. and C. M. Gold (2008). «Modelling three dimensional geoscientific fields with the Voronoi diagram and its dual.» In: *International Journal of Geographical Information Science* 22.5, pp. 547–574. ISSN: 1365-8816.
- Ledoux, H. (2013). «On the Validation of Solids Represented with the International Standards for Geographic Information.» In: *Computer-Aided Civil and Infrastructure Engineering* 28.9, pp. 693–706. ISSN: 10939687.
- Lee, S., J. Suh, and H. D. Park (2015). «BoreholeAR: A mobile tablet application for effective borehole database visualization using an augmented reality technology.» In: *Computers and Geosciences* 76, pp. 41–49. ISSN: 00983004.
- Leeuwenburgh, O., J. Brouwer, and M. Trani (2011). «Ensemble-based conditioning of reservoir models to seismic data.» In: *Computational Geosciences* 15.2, pp. 359–378. ISSN: 14200597.
- Lefebvre, S., S. Hornus, and F. Neyret (2005). «Octree Textures on the GPU.» In: *Programming Techniques for High-performance Graphics and General-Purpose Computation*. Ed. by P. M. editors. Vol. GPU Gems 2. Addison Wesley, pp. 595–613.
- Lemon, A. M. and N. L. Jones (2003). «Building solid models from boreholes and user-defined cross-sections.» In: *Computers and Geosciences* 29.5, pp. 547–555. ISSN: 00983004.
- Levoy, M. (1988). «Display of Surfaces from Volume Data.» In: *IEEE Comput. Graph. Appl.* 8.3, pp. 29–37. ISSN: 0272-1716.
- Li, J. and A. D. Heap (2014). «Spatial interpolation methods applied in the environmental sciences: A review.» In: *Environmental Modelling & Software* 53, pp. 173–189. ISSN: 13648152.
- Li, J., Y. Jiang, C. Yang, Q. Huang, and M. Rice (2013). «Visualizing 3D/4D environmental data using many-core graphics processing units (GPUs) and multi-core central processing units (CPUs).» In: *Computers & Geosciences* 59, pp. 78–89. ISSN: 00983004.
- Li, Q., Q. Li, X. Liu, Z. Wei, and Q. Dong (2018). «Isosurface Algorithm Based on Generalized Three Prism Voxel.» In: *Advances in Image and*

- Graphics Technologies*. Singapore: Springer Singapore, pp. 20–31. ISBN: 978-981-10-7389-2.
- Li, X., X. Li, and D. Zhang (2018). «Generalized prism grid: a pillar-based unstructured grid for simulation of reservoirs with complicated geological geometries.» In: *Computational Geosciences* 22.6, pp. 1561–1581. ISSN: 1573-1499.
- Li, Z., C. Zhu, and C. Gold (2004). *Digital terrain modeling: principles and methodology*. CRC press.
- Lidal, E. M., H. Hauser, and I. Viola (2012). «Design Principles for Cut-away Visualization of Geological Models.» In: *Proc. Spring Conference on Computer Graphics (SCCG 2012)*, pp. 53–60.
- Lie, K. A., O. Møyner, J. R. Natvig, A. Kozlova, K. Bratvedt, S. Watanabe, and Z. Li (2017). «Successful application of multiscale methods in a real reservoir simulator environment.» In: *Computational Geosciences* 21.5-6, pp. 981–998. ISSN: 15731499.
- Lie, K. A., S. Krogstad, I. S. Ligaarden, J. R. Natvig, H. M. Nilsen, and B. Skaflestad (2012). «Open-source MATLAB implementation of consistent discretisations on complex grids.» In: *Computational Geosciences* 16.2, pp. 297–322. ISSN: 14200597.
- Lienhardt, P. (1994). «N-Dimensional generalized combinatorial maps and cellular quasi-manifolds.» In: *International Journal of Computational Geometry & Applications* 04.03, pp. 275–324.
- Liu, Y., M. F. Goodchild, Q. Guo, Y. Tian, and L. Wu (2008). «Towards a General Field model and its order in GIS.» In: *International Journal of Geographical Information Science* 22.6, pp. 623–643. ISSN: 1365-8816.
- Ljung, P., J. Krüger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman (2016). «State of the Art in Transfer Functions for Direct Volume Rendering.» In: *Computer Graphics Forum* 35.3, pp. 669–691. ISSN: 14678659.
- Löffler, F, A. Müller, and H. Schumann (2011). «Real-time Rendering of Stack-based Terrains.» In: *Vmv*.
- Longley, P. A., M. Goodchild, D. J. Maguire, and D. W. Rhind (2015). *Geographic Information Systems and Science*. 4th. Wiley Publishing. ISBN: 9781118676950.
- Lorensen, W. E. and H. E. Cline (1987). «Marching Cubes: A High Resolution 3D Surface Construction Algorithm.» In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: ACM, pp. 163–169. ISBN: 0-89791-227-6.
- Los Alamos National Laboratory (2016). *Los Alamos Grid Toolbox*.

- Losasso, F. and H. Hoppe (2004). «Geometry clipmaps: terrain rendering using nested regular grids.» In: *ACM Transaction on Graphics* 1.212, pp. 769–776. ISSN: 0730-0301.
- Maciejewski, M., M. Pomianek, and M. Piszczek (2018). «Information potential of the 3D GIS application with the use of virtual technologies.» In: October 2018, p. 58.
- Mallet, J. L. (1992). «GOCAD: A Computer Aided Design Program for Geological Applications.» In: *Three-Dimensional Modeling with Geoscientific Information Systems*. Ed. by A. K. Turner. Dordrecht: Springer Netherlands, pp. 123–141. ISBN: 978-94-011-2556-7.
- (1997). «Discrete modeling for natural objects.» In: *Mathematical Geology* 29.2, pp. 199–219. ISSN: 0882-8121.
- Mallet, J.-L. (2002). *Geomodeling*. ISBN: 0195144600.
- Mantler, S. and S. Jeschke (2006). «Interactive landscape visualization using GPU ray casting.» In: *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and South-east Asia - GRAPHITE '06*, p. 117.
- Mateo Lázaro, J., J. Á. Sánchez Navarro, A. García Gil, and V. Edo Romero (2014). «3D-geological structures with digital elevation models using GPU programming.» In: *Computers & Geosciences* 70, pp. 138–146. ISSN: 00983004.
- Maxelon, M., P. Renard, G. Courrioux, M. Brändli, and N. Mancktelow (2009). «A workflow to facilitate three-dimensional geometrical modelling of complex poly-deformed geological units.» In: *Computers & Geosciences* 35.3, pp. 644–658. ISSN: 00983004.
- McBratney, A. B., M. L. Mendonça Santos, and B. Minasny (2003). *On digital soil mapping*. Vol. 117. 1-2, pp. 3–52. ISBN: 6129351321.
- Mennis, J., R. Viger, and C. D. Tomlin (2005). «Cubic Map Algebra Functions for Spatio-Temporal Analysis.» In: *Cartography and Geographic Information Science* 32.1, pp. 17–32. ISSN: 15230406.
- Ming, J., M. Pan, H. Qu, and Z. Ge (2010). «GSIS: A 3D geological multi-body modeling system from netty cross-sections with topology.» In: *Computers and Geosciences* 36.6, pp. 756–767. ISSN: 00983004.
- Molenaar, M (1992). «A topology for 3D vector maps.» In: *ITC Journal* 1992-1, pp. 25–33. ISSN: 03032434.
- Moore, I. D., R. B. Grayson, and A. R. Ladson (1991). «Digital terrain modelling: A review of hydrological, geomorphological, and biological applications.» In: *Hydrological Processes* 5.1, pp. 3–30.

- Mücke, E. P., I. Saias, and B. Zhu (1999). «Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations.» In: *Computational Geometry: Theory and Applications* 12.1-2, pp. 63–83. ISSN: 09257721.
- Munkberg, J., P. Clarberg, J. Hasselgren, and T. Akenine-Möller (2006). «High Dynamic Range Texture Compression for Graphics Hardware.» In: *ACM Trans. Graph.* 25.3, pp. 698–706. ISSN: 0730-0301.
- Natali, T. G. Klausen, and D. Patel (2014). «Sketch-based modelling and visualization of geological deposition.» In: *Computers and Geosciences* 67, pp. 40–48. ISSN: 00983004.
- Natali, M, E. Lidal, and J Parulek (2012). «Modeling terrains and subsurface geology.» In: *Eurographics 2013-State of the Art Reports*. c, pp. 155–173.
- Nystad, J., A. Lassen, A. Pomianowski, S. Ellis, and T. Olson (2012). «Adaptive Scalable Texture Compression.» In: *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*. Ed. by C. Dachsbacher, J. Munkberg, and J. Pantaleoni. The Eurographics Association. ISBN: 978-3-905674-41-5.
- OGC (2003). *Geographic information — Spatial schema*. Tech. rep. International Standard Organization.
- (2007a). *OpenGIS® Geography Markup Language ( GML ) Encoding Standard*. Tech. rep. Open Geospatial Consortium Inc.
  - (2007b). *The OpenGIS® Abstract Specification*. Tech. rep. Open Geospatial Consortium Inc.
  - (2015). *OGC® Coverage Implementation Schema Contents*. Tech. rep. Open Geospatial Consortium Inc.
- Ortega, L. and A. Rueda (2010). «Parallel drainage network computation on CUDA.» In: *Computers and Geosciences* 36.2, pp. 171–178. ISSN: 00983004.
- Owen, S. J., J. A. Brown, C. D. Ernst, H. Lim, and K. N. Long (2017). «Hexahedral Mesh Generation for Computational Materials Modeling.» In: *Procedia Engineering* 203, pp. 167–179. ISSN: 18777058.
- Patel, D., Ø. Sture, H. Hauser, C. Giertsen, and M. Eduard Gröller (2009). «Knowledge-assisted visualization of seismic data.» In: *Computers and Graphics (Pergamon)* 33.5, pp. 585–596. ISSN: 00978493.
- Pellerin, J., G. Caumon, C. Julio, P. Mejia-Herrera, and A. Botella (2015). «Elements for measuring the complexity of 3D structural models: Connectivity and geometry.» In: *Computers & Geosciences* 76, pp. 130–140. ISSN: 00983004.
- Pellerin, J., A. Botella, F. Bonneau, A. Mazuyer, B. Chauvin, B. Lévy, and G. Caumon (2017). «RINGMesh: A programming library for developing

- mesh-based geomodeling applications.» In: *Computers and Geosciences* 104, October 2016, pp. 93–100. ISSN: 00983004.
- Penninga, F. and P. J. M. Van Oosterom (2008). «A simplicial complex-based DBMS approach to 3D topographic data modelling.» en. In: *International Journal of Geographical Information Science* 22.7, pp. 751–779. ISSN: 1365-8816.
- Penninga, F. (2008). «A Simplicial Complex-based Solution in a Spatial DBMS.» PhD thesis, p. 193.
- Petrasova, A., B. Harmon, V. Petras, and H. Mitasova (2015). *Tangible Modeling with Open Source GIS*. 1st. Springer Publishing Company, Incorporated. ISBN: 3319257730, 9783319257730.
- Peytavie, A., E. Galin, J. Grosjean, and S. Merillou (2009). «Arches: A framework for modeling complex terrains.» In: *Computer Graphics Forum* 28.2, pp. 457–467. ISSN: 01677055.
- Pilouk, M. (1996). «Integrated modelling for 3D GIS.» PhD thesis. ITC, The Netherlands.
- Pinet, F. (2012). «Entity-relationship and object-oriented formalisms for modeling spatial environmental data.» In: *Environmental Modelling & Software* 33, pp. 80–91. ISSN: 13648152.
- Portele, C. (2018). *Mapping UML to GML Application Schemas: ShapeChange - Architecture and Description (version 2.5.0)*.
- Poupeau, B. and O. Bonin (2006). «Cristage: A 3D GIS with A Logical Crystallographic Layer To Enable Complex Analyses.» In: *Innovations in 3D Geo Information Systems*. Ed. by A. Abdul-Rahman, S. Zlatanova, and V. Coors. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 225–234. ISBN: 978-3-540-36998-1.
- Pryet, A., J. Ramm, J. P. Chilès, E. Auken, B. Deffontaines, and S. Violette (2011). «3D resistivity gridding of large AEM datasets: A step toward enhanced geological interpretation.» In: *Journal of Applied Geophysics* 75.2, pp. 277–283. ISSN: 09269851.
- Ritter, G. X., J. N. Wilson, and J. L. Davidson (1990). «Image algebra: An overview.» In: *Computer Vision, Graphics and Image Processing* 49.3, pp. 297–331. ISSN: 0734189X.
- Rocha, A., R. C. Mota, H. Hamdi, U. R. Alim, and M. Costa Sousa (2018). «Illustrative Multivariate Visualization for Geological Modelling.» In: *Computer Graphics Forum* 37.3, pp. 465–477. ISSN: 14678659.
- Roettger, S., S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser (2003). «Smart Hardware-accelerated Volume Rendering.» In: *Proceedings of the Sym-*

- posium on Data Visualisation 2003*. VISSYM '03. Grenoble, France: Eurographics Association, pp. 231–238. ISBN: 1-58113-698-6.
- Rueda, A., J. M. Noguera, and C. Martínez-Cruz (2013). «A flooding algorithm for extracting drainage networks from unprocessed digital elevation models.» In: *Computers and Geosciences* 59, pp. 116–123. ISSN: 00983004.
- S. Høyer, a., F. Jørgensen, N. Foged, X. He, and a.V. Christiansen (2015). «Three-dimensional geological modelling of AEM resistivity data – A comparison of three methods.» In: *Journal of Applied Geophysics* 115, pp. 65–78. ISSN: 09269851.
- Schmitz, O., D. Karssenberg, K. de Jong, J. L. de Kok, and S. M. de Jong (2013). «Map algebra and model algebra for integrated model building.» In: *Environmental Modelling & Software* 48, pp. 113–128. ISSN: 13648152.
- Sellers, G., R. Wright Jr, and N. Haemel (2016). *OpenGL SuperBible. Comprehensive Tutorial and Reference*. 7th ed. Addison Wesley.
- Sentís, M. L. and C. W. Gable (2017). «Coupling LaGrit unstructured mesh generation and model setup with TOUGH2 flow and transport: A case study.» In: *Computers and Geosciences* 108, June 2016, pp. 42–49. ISSN: 00983004.
- She, J., Y. Zhou, X. Tan, X. Li, and X. Guo (2017). «A parallelized screen-based method for rendering polylines and polygons on terrain surfaces.» In: *Computers and Geosciences* 99, January 2016, pp. 19–27. ISSN: 0098-3004.
- Shen, D., D. W. Wong, F. Camelli, and Y. Liu (2013). «An ArcScene plugin for volumetric data conversion, modeling and spatial analysis.» In: *Computers & Geosciences* 61, pp. 104–115. ISSN: 00983004.
- Sigg, C., T. Weyrich, M. Botsch, and M. Gross (2006). «GPU-Based Ray-Casting of Quadratic Surfaces.» In: *Symposium on Point-Based Graphics*, pp. 59–65.
- Smirnov, A., E. Boisvert, and S. J. Paradis (2008). «Support vector machine for 3D modelling from sparse geological information of various origins.» In: *Computers & Geosciences* 34, pp. 127–143.
- Sobhanpanah, C. (1989). «Extension of a boundary representation technique for the description of N dimensional polytopes.» In: *Computers and Graphics* 13.1, pp. 17–23. ISSN: 00978493.
- Song, R., X. Qin, Y. Tao, X. Wang, B. Yin, Y. Wang, and W. Li (2019). «A semi-automatic method for 3D modeling and visualizing complex geological bodies.» In: *Bulletin of Engineering Geology and the Environment* 78.3, pp. 1371–1383. ISSN: 14359529.

- Spear, A. D., J. D. Hochhalter, A. R. Cerrone, S. F. Li, J. F. Lind, R. M. Suter, and A. R. Ingraffea (2016). «A method to generate conformal finite-element meshes from 3D measurements of microstructurally small fatigue-crack propagation.» In: *Fatigue and Fracture of Engineering Materials and Structures* 39.6, pp. 737–751. ISSN: 14602695.
- Št'ava, O., B. Beneš, M. Brisbin, and J. Křivánek (2008). «Interactive terrain modeling using hydraulic erosion.» In: *EuroGraphics Symposium on Computer Animation*, pp. 201–210.
- Sutherland, I. E. (1963). «Sketchpad: A Man-Machine Graphical Communication System.» PhD thesis. Massachusetts Institute of Technology, Lincoln Lab.
- Takeyama, M. (1997). «Building spatial models within GIS through Geo-Algebra.» In: *Transactions in GIS* 2.3, pp. 245–256. ISSN: 1361-1682.
- Takeyama, M. and H. Couclelis (1997). «Map dynamics: integrating cellular automata and GIS through Geo-Algebra.» In: *International Journal of Geographical Information Science* 11.1, pp. 73–91. ISSN: 1365-8816.
- Tegtmeier, W., S. Zlatanova, P. van Oosterom, and H. Hack (2014). «3D-GEM: Geo-technical extension towards an integrated 3D information model for infrastructural development.» In: *Computers & Geosciences* 64, pp. 126–135. ISSN: 00983004.
- Tegtmeier, W., P. van Oosterom, S. Zlatanova, and R. Hack (2009). «Information management in civil engineering infrastructural development: with focus on geological and geotechnical information.» In: *Proceedings of the ISPRS workshop GeoWeb 2009 Academic Track Cityscapes XXXVIII-3-*, pp. 68–73.
- Terrington, R. L., S. J. Mathers, H Kessler, V Hulland, and S. J. Price (2009). *Subsurface Viewer 2009: User Manual*. Tech. rep. Geological Modelling Systems Team. British Geological Survey.
- Tevs, A., I. Ihrke, and H.-P. Seidel (2008). «Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering.» In: *Symposium on Interactive 3D Graphics and Games (i3D'08)*, pp. 183–190.
- The CGAL Project (2019). *CGAL User and Reference Manual*. 4.14. CGAL Editorial Board.
- Thiele, S. T., M. W. Jessell, M. Lindsay, V. Ogarko, J. F. Wellmann, and E. Pakyuz-Charrier (2016). «The topology of geology 1: Topological analysis.» In: *Journal of Structural Geology* 91, pp. 27–38. ISSN: 01918141.
- Tomlin, C. D. (1990). *Geographic information systems and cartographic modeling*. Prentice Hall series in geographic information science. Prentice Hall. ISBN: 9780133509274.



- Treib, M., F. Reichl, S. Auer, and R. Westermann (2012). «Interactive editing of GigaSample terrain fields.» In: *Computer Graphics Forum* 31.2, pp. 383–392. ISSN: 01677055.
- Turner, A. K. (1992). *Three-Dimensional Modeling with Geoscientific Information Systems*. C]: [Nato ASI series. Springer Netherlands. ISBN: 9780792315506.
- (2006). «Challenges and trends for geological modelling and visualisation.» In: *Bulletin of Engineering Geology and the Environment* 65.2, pp. 109–127. ISSN: 1435-9529.
- Voudouris, V. (2010). «Towards a unifying formalisation of geographic representation: the object–field model with uncertainty and semantics.» In: *International Journal of Geographical Information Science* 24.12, pp. 1811–1828. ISSN: 1365-8816.
- Wang, C., T. R. Wan, and I. J. Palmer (2010). «Urban flood risk analysis for determining optimal flood protection levels based on digital terrain model and flood spreading model.» In: *The Visual Computer* 26.11, pp. 1369–1381. ISSN: 0178-2789.
- Wang, J., M. Duckham, and M. Worboys (2015). «A framework for models of movement in geographic space.» In: *International Journal of Geographical Information Science* 8816.May, pp. 1–23. ISSN: 1365-8816.
- Wang, J., F. Yang, and Y. Cao (2014). «Cache-aware sampling strategies for texture-based ray casting on GPU.» In: *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on*, pp. 19–26.
- Wang, Z., H. Qu, Z. Wu, H. Yang, and Q. Du (2016). «Formal representation of 3D structural geological models.» In: *Computers & Geosciences* 90, pp. 10–23. ISSN: 00983004.
- Wang, Z., H. Qu, Z. Wu, and X. Wang (2018). «Geo3DML: A standard-based exchange format for 3D geological models.» In: *Computers and Geosciences* 110.August 2017, pp. 54–64. ISSN: 00983004.
- Weiss, K., L. De Floriani, R. Fellegara, and M. Velloso (2011). «The PR-star octree: a spatio-topological data structure for tetrahedral meshes.» In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11*. New York, New York, USA: ACM Press, p. 92. ISBN: 9781450310314.
- Wilson, J. and A. Fotheringham (2008). *The handbook of geographic information science*. Blackwell companions to geography. Blackwell Pub. ISBN: 9781405107952.
- Worboys, M. and M. Duckham (2004). *GIS: A Computing Perspective, Second Edition*. Taylor & Francis. ISBN: 9780415283755.

- Wu, L. (2004). «Topological relations embodied in a generalized tri-prism (GTP) model for a 3D geoscience modeling system.» In: *Computers & Geosciences* 30.4, pp. 405–418. ISSN: 00983004.
- Wycisk, P., T. Hubert, W. Gossel, and C. Neumann (2009). «High-resolution 3D spatial modelling of complex geological structures for an environmental risk assessment of abundant mining and industrial megasites.» In: *Computers and Geosciences* 35.1, pp. 165–182. ISSN: 00983004.
- Yagel, R., D. Cohen, and A. Kaufman (1992). «Normal estimation in 3D discrete space.» In: *The Visual Computer* 8.5-6, pp. 278–291. ISSN: 01782789.
- Zanchi, A., S. Francesca, Z. Stefano, S. Simone, and G. Graziano (2009). «3D reconstruction of complex geological bodies: Examples from the Alps.» In: *Computers & Geosciences* 35.1, pp. 49–69. ISSN: 00983004.
- Zehner, B., N. Watanabe, and O. Kolditz (2010). «Visualization of gridded scalar data with uncertainty in geosciences.» In: *Computers and Geosciences* 36.10, pp. 1268–1275. ISSN: 00983004.
- Zehner, B., O. Hellwig, M. Linke, I. Görz, and S. Buske (2016). «Rasterizing geological models for parallel finite difference simulation using seismic simulation as an example.» In: *Computers and Geosciences* 86, pp. 83–91. ISSN: 00983004.
- Zeiler, M. (1999). *Modeling Our World: The ESRI Guide to Geodatabase Design*. ESRI Press. ISBN: 9781879102620.
- Zhao, Y., A. Padmanabhan, and S. Wang (2013). «A parallel computing approach to viewshed analysis of large terrain data using graphics processing units.» In: *International Journal of Geographical Information Science* 27.2, pp. 363–384. ISSN: 1365-8816.
- Zhu, R., P. C. Kyriakidis, and K. Janowicz (2017). «Beyond Pairs: Generalizing the Geo-dipole for Quantifying Spatial Patterns in Geographic Fields.» In: *Societal Geo-innovation: Selected papers of the 20th AGILE conference on Geographic Information Science*. Ed. by A. Bregt, T. Sarjakoski, R. van Lammeren, and F. Rip. Springer International Publishing, pp. 331–348. ISBN: 978-3-319-56759-4.
- Zlatanova, S. (2000). «3D GIS for urban development.» PhD thesis. ITC, The Netherlands, p. 222.



This document was typeset using the typographical look-and-feel classicthesis developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst’s seminal book on typography “*The Elements of Typographic Style*”. classicthesis is available for both  $\LaTeX$  and  $\text{LyX}$ : <https://bitbucket.org/amiede/classicthesis/>