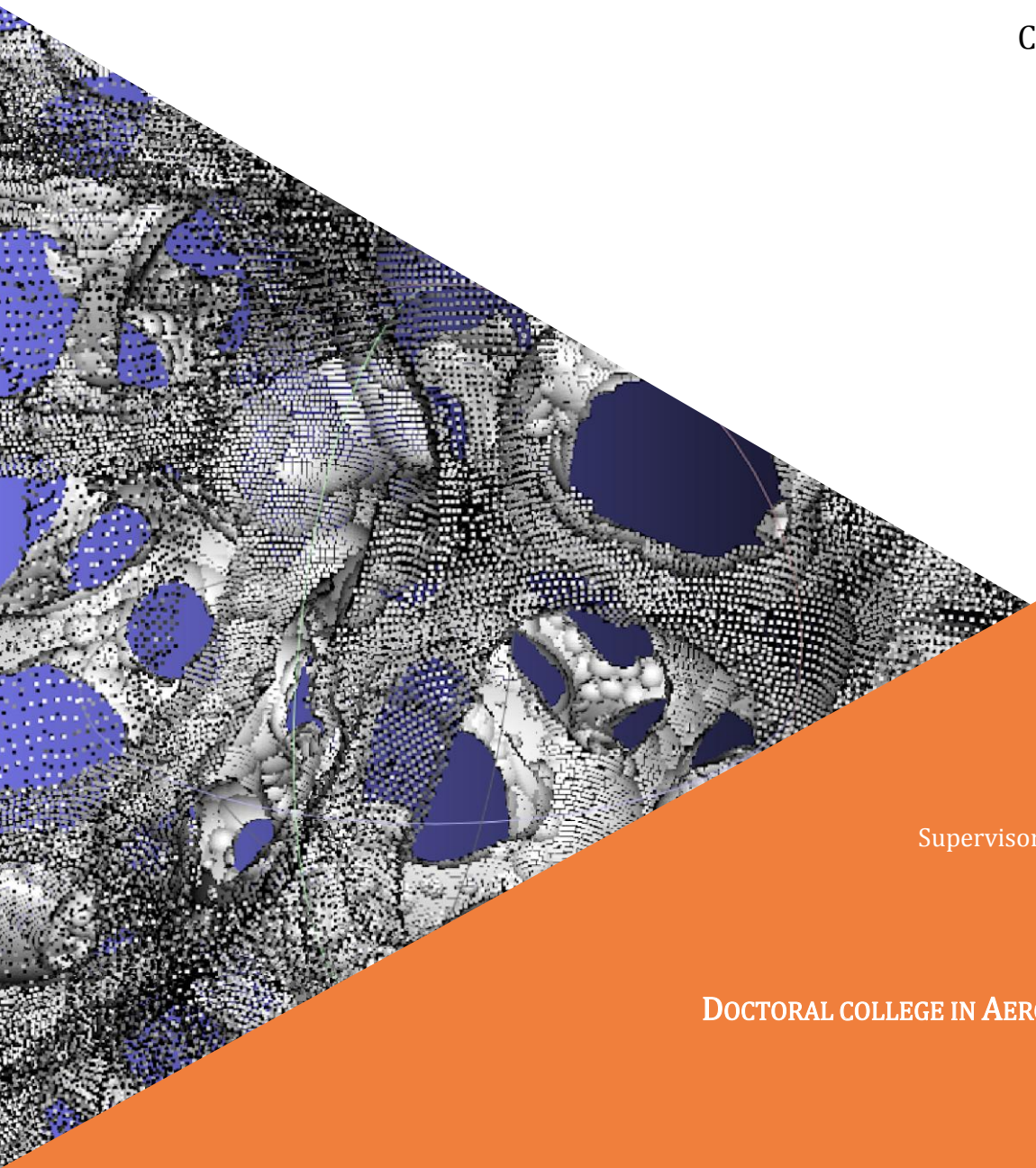
LIÈGE université
Sciences Appliquées

Finite element computations on foam geometry reconstructed from tomographic images

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosphy (Phd) in Engineering Science

by

Christophe LEBLANC



Supervisor: Eric BECHET

DOCTORAL COLLEGE IN AEROSPACE AND MECHANICS

FEBRUARY 2024

Members of the Examination Committee

Supervisor:
Prof. Eric Béchet
Université de Liège (Liège, Belgium)

Members:
Prof. Pierre Duysinx
Université de Liège (Liège, Belgium)

Prof. Christophe Geuzaine
Université de Liège (Liège, Belgium)

Prof. Ludovic Noels
Université de Liège (Liège, Belgium)

Prof. Thierry Massart
Université Libre de Bruxelles (Bruxelles, Belgium)

Prof. Sébastien Mercier
Université de Lorraine (Metz, France)

"Poets say sciences takes away from the beauty of the stars... but far more marvellous is the truth than any artist imagined it."

Richard Feynman

"I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone."

Bjarne Stroustrup

Abstract

Christophe LEBLANC

Finite element computations on foam geometry reconstructed from tomographic images

The present thesis introduces an automated procedure for constructing geometrical representative volume elements of cellular materials from computerised tomography, with an emphasis on open foams. The final aim of this procedure is to generate meshable geometries that can be used in some finite element method in order to analyse their mechanical behaviour. The methodology consists in growing and fitting a set of ellipsoids to each cell foam. These ellipsoids are seeded by local maxima of the distance to the struts/walls obtained from computerised tomography images. Then, auxiliary ellipsoids are constructed in order to better fit local struts' geometries. From ellipsoids' surfaces, a set of "exterior" points and associated normals is extracted. Finally, from this set, a geometry is obtained using a Poisson surface reconstruction. This methodology is thus fully voxel-based and does not depends on any assumption about the statistical distributions of the foam cells. Therefore, it is able to reproduce an accurate geometric model of the foams' microstructure and its possible irregularities. Moreover, this procedure allows the processing of large 3D data sets that do not fit the random access memory by slicing it into smaller independent chunks. The minimal thickness of each chunk is only limited by the maximum cell size lying inside this chunk. The effectiveness of the proposed approach is illustrated by comparing it to finite element simulations with experimental results of uniaxial compressions of an open foam.

Abstract

Christophe LEBLANC

Finite element computations on foam geometry reconstructed from tomographic images

Cette thèse présente une procédure automatisée pour construire des volumes représentatifs à partir d'images tomographiques de matériaux cellulaires, en considérant tout particulièrement le cas des mousses. L'objectif final de cette procédure est de produire des géométries qui peuvent être maillées à des fins d'utilisations dans des solveurs éléments finis; avec l'objectif d'analyser les comportements mécaniques de mousses ouvertes. La méthode proposée consiste à dilater et ajuster un ensemble d'ellipsoïdes à chaque cellule de la mousse considérée. Ces ellipsoïdes sont initialement déterminés en calculant les maxima locaux du champ des distances aux entretoises/parois des cellules obtenues à partir d'images tomographiques. Ensuite, des ellipsoïdes auxiliaires sont construits afin de mieux coller à la géométrie de la mousse. A partir des surfaces des ellipsoïdes est extrait un ensemble de points "extérieurs" associés à des normales locales. Finalement, à partir de cet ensemble de points et normales associées, une représentation de la surface de la mousse est obtenue au moyen de l'algorithme de reconstruction de surface de Poisson. La procédure proposée dépend donc entièrement des voxels de l'image tomographique et est indépendante de toute distribution statistique que pourraient adopter les cellules de la mousse. En conséquence, cette procédure est capable de reproduire de manière précise un modèle géométrique fidèle de la microstructure d'une mousse et ses éventuelles irrégularités. De plus, cette procédure est capable de traiter de grandes quantités de données qui dépasseraient la RAM disponible en découpant celles-ci en tranches indépendantes plus petites. L'épaisseur minimale de chaque tranche est uniquement limitée par la taille maximale des cellules occupant cette tranche. L'efficacité de cette méthode est illustrée en comparant un modèle éléments finis utilisant un maillage généré à partir d'une géométrie obtenue par cette procédure, à des résultats expérimentaux d'une mousse d'aluminium ouverte soumise à une compression uniaxiale.

Acknowledgements

I would like to take here the opportunity to express my gratitude to everyone who has supported me during my studies and research:

Firstly, I would like to thank Prof. Eric Béchet for providing me the opportunity to conduct a doctoral research under his advised guidance. His countless suggestions and comments all along his supervision have been priceless. The opportunity he offered me to teach his students has been an enriching experience.

I am grateful to the members of the thesis committee, Ludovic Noels, Pierre Duysinx and Christophe Geuzaine, as well as Dr. Anne Jung and Prof. Thierry Massart, for their time spending on reviewing and commenting my works, and for theirs help in submitting my research results to the community.

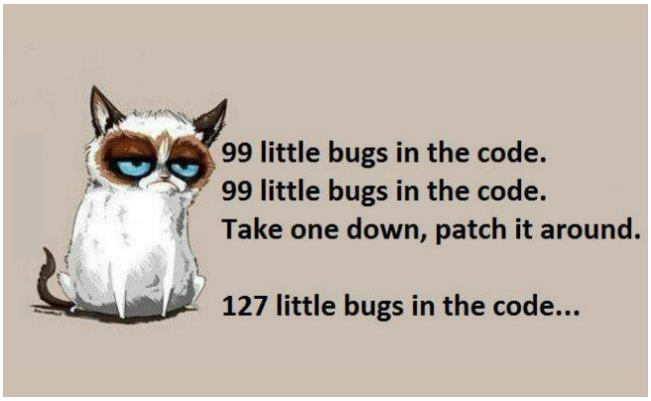
I would like to thank the University of Liège for providing an awesome platform and workplace for enabling me to conduct all my works and duties. I would like also to express my gratitude to the "little personnel", such as the cleaning ladies and waitresses, whose daily work is underappreciated, though essential to everybody.

I gratefully acknowledge the funding by the *Actions de recherche concertées ARC 09/14-02 BRIDGING* and granted by the *Académie Universitaire Wallonie-Europe* and by the *F.R.S-F.N.R.S. Enlighten It project*, grant number PDR T.0038.16.

I would to like to thanks my parents, Liana and Michel for their long-life help, support and encouragement along all my studies. I will literally not be here without them. I also thank my brother Gabriel and my sister-in-law Liuba.

I thank my cat Pilou for reminding me to take some rest during those months of quarantine by sitting on my keyboard and leaving cryptic messages on my screen, such as "hjuojdavubaj".

Finally, I thank the compiler g++ for sometimes entertaining my boring coding sessions with its verbose and difficult to read and understand debugging messages.



Contents

List of Figures

List of Tables

List of Abbreviations

RGB	**R**ed, **G**reen, **B**lue
FIFO	**F**irst **I**n, **F**irst **O**ut
RVE	**R**epresentative **V**olume **E**lement
CT	**C**omputerised **T**omography
FEM	**F**inite **E**element **M**ethod
XFEM	e**X**tended **F**inite **E**lement **M**ethod
RAM	**R**andom **A**ccess **M**emory
SEM	**S**canning **E**lectron **M**icroscopy
DN-RSA	**D**istance **N**eighbour based **R**andom **S**equential **A**ddition
B-REP	**B**oundary **REP**resentation
ITK	**I**nsight **T**ool**Kit**
OTB	**O**rfeo **T**ool**B**ox
DT	**D**istance **T**ransform
FP	**F**eature **P**ixel
CFP	**C**losest **F**eature **P**ixel
KKT	**K**arush-**K**uhn-**T**ucker
GDBSCAN	**G**eneralised **D**ensity **B**ased **SCAN**
SVD	**S**ingular **V**alue **D**ecomposition
MVCE	**M**inimum **V**olume **C**overing **E**llipsoid
MVCEE	**M**inimum **V**olume **C**overing **E**llipsoid of **E**llipsoids
BVP	**B**oundary **V**alue **P**roblems

List of Symbols

$\emptyset$	empty set.
$A \subset B$	set A is a subset of set B.
$A \subseteq B$	set A is a subset of or equal to set B.
$A \supset B$	set A includes set B as a subset.
$A \supseteq B$	set A includes set B and may be equal to set B.
$\cup_i B_i$	union of sets B_i.
$\cap_i B_i$	intersection of sets B_i.
$A \setminus B$	set A minus set B.
∂A	boundary of set A.
$intA$	interior of set A.
$a \in A$	element a belongs to set A.
$\forall a$	for all a...
$\exists a$	it exists a...
s.t.	such that.
resp.	respectively.
$Vol(A)$	volume of (compact) set A.
sup A	supremum of (ordered) set A, i.e. minimum of majorant set of A.
inf A	infininum of (ordered) set A, i.e. maximum of minorant set of A.
$\mathcal{N}$	set of positive integers (including zero).
$\mathcal{Z}$	set of integers.
$\mathcal{R}$	set of real numbers.
$\mathcal{R}_+$	set of positive real numbers (including zero).
$\mathcal{R}_-$	set of negative real numbers (including zero).
$\mathcal{R}^n$	abbreviation for $\mathcal{R}^{n\times 1}$ n–dimensional vector space on $\mathcal{R}$.
$\|a\|$	absolute value of number a.
Bold letters such as $\mathbf{Q}$ or $\mathbf{v}$	column vectors in $\mathcal{R}^n$.
v_i	i–th component of vector $\mathbf{v}$.
$\|\mathbf{v}\|$	Euclidean norm of vector $\mathbf{v}$.
$< \mathbf{u}\|\mathbf{v} >$ or $\mathbf{u} \,.\, \mathbf{v}$	Euclidean dot product of vectors $\mathbf{u}$ and $\mathbf{v}$.
$\mathbf{u} \wedge \mathbf{v}$	cross product of vectors $\mathbf{u}, \mathbf{v} \in \mathcal{R}^3$.
$\hat{\mathbf{v}}$	normalized vector $\mathbf{v}$, i.e. $\|\mathbf{v}\| = 1$.
$\mathbf{v}^t, M^t$	transpose of vector $\mathbf{v}$ and matrix M.
I	identity matrix.
M^{-1}	inverse of matrix M.
$M \succ 0$, $M \succeq 0$	matrix M is positive definite, resp. positive semi–definite.
$M \succ A$, $M \succeq A$	matrix $M - A$ is positive definite, resp. positive semi–definite.
$M \prec 0$, $M \preceq 0$	matrix M is negative definite, resp. negative semi–definite.
$M \prec A$, $M \preceq A$	matrix $M - A$ is negative definite, resp. negative semi–definite.
$\mathcal{E}$	ellipsoid.
δ_{ij}	Kronecker delta.
$\frac{\partial}{\partial x_i}$	partial derivative along component x_i.
$\vec{\nabla} = \frac{\partial}{\partial \vec{x}}$, $\Delta = \vec{\nabla} \,.\, \vec{\nabla}$	gradient and Laplacian.

Dedicated to my parents,
Liana Raţiu and Michel Leblanc.

Foreword

This thesis is divided in three main parts: an *image analysis* (chapters 2 and 3), *geometry reconstruction* (chapter 4), and *finite element simulations* (chapter 5). Many of these parts require some knowledge in different fields that the reader may not be familiar with. In order to ease the reading of this thesis, some used concepts that may be unfamiliar to the reader are further explained in the appendices. The appendices contain explanations which have been rewritten, condensed and adapted with homogenised notations from external sources by the author of this thesis. As such, the contents of the appendices are already existing works produced by other authors. They are mentioned in this thesis for the sake of completeness and to avoid the interested reader to have to search for external sources and algorithms each have their own sets of notations.

Chapter 1

Introduction, state of the art, and summary of contributions

Nowadays, cellular materials represent promising approach for obtaining simultaneously strength, stiffness, dissipation, and weight reduction, and are increasingly used in numerous engineering applications. Among other things, they are used as insulators, filters and crash absorbers [56, 88], see Figures 1.1 and 1.2. In particular, the class of open–foam materials has found countless applications in many domains such as energy absorbers [71], vibration dampening [186] (see Figure 1.3), hydrogen production [174] and aerospace [40], for instance. As a matter of fact, cellular materials also exist in nature such as in bones [41] (see Figure 1.4) or wood.

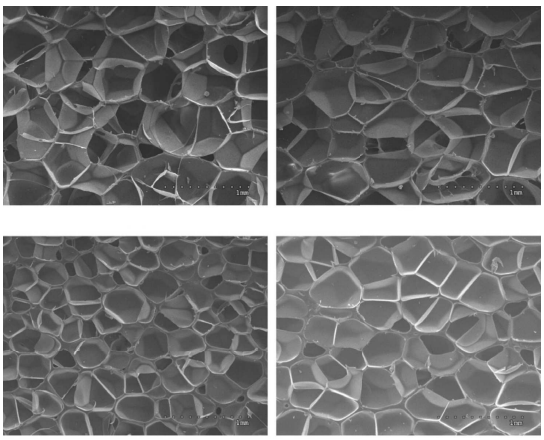


FIGURE 1.1: Scanning electron microscopy images of a closed foam [100].

1.1 Foam manufacturing

Manufacturing of foams can be achieved via several processes. For instance, one approach for producing metallic open–cell foams is to turn to the technique of electro–deposition onto a sacrificial polymer foam with open–cells. The polymer phase is then etched away, resulting in an open foam with hollow struts [12] (see Figure 1.5). Foams can also be manufactured by direct blowing agent injection (such as CO_2 gas) into melt, the viscosity of which is controlled by temperature and by adding ceramics powder (see Figure 1.6). This technique is extensively used to obtain aluminium alloy foams. For an extensive survey of metallic foam manufacturing techniques, the interested reader may refer to Reference [154].

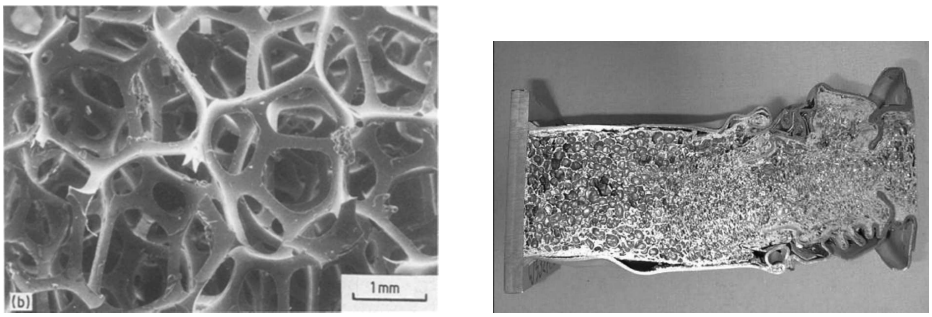


FIGURE 1.2: Scanning electron microscopy image of an open foam [65], and Longitudinal crash absorber with cellular internal structure [172].

FIGURE 1.3: From right to left: silicon rubber specimen, silicon rubber/aluminum foam composite, aluminum foam only, room-temperature–vulcanising silicone/aluminum foam composite, room-temperature-vulcanising silicone only [186].

The microstructure of foams is complex and consists in an interconnected network of ligaments located along the edges of random packed cells. Cells can be open and closed or partially closed by walls. Foam's struts can themselves adopt various cross sections, from sharp edged cross sections, to smooth ones and can be hollow [74, 192] (see Figure 1.7). In regards of their manufacturing processes, foams share structural similarities with equilibrated liquid foams, such as dry foam and soap froth, where the cells originated from bubbles separated by tensioned liquid films adopt the shapes of near-polyhedral volumes. Under conditions of mechanical equilibrium where the surface free energy is minimised, these latter foams obey locally Plateau's laws [132, 87]. Under these conditions, each face adopts a constant mean curvature and each face meets at each cell edge at equal dihedral angles of 120°; while each vertex is adjacent to four joining edges ate equal tetrahedral angles of $\cos^{-1}(-1/3) = 109.47°$.

1.2 Microstructure representations

In the current context, there is a need to develop foam models able to predict the homogenised or apparent behaviour based on the microstructural features. The improvements of computer tomography during the last decade made the direct observation of the foam internal microstructure easier [157], and this approach has been used extensively for building geometrical foam models [57, 187, 193, 117]. Though very detailed, images obtained from computer tomography need to be pre-processed in order to allow their use as a geometrical basis in FEM models. Generally, the pre-processing involves, among other steps depending on the acquisition process and quality of the obtained images, a watershed and, optionally, a H-maxima transform [57, 71, 138, 73], both of which can be computationally expensive.

FIGURE 1.4: Variability in foam–like trabecular bone. On the left is the trabecular bone of the bighorn sheep's horn core and on the right is the trabecular bone in the proximal femur of a black bear [41].

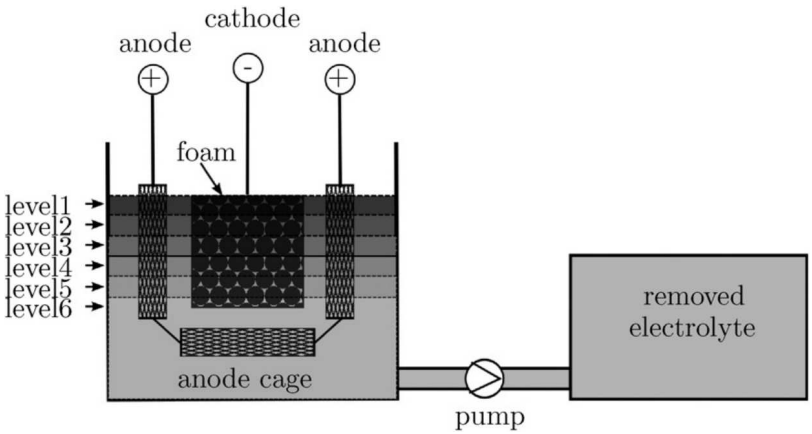


FIGURE 1.5: Schematic drawing of the electro–deposition process to produce gradually coated foams. Outlined are six different electrolyte levels, which occur successively over the plating time. [71].

Some efforts have been made to asses this difficulty, by either designing new algorithms with improved complexity [176, 155, 159], or by parallelising them [21, 82]. Notwithstanding, computing a watershed still remains to this day a memory expensive algorithm for large 3D images.

1.2.1 Direct discretisation

One strategy to avoid these computationally expensive steps is to resort to direct image discretisation [3, 193] (see Figure 1.8). However, obtaining FEM models from discretised images [173, 48] remains a daunting task since characterising large data samples requires both time and computational capacity, as well as frequent user input; currently restricting this method to smaller samples [113]. An interesting strategy, though, has been proposed in Reference [91]. It consists in coupling extended finite element methods (XFEM) with level sets on non-conforming meshes. The use of non-conforming meshes allows to alleviate near zero or negative Jacobian mesh element issues. Indeed, the complex geometry of the microstructure is implicitly encoded by the level sets. Elements of the non-conforming mesh intersecting the level sets are then locally enriched within the XFEM framework in order to take into account the presence of physical interfaces.

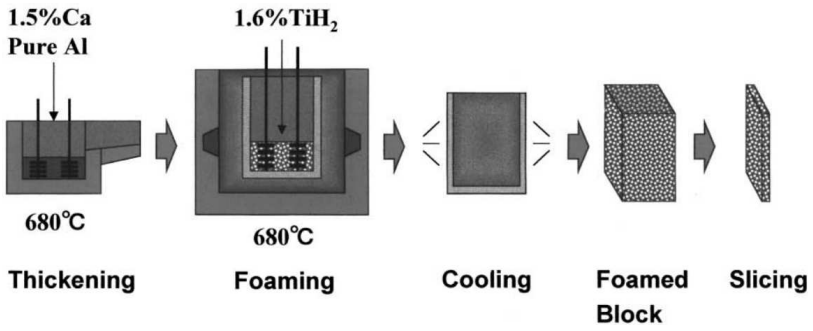


FIGURE 1.6: Direct foaming of melts with blowing agents ("Alporas"–process) [12].

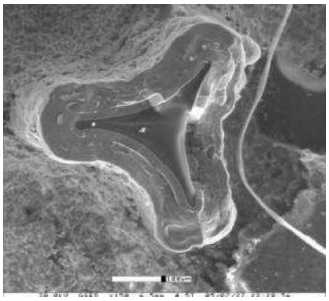


FIGURE 1.7: SEM image of the cross section of a typical strut [74].

1.2.2 Idealised geometries

Another possibility is to turn to idealised microstructural geometries. One advantage of this approach is to allow exploring the basic mechanism linking the microstructure of the considered material to its mechanical behaviour. A simple model using rectangular prisms, as proposed in Reference [56], can already provide some insights. However, a better idealised microstructure has been proposed by Kelvin [163] whose model tessellates space with a minimum partitional area using tetrakaidecahedra cells (see Figure 1.9). Kelvin's model has been later improved by the Weaire-Phelan foam model [181] (see Figure 1.10). As these models show a highly periodic geometry, they are unable to capture the randomness of real microstructures, and, consequently, are poorly effective for capturing the mechanical properties of real foams.

In order to tackle this randomness issue, various strategies are proposed in the literature (see, e.g., References [61, 27, 191] to mention a few). A straightforward strategy is to start from a periodic cell structure and add some imperfections to it. This method has been studied in Reference [177], leading to a better agreement with experimental data for the elastic response and the plateau stage of the strain-stress curve for a foam sample undergoing uniaxial compression. A common strategy consists in using Voronoï tessellations that are based on the distribution of nuclei [166]; and their generalisations, especially the Laguerre tessellations [89, 88, 95] (see Figure 1.11). The Laguerre tessellation is a form of weighted Voronoï tessellation in which the space is partitioned using (possibly) random packings of spheres of different radii instead of pointwise nuclei. Though versatile and leading to more realistic geometric representation of foam's microstructures, Laguerre tessellations are

FIGURE 1.8: Voxellisation and segmentation of a closed foam sample obtained from a 3D CT-scan. [91].

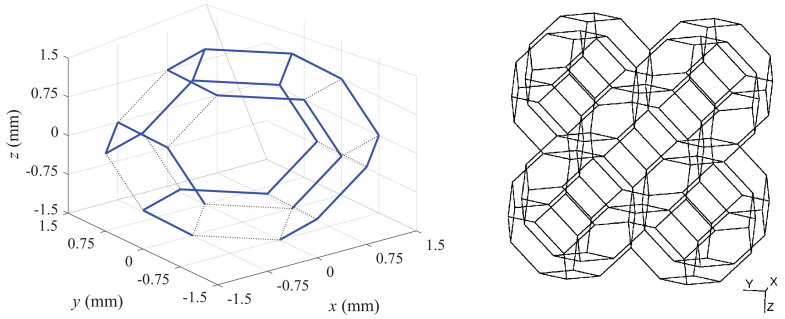


FIGURE 1.9: (a) Single Kelvin cell, and (b) 2×2 Kelvin cell array [105].

not of straightforward use. Indeed, the question arise of how to randomly pack the spheres (loosely or densely), following which distribution (normal, log-normal, gamma, Poisson…) and how to choose the sphere radii. The link between these model parameters and geometric characteristics of the foams (such as the porosity, the average pore size and the pore size variation) is also far from being trivial and has been subject of numerous studies [136, 135, 169, 96, 107, 106]. However, Laguerre tessellation are not able, by construction, to capture non-convex cells, cells with curved boundaries, and suffer from inconsistency of presenting seedless cells [90] (see Figure 1.12). Also, when available, a Laguerre tessellation does not fully exploit the voxel informations present in CT-images. These issues have motivated the development of more generalised models exploiting voxel informations and/or generalised metrics such as in References [22, 148, 4, 157, 150, 151, 25], and the present thesis.

Another approach for trying to generate realistic foam geometries consists in randomly packing spheres. Numerous packing algorithms can be found in the literature and can be classified under collective rearrangement, sequential addition, or sedimentation methods [19]. For instance, the random sequential addition consists in adding one by one spheres with randomly chosen diameters under the action of some "gravitational" field. In order to obtain a packing as tight as possible, simulated annealing can be used subsequently for randomly perturbing the locations of the spheres and improve the tightness of the packing. An alternative option is to

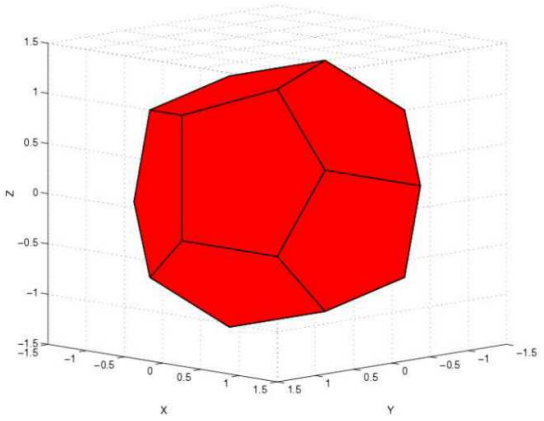


FIGURE 1.10: Dodecahedral cell of the Weaire-Phelan foam [149].

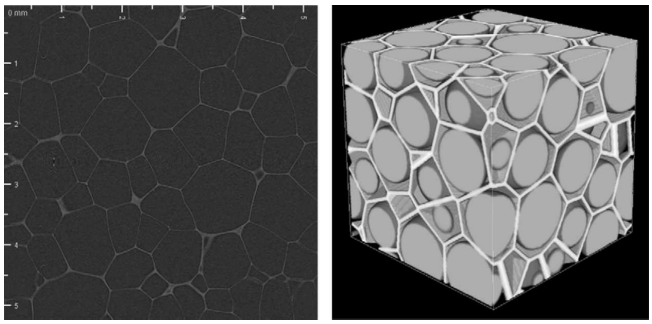


FIGURE 1.11: Example of a Laguerre tessellation of a foam. Left: Section of a tomographic image of a polymer foam (pixel size: 10.21 μm, Source: Fraunhofer IWM). Right: Visualisation of a Laguerre tessellation generated by a dense packing of spheres. The spheres are inscribed in their cells [135].

start with a very dense configuration of large overlapping spheres and then reduce stepwise their diameters and location in order to reduce the overlapping [20].

However, in order to reproduce accurately the morphological parameters of the considered foam, some minimisation criterion has to be designed; which is a task far from being trivial. To this aim, strategies using surface minimisation principles by evolving tessellations thanks to the Surface Evolver Software have been studied [86, 85, 170]. Moreover, in the case of poly-dispersed spheres, some statistical distribution for the sphere diameters must be chosen (e.g., Poisson, log-normal, gamma, or Gaussian) [87]. Finally, from these algorithms it is only possible to reconstruct convex cells and tessellation of the obtained packing only provides infinitely thin struts/walls that need to be "dressed". Despite all these issues and difficulties, random packing algorithms have been extensively used in conjunction with Voronoï tessellations [114], as well as Laguerre tessellations [137, 184].

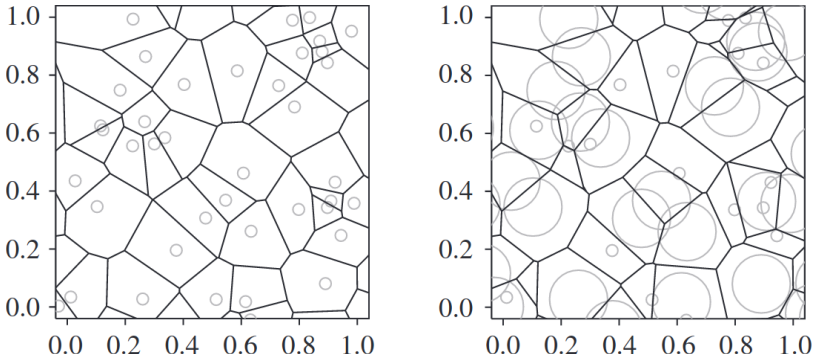


FIGURE 1.12: A Poisson–Voronoï tessellation in $\mathcal{R}^2$ (left) and the Laguerre tessellation of the same set of points with radii chosen from a two-atom distribution (right). Note how the cell geometry is altered by the introduction of weights, that some Laguerre cells are empty (around position $(0.3, 0.6)$ or $(0.9, 0.3)$), and that some cells do not contain the nucleus (for instance, the almost triangular cell at $(0.1, 0.1)$) [90].

1.2.3 "Dressed" idealised geometries

All in all, the above idealised models are able to reproduce accurately some foam morphological parameters such as the cell-size distribution, face-by-cell count, and edge-by-cell count; provided that the considered sample is large enough to be statistically representative of the overall foam [141]. However, they fail to account for local morphological variations as, for instance, struts cross-section shapes, variation of cell-wall curvatures, local defects (such as damaged or partially missing struts) or the existence of partially reticulated foams (presenting an intermediate state between closed and open microstructures). Such instances of local morphological changes require further construction [67]. In order to introduce these microstructural features, the Distance Neighbour based Random Sequential Addition (DN-RSA) was developed in Reference [156] and further improved and enriched in Reference [79] to represent complex struts morphologies through level set functions.

An alternative model was also developed in References [13, 51] in order to overcome some issues and limitations of random packing algorithms. This model consists in generating a "skeletal" foam using a Voronoï tessellation constructed from randomly packed mono-dispersed spheres. The constructed ligaments are then "dressed" with circular cross-sections (see Figure 1.13), the areas of which vary along the length of the ligaments; based on empirical expressions developed from in-situ measurements. Nevertheless, though theses models are able to produce faithful geometries (at least statistically[1]) and quality meshes for FEM simulations, they still require some input parameters that are difficult to assess.

[1] In the sense that the main statistical geometric features of the considered foam, such as the cell-size distribution, the mean number of faces per cell, etc., are reproduced

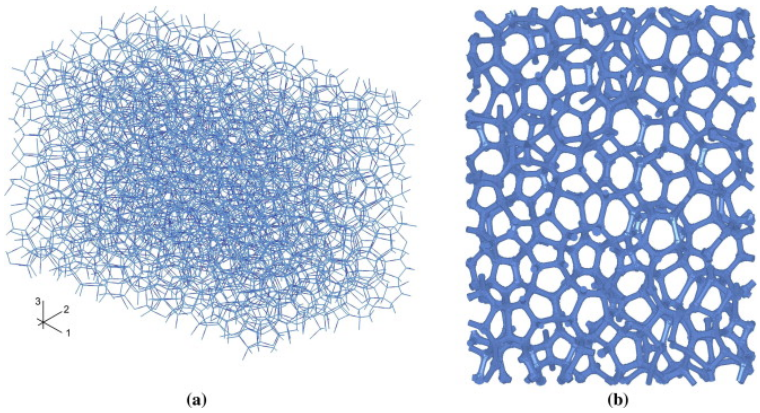


FIGURE 1.13: (a) 3D rendering of random soap froth with $12 \times 8 \times 8$ cells and (b) slice from the same model after ligaments are dressed with beam elements [51].

1.3 Motivation and present contributions

As geometrical features have a considerable impact on the mechanical behaviour of foams [58, 70], manufacturing cellular materials with the desired mechanical behaviour is a challenging task. Indeed, on the one hand, foams exhibit a large versatility in their physical characteristics (mechanical, thermic and/or electromagnetic); allowing a large field of applications. On the other hand, obtaining a specific physical property can be complex and cumbersome as the link between a given property and a foam's microstructure is far from being trivial. For instance, inside a same sample different pores may have variable mechanical properties [62]. Moreover, ensuring the production of foams with nominal and stable microstructural and mechanical properties is still a challenging task in the research and industrial domain [117, 154]. As a consequence, there is a significant demand for quality control and characterisation of new foam products in the industry. In this context, reliable computerised foam models that require minimum user input and reasonable computer means are of potential use [113]. Such numerical model may need to exploit the geometrical information of Representative Volume Elements (RVEs) obtained from Computer Tomography scans (CT-scans) of foams; while requiring affordable computer resources.

The current work proposes a new model, hereafter called *Ellipsoidal Model*, to obtain a geometric representation of foams from CT-scan images. This model relies on the Poisson surface reconstruction [76] from points and normals associated to the exterior surfaces of ellipsoids. It allows obtaining a closed surface representation of the considered foam. The surface provided by the Ellipsoidal Model can be subsequently meshed for based finite element methods simulations. Providing a geometric model with the properties of a Boundary REPresentation (B-REP) instead of a mesh offers the possibility to "tune" to some extent the mesh generated from that B-REP (for instance different levels of mesh refinements can be generated from the same geometric surface). In the present thesis, the three-dimensional mesh generator used is Gmsh [54].

Last be not least, the image analysis procedure has been designed to support the processing or large CT-scan images, i.e. image data that can not be fitted entirely into the Random Access Memory (RAM). This feature is achieved by using the Insight ToolKit (ITK) framework [69] and its streaming capabilities [194]: large CT-scan images can be processed slice-by-slice by loading only one image slice at a time into the RAM. This specific feature allows even standard consumer computers to process large CT-scan images, at a cost of a larger computational time and with a limitation on the minimum allowed slice thickness in the CT-scan images. This feature and associated costs and limitations are described in Section 3.

The remainder of the present thesis is organised as follows. Chapter 2 describes the proposed image analysis steps, along with the corresponding streaming capabilities (Chapter 3) and its impact in peak memory usage (Section 3.12). Section 3.13 shows how ellipsoids (parents and auxiliaries) are fitted to the cells of a foam. Chapter 4 describes how a B-REP is extracted from these ellipsoids. Finally, Chapter 5 compares FEM simulations of a real-world foam, which geometry has been reconstructed from CT-scan images, with respect to experimental data taken from [62].

The novelties presented in this thesis for the reconstruction of the geometry from 3–dimensional CT–scan images of foams are the following:

- Geometric reconstruction using ellipsoids from identifying cells and "auxiliary" ellipsoids for reproducing faithfully the microstructure, including imperfections such as missing or deformed struts.
- Avoid computationally expensive algorithms such as watershed and H-maxima transform, commonly used to prevent oversegmentation, by clustering and merging overlapping ellipsoids.
- Geometric surface representations generation of foams microstructures from which different meshes can be constructed or merged with user-supplied geometries.
- Streamed image-analysis procedure which allows either to process 3-dimensional CT-scan images that can not fit into the available RAM and, possibility, to parallelise it among several processing units.
- Possibility to "feed" other reconstruction models as demonstrated in Section 4.1.

In order to keep a global insight of the entire proposed procedure, Algorithm 1 summarises it.

Algorithm 1 Ellipsoidal Model summarising algorithm

Input: 3D CT-scan image of a foam.
Output: Surface geometric model of the foam.

1. **Identification of cells.**
   - a: Threshold the image (Section 2.3.4) and clean it from spurious pixels (Section 2.3.5).
   - b: Replace each pixel value by its distance value from the closest pixel representing a wall/strut (Section 2.3.6).
   - c: Identify cell centre candidates by computing local maxima (Section 2.3.8) of the previously computed distances.
   - d: From each cell centre candidates, grow "parent" ellipsoids (Section 2.3.9).
   - e: Merge overlapping "parent" ellipsoids in order to avoid oversegmentation (Section 2.3.10).
   - f: Identify the microstructure by growing "auxiliary" ellipsoids from the "parents" (Section 2.3.11).
2. **Reconstruction of the microstructure.**
   - a: Discretise all ellipsoids into a set of points and keep all "exterior" points (Section 4.2.1).
   - b: Get a geometric representation of the microstructure using a Poisson surface reconstruction (Section 4.2.3).

1.4 Publications

This work has directly resulted in the publication of the following article in peer reviewed journal as main author:

- Christophe Leblanc, Nanda Gopala Kilingar, Anne Jung, Ehab Moustafa Kamel, Thierry Jacques Massart, Ludovic Noels, Eric Béchet. *Analysis of an open foam generated from computerised tomography scans of physical foam samples*. International Journal for Numerical Methods in Engineering, 2022; 1- 29. doi:10.1002/nme.7008

This work has been presented at the following international conferences:

- Christophe Leblanc, Van Dung Nguyen, Ludovic Noels, Eric Béchet. *Streamable Laguerre-Voronoi Tessellation Model for Tomographic Images*. 11th World Congress On Computational Mechanics WCCM XI - ECCM V - ECFD VI, Barcelona, Spain 2014.
- Christophe Leblanc, Van Dung Nguyen, Ludovic Noels, Eric Béchet. *Streamable Generalized Voronoi Tessellation Model for Tomographic Images*. European Congress on Computational Methods in Applied Sciences and Engineering - ECCOMAS Congress; Hersonissos, Grece 2016.
- Nanda Gopala Kilingar, Ludovic Noels, Thierry Jacques Massart, Karim Ehab Moustafa Kamel, Bernard Sonon, Christophe Leblanc, Anne Jung. *Data driven computational analysis of open foam RVEs*. Computational Methods in Multiscale, multi-uncertainity and multi-physics problems conference; Porto, Portugal 2019.

1.5 Summary of the thesis

1.5.1 Contribution

The contribution presented in this thesis is two-fold: First a new generic image analysis procedure is proposed were computationally expensive morphological algorithms such as the watershed and the H-maxima transforms are replaced by merging of overlapping ellipsoids. Moreover, the image analysis procedure is streamable, allowing the processing of huge CT-scan images with commercial personal computers. Second, the reconstruction of the geometry of the microstructure is performed by using growth of ellipsoids, instead of spheres as in the Laguerre-Voronoï models, allowing to better take into account cell anisotropies. Moreover, from these "parent" ellipsoids, "auxiliary" ellipsoids are grown such to accurately reproduce the microstructure. From these ellipsoids, a suitable geometric model can be extracted and meshed for FEM models.

1.5.2 Chapter 2

This chapter presents the ITK framework used for implementing the different proposed steps; as well as those steps for identifying cells from CT-scan images of foams. The use of the ITK framework adds flexibility to the image analysis procedure, allowing the user to add/remove/replace some steps by other as he or she sees fit for its own needs.

Step 1 in Section 2.3.4 is the threshold of gray-level images. In this thesis the algorithm of Ridler and Calvard [139] was used. This algorithm analyses the histogram of the CT-scan image to process and determines automatically a suitable threshold value. Thanks to the ITK framework, it is possible to use more advanced thresholding techniques, such as the popular Otsu algorithm [126], if deemed necessary by the user.

As threshold algorithms may still classify background pixels as foreground pixels, step 1 is completed in Section 2.3.5 by a *Box filter* which aims to remove noisy pixel. The *Box filter* simply removes chunks of connected pixels which fit inside a given box. Although usually the binary opening is used for such situations, it has been found that this morphological operation (see Appendix C) is not suitable in the context of foam. Indeed, as demonstrated in Figure 2.13, a simple binary opening with a spherical stencil tends to remove noisy pixels along to legit ones, creating artificial holes in thin cell walls or removing thin struts. It should be possible to reduce this effect by choosing a better structuring element, but this choice is not obvious and require user's input. Instead, the *Box filter* is simpler to use and leads to better results.

Step 2 in Section 2.3.6 is devoted to compute the distance transform (see Appendix B). Given that cell centres are usually the pixels located the furthest from the cell walls/struts, the local maxima of the distance transform can be a good indicator of these cell centres. Usually, in the context of foams, the Euclidean distance is used for computing the distance transform. Several algorithms for computing the Euclidean distance transform exist. In this thesis, the algorithm of Maurer et al [111]. has been chosen. This algorithm is indeed reasonably fast (linear with the number of pixel in an image), and easily adaptable for streaming (see Appendix A).

Step 3 in Section 2.3.7 is an optional step for post-processing the Euclidean distance transform. Here, no such post-processing has been used. However, on some occasions, a smoothing step may be useful and can be applied here.

Step 4 in Section 2.3.8 describes how local maxima of the distance transform can be computed. Again, several algorithms exist for computing local maxima. The native algorithm proposed in the ITK library was not used because not streamable. Instead, an adapted and customised version for 3D images of the algorithm of T.Q. Pham [130] was designed and used. The fact that this algorithm scans for local maxima candidates along lines, and then determine the "true" local maxima by inspecting the neighbourhood of each candidate, makes it easy to adapt for streaming images by slices: it is enough for the slices to contain the scanning lines and have a thickness equal to the extent of the inspection neighbourhoods.

Unfortunately, if local maxima usually represent suitable identifiers for the cell centres, some of them may not. It may happen that some local maxima are not "true" maxima, but rather saddle points of the distance transform. Algorithms computing local maxima usually can not discriminate genuine maxima from saddle points. It is the case of the algorithm of T.Q. Pham. In order to remove saddle points, a procedure proposed by Lopez-Reina [99] in her master-thesis has been applied. For each detected local maxima (genuine and saddle), this procedure remove saddle points by considering the eigenvalues of the local Hessians at each detected local maximum. However, computing local Hessians in discrete images is an ill-conditioned problem. This ill-conditioning issue was solved by locally computing a discretised scale-space representation of the distance transform at each local maximum, which has the property of smoothing local Hessians.

Step 5 in Section 2.3.9 discusses how genuine local maxima are used to "seed" parent ellipsoids in cells. From the genuine local maxima, parent ellipsoids are initialised as unit spheres. They are then grown using the algorithm of R. Deits et al [39]. which consists in growing iteratively each ellipsoid independently inside a polyhedron (see Appendix E). This polyhedron is itself iteratively updated using its associated ellipsoid and the surrounding pixels which are seen as obstacles limiting the growth of the ellipsoid (see Figure 2.28). Contrary to other spheres/ellipsoids growing algorithms such the ones developed by, e.g., A. Alpers and O. Sedivy [4, 150, 151], the algorithm of R. Deits et al. has been designed to be fast[2] and highly parallelisable. These characteristics allow to process rapidly a large number of ellipsoids. As a matter of fact, the algorithm of R. Deits et al. is of several orders of magnitudes faster than the algorithm of A. Alpers and O. Sedivy. The only drawback, though, is that there is no guarantee that grown ellipsoids will not overlap with each other, as it is the case for A. Alpers and O. Sedivy. However, this drawback is not an issue in this thesis.

Actually, it is even an advantage. Indeed, in Step 4 several local maxima can be associated to a same cell. Classically, this issue is tackled using the (expensive) watershed transform (and, possibly, the H-maxima transform (see Appendix C) in order to guarantee a univocally correspondence between local maxima and cells. Here, grown ellipsoids by the algorithm of R. Deits et al. associated to local maxima

[2] It has been initially designed to determine in real time a *safe* region around a moving robot surrounded by obstacles.

located in a common cell will likely more overlap with each other rather than overlap with ellipsoids associated to a neighbouring cell. This is the subject of Section 2.3.10 which explains how this overlapping criterion is computed using a Monte-Carlo method for evaluating intersection volumes between ellipsoids.

As obtaining an accurate volume estimation of a given region using a Monte-Carlo method (see Appendix H) can be computationally expensive, three "cheap" tests are carried out, previously to maybe requiring the computation using the Monte-Carlo method. The first test checks if two neighbouring ellipsoids are colliding. If not, there intersection volume is zero and no further computation is needed. The second test consist in computing an upper bound of the intersection volume, while the third test calculates a lower bound of the intersection volume. If the overlapping criterion falls outside these bounds, a Monte-Carlo computation is not needed.

This section also describes how the ellipsoids belonging to a common cell are clustered together following this overlapping criterion using the GDBSCAN algorithm (see Appendix F). Although numerous other clustering algorithms exists, the GDBSCAN algorithm has been chosen mainly because it designed to cluster objects based on a density criterion. In the context of clustering ellipsoids which overlap the most (thus which are the most densely packed) this is a highly desirable feature.

Finally, each cluster of ellipsoids should univocally corresponds to a given cell of the considered foam. Ellipsoids belonging to a common cluster are then merged into one MVCEE using the algorithm of E.A. Yildirim [185]. The advantage of this algorithm, is that it given sets of ellipsoids, it still outputs ellipsoids that can be used in the next step (step 6 auxiliary ellipsoids).

It should be noted that clustering and merging ellipsoids is not mandatory for this next step. However, as discussed in Section 2.3.9 and illustrated in Figure 2.44, clustering and merging ellipsoids can be useful in the case of sets of undesirable connected voxels that the previous steps did not remove. Moreover, clustering and merging ellipsoids is also useful if one would like to obtain some statistics such as the presence of a preferred cell orientation/anisotropy.

Step 6 in Section 2.3.11 shows how auxiliary ellipsoids are computed. They are seeded from discrete points evaluated from the surface of the parent ellipsoids and then grown using again the algorithm of R. Deits et al. The only difference with the growth of the parent ellipsoids, is that artificial obstacles are added for the auxiliary ellipsoids in order for them to "decay" to their respective parent ellipsoids. As illustrated in Figures 2.49 and 4.12, auxiliary ellipsoids are highly effective for reproducing small features and defect present in a given microstructure. Section 3.13 shows qualitative results of the reconstructions of the microstructures of two real-world foams, one with open cells and the other with closed cells. It can be seen in Figures 3.32 and 3.35 that auxiliary ellipsoids closely reproduce the microstructure.

1.5.3 Chapter 3

This chapter discuss how the algorithms chosen in steps 1 to 6 can be used for streaming. Streaming is performed by extracting slices of a 3D CT-scan image and processing each slices independently through steps 1 to 6. If needed, the thickness of given slice can be increased for accommodating the needs of one or another algorithm through the *negotiation process* between filters in the ITK framework. For instance, the algorithm of T.Q. Pham may require a neighbourhood around a given local maximum candidate that falls outside the current slice[3]. In this case, a slightly thicker slice can be returned through the *negotiation process* for accommodating the neighbourhood.

The situation for the distance transform is however different and is discussed thoroughly in Section 3.7: A first wrong distance transform is first computed on the current slice. Then, an *extended region strategy* is applied for determining the amount by which the thickness of the current slice should be increased so that for the distance transform to become exact on this slice.

The growth of ellipsoids while streaming is discussed in Section 3.10. The strategy used is similar to the one used for algorithms requiring a given neighbourhood, such as the algorithm of T.Q. Pham. The difference, is that the required neighbourhood around a given ellipsoid is not known in advance. This neighbourhood is therefore iteratively computed as the ellipsoid grows: while the ellipsoid grows outside the current slice, a thicker slice is requested through the *negotiation process* of ITK, until the ellipsoid has reached its final volume.

Section 3.11 discusses the fact that the applied above strategies for streaming have one major limitation: it is useless if a cell traverses the whole foam. However, this case can be considered quite exceptional, especially for large CT-scan images where the streaming can be useful.

Section 3.12 shows the effectiveness of the strategies used for streaming on three different datasets, where two were artificially generated and one was obtained from a real-world foam sample. For each of the three datasets, Figure 3.28 shows that indeed peak memory usage using the proposed image analysis steps are lower than a classic image analysis steps involving the watershed transform. Besides, the usage of streaming (here with ten slices) allowed to reduce the peak memory usage further by approximately a factor of ten. It even allowed to generate the geometry of an artificial honeycomb foam containing $34,410$ cells, while this was not possible with the classical image analysis steps and the available computation resources.

Finally, Section 3.13 shows qualitatively that the proposed algorithms in steps 1 to 6 are indeed able to faithfully reproduce the complex geometry of real-world foam. More precisely parent ellipsoids were able to correctly and uniquely identify cell pores, while auxiliary ellipsoids were able to closely fit cell struts.

[3] The neighbourhood, not the local maximum candidate.

1.5.4 Chapter 4

In this chapter, two models for the reconstruction of the geometry of foams from 3D CT-scan images were presented: the *DN-CT-SCAN* model and the *Ellipsoidal Model*.

Both models rely on a first step consisting into an image analysis of 3D CT-scan images. The image analysis stage provides the models with uniquely fitted ellipsoids and associated polyhedra to identified cells.

Section 4.1 gives a brief description of the *DN-CT-SCAN* model, as well as reconstruction results by this model using parent ellipsoids and their associated polyhedra. In Figure 4.6, it can be seen that the reconstruction using polyhedra gives better results in terms of the Hausdorff distances between the reconstructed geometry and the (thresholded) CT-scan images of a real-world open foam provided by A. Jung from the Saarland University, Germany. This is an expected result, as each ellipsoid is circumscribed by its corresponding polyhedron. Polyhedron thus closer matches the geometry of the cells.

Section 4.2 describes how a geometric model is obtained from parent and auxiliary ellipsoids: All ellipsoids are discretised in a set of points and associated normals. Points and associated normals strictly inside at least one ellipsoid are discarded and the other are kept and used via the Poisson surface reconstruction to obtain a surface of the microstructure of the foam. Again, Figure 4.11 shows a good agreement in terms of Hausdorff distances between the geometric model obtained by the *Ellipsoidal Model* and the (thresholded) CT-scan image of a real-world open foam provided by A. Jung.

Geometric models generated by the *Ellipsoidal Model* are output in **.geo* file format that can be subsequently used by the software *Gmsh*. This allows to generate several meshes from one file. This feature can be useful for generating meshes at different refinement levels from the same geometry, or combining the geometric model with other geometric elements. For instance for adding some external geometries using boolean operations for the aim of satisfying any specific needs or study. This feature was actually exploited in this thesis for generating from a unique geometric model the four meshes used for computing by FEM the strain-stress curves shown in Figure 5.10.

1.5.5 Chapter 5

This chapter is devoted to assess the usefulness of both the *DN-CT-SCAN* model and *Ellipsoidal Model* for generating suitable geometries for FEM simulations from CT-scan images of a real-world open foam provided by A. Jung [62], using the above proposed image analysis steps.

Section 5.2 introduces the main computational approaches for simulating the mechanical behaviour of a cellular or composite material. A popular and standard approach, is the so-called multi-scale approach which needs RVEs of the body to simulate. The ability to provide accurate geometric model of RVEs for the multi-scale approach is one of the main contribution of this thesis; another being the ability to provide identifications of cells in cellular materials using ellipsoids and polyhedra for the use in other models. This was demonstrated in this thesis with the use of the *DN-CT-SCAN* model.

The next sections discuss simulations using the geometries generated by both the *DN-CT-SCAN* model and the *Ellipsoidal Model* with the J_2-elasto-plastic material law, and for three different sets of boundary conditions. It has been concluded that the so-called *mixed* boundary conditions leads to the better simulated strain-stress results for both models compared to experimental strain-stress curves. In particular, the initial strain-stress slope is reproduced, as well as the plateau stage. The densification stage was however not reproduced due to the fact that contacts between struts were ignored in the FEM model.

Chapter 2

Image analysis

2.1 Summary

The present chapter details all the steps of part 1 ("*Identification of cells*") of the summarising algorithm 1 and discuss the differences with the standard analysis procedure generally performed when identifying cells form 3D CT-scan images (Section 2.2).

Figure 2.1 illustrates both standard and proposed image analysis procedures. Steps 1 to 4 are similar in both procedures, up to the minor details in steps 2 and 4. Their aims are the following:

1. Thresholding: provide a gray–level image for the distance transform to be performed on; and, possibly, remove spurious voxels.
2. (Inverse) Euclidean distance transform: use the fact that voxels located at cell centres should, ideally, have lower/higher associated distance values than their neighbouring voxels.
3. Smoothing (optional): smooth the distance transform in order to reduce the number of spurious local minima/maxima to be identified.
4. Identification of local minima/maxima: provide a first guess for cell centres and seed, respectively, the watershed transform or the initial "parent" ellipsoids to be grown.
5. Watershed transform/"parent" ellipsoids: identify cells univocally.
6. "Auxiliary" ellipsoids: reproduce the local geometric features of each identified cell.

For steps 2 and 4, respectively, the Euclidean distance transform instead of its inverse is computed, and local maxima instead of local minima are searched for. Moreover, step 4 includes a small difference where spurious local maxima located on missing cell walls are discarded.

The main differences between both procedures appear from step 5 where the watershed transform (and, possibly, the H-maxima transform) of the standard procedure is replaced by the growing and merging of so–called "parent" ellipsoids. The main advantages of this replacement are:

- Growing and merging ellipsoids is computationally cheaper than a watershed transform (see Section 3.12 for an assessment).

- Over–segmentation is controlled through merging of ellipsoids (see Section 2.3.9).

Last but not least, the growth of "auxiliary" ellipsoids from their respective "parents" allows to reproduce faithfully small features captured in the 3D CT–scan images (see Section 4.2 for an assessment).

Finally, all the proposed steps have their "streamed" counter-part (see Section 3) allowing the processing of data that does not fit into the RAM.

2.2 Standard image analysis procedure

The microstructure of a foam sample is described in terms of geometric characteristics of its cells. For algorithms based on the image analysis of CT-scan images, identifying these cells and extracting the associated geometric characteristics, requires the use of a chain of image processing algorithms [88]. The aim of this processing chain is to unequivocally identify the cells and their shapes. A typical processing chain is given in References [138, 88] and outlined in Figure 2.1a.

In principle, identifying the cells is easy: once a black and white image obtained via thresholding (step 1), it is only necessary to identify where the inverse distance transform (step 2) presents local minima which identify cell centres. However, in practice, the (inverse) distance transform may present several local minima for a given cell. As a consequence, superfluous local minima have to be removed. Typically, this is achieved using morphological transforms such as the watershed transform, optionally preceded by a (possibly adaptive) H-maxima transform [88, 125, 33, 57], or even gray-scale reconstruction [176], in order to avoid cell oversegmentation. Figure 2.2 illustrates this procedure.

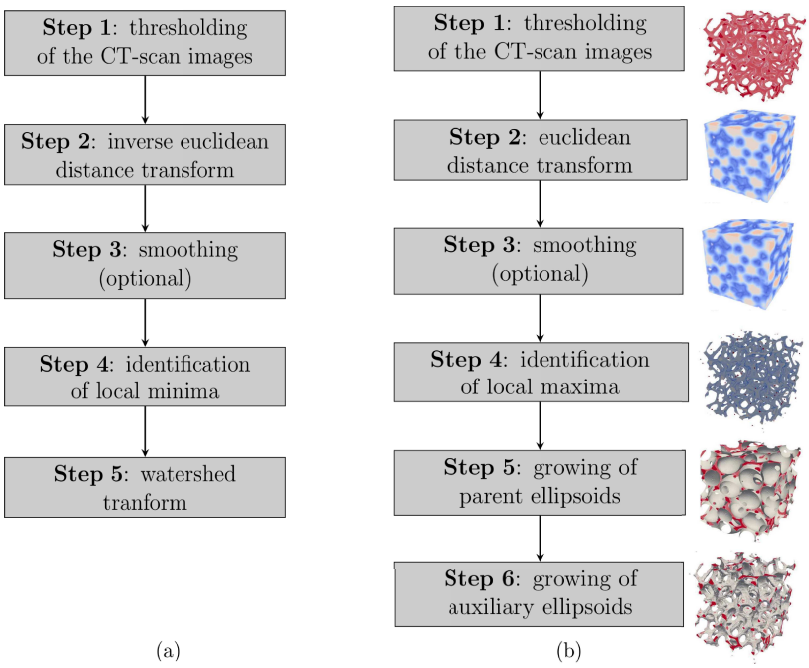


FIGURE 2.1: (a) Standard image processing steps. (b) Proposed image processing steps.

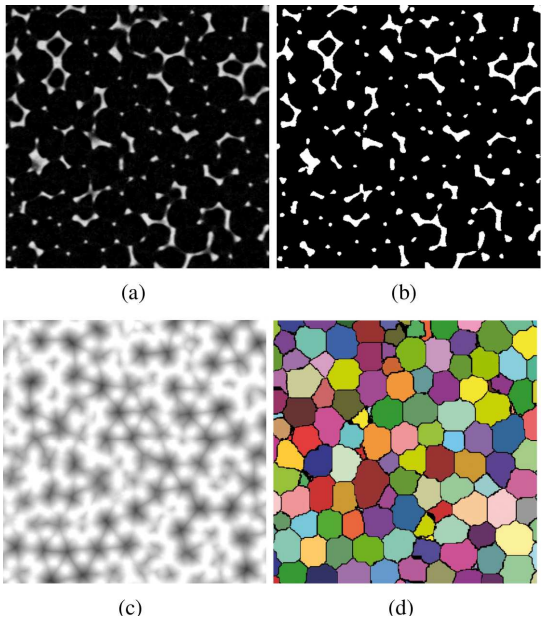


FIGURE 2.2: Foam reconstruction: (a) sections of the original 3D image, (b) thresholding, (c) the inverted distance image, (d) and the reconstructed foam cells [138].

2.3 Proposed image analysis procedure

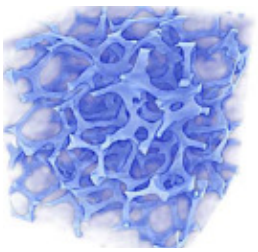
The following section is structured as follow:

- Description of the proposed image analysis procedure and the framework used (Section 2.3.1) and the different steps involved, namely (see Figure 2.1b):
  - Section 2.3.2 (step 0) shows of to convert an RGB image to a gray-level image. This step is optional if a gray-level image is already available.
  - Section 2.3.3 (step 0bis) describes how to take into account CT-scan images with non-cubic voxels. This step is also optional.
  - Section 2.3.4 (step 1) explains how to identify struts and/or walls boundaries in a gray-scale image by applying a threshold.
  - Section 2.3.5 (step 1bis) describes how to remove spurious voxels in a noisy image.
  - Section 2.3.6 (step 2) shows how to compute the distance transform from a thresholded image; with the aim of identifying the centres of cell pores by locating local maxima of the distance transform.
  - Section 2.3.7 (step 3) mentions the possibility to add an optional smoothing filter to the distance transform in order to reduce the number of spurious local maxima.
  - Section 2.3.8 (step 4) describes how to compute these local maxima, how to remove local maxima located on saddle points using Hessians-based maxima selection, how these Hessians are locally computed on a given neighbourhood, and how this neighbourhood are themselves determined.

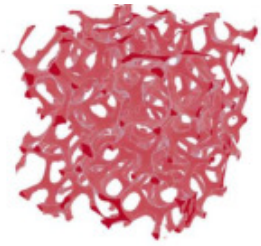
- Section 2.3.9 (step 5) explains how cell pores sizes and orientations are determined based on the growths of "parent" ellipsoids. The detection, clustering and merging of parent ellipsoids belonging to a single pore are explained in Section 2.3.10.
- Section 2.3.11 (step 6) shows how the "auxiliary" ellipsoids are constructed. Their aim is to faithfully describe the microstructure of a foam, defects and missing features included.

The standard image analysis steps usually require the use of morphological algorithms, such as the watershed transform, for unequivocally identifying cells. These algorithms are often computationally heavy, despite efforts for tackling the issue (see, e.g., References [176, 21, 159, 82] for the watershed transform). The present contribution presents an altenative processing chain for identifying cells as outlined in Figure 2.1b.

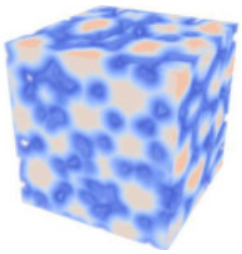
This alternative processing chain replaces the watershed transform (and, optionally, the H-maxima transform or other smoothing algorithms) by a stage of growing and clustering ellipsoids. This, as it will be shown later, is memory-wise much cheaper than morphological algorithms since it does not operate at the voxel level. Figure 2.3 shows an example of the proposed processing steps applied on a 3D CT-scan image of an open aluminium foam.



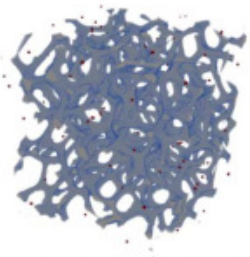
(a) Initial CT-scan image.



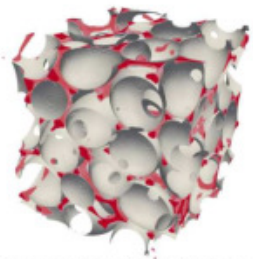
(b) Threshold image.



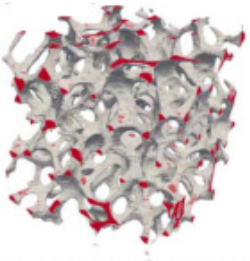
(c) Distance transform image. Low values represented in blue, high values represented in yellow and red.



(d) Local maxima (red balls), exaggerated for visualisation purposes.



(e) Parent ellipsoids (gray surfaces) in their respective cells.



(f) Surface reconstruction obtained with the auxiliary ellipsoids.

FIGURE 2.3: Example of the proposed processing steps of Figure 2.1b applied on a 3D CT-scan image of an open aluminium foam.

2.3.1 Framework used

The proposed processing chain illustrated in Figure 2.1b and steps 1 to 6 of Algorithm 1 has been implemented using the Insight ToolKit (ITK) library as a backbone [69]. The ITK library is handful and versatile: it allows the user to construct its own processing chain using a large choice of pre-implemented algorithms. This library also allows the user to implement customised algorithms which can be integrated in any processing chain. The capabilities of the ITK library can be handful if some CT-scan images need extra processing steps due to some particular characteristics.

The ITK's paradigm lies in the use of processing objects (called hereafter *filters*), and multi–dimensional image data objects. Filters process the image data and can be connected together. Filters then can pass their processed images to each other. A set of connected filters passing data objects to each other is called a *pipeline*, see Figure 2.4.

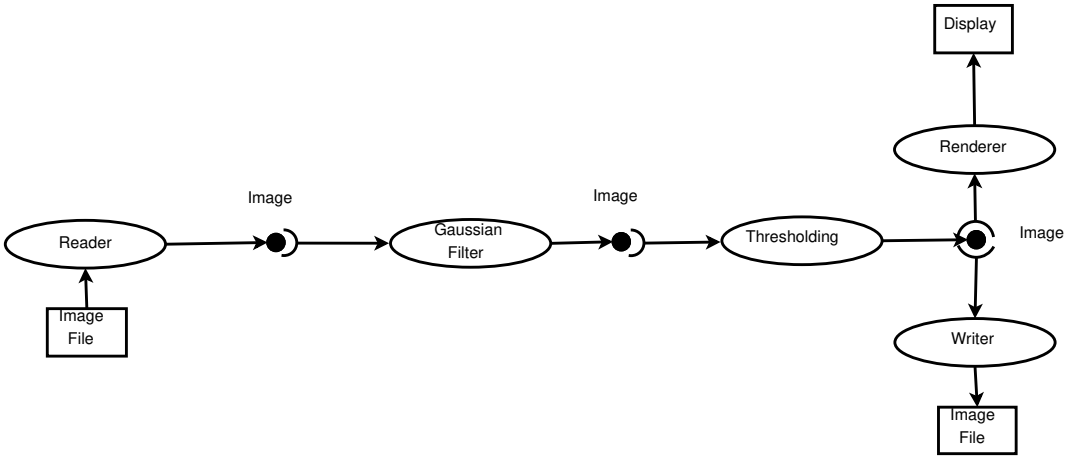


FIGURE 2.4: Example of pipeline. (Image source: *ITK Software Guide* [69], pp. 198).

Once set, a pipeline is executed backward through the *Update()* method and a negotiation process takes place with the filters that define the pipeline. Once the requested amount of data for each filter has been determined, the pipeline is executed forward. Each upstream filter generates the requested data to the downstream filters via the *GenerateData()* method. Indeed, some filters may require images of different dimensions on input and/or on output. For example, an erosion filter requires an extra input (boundary padding) given the size of the requested output. On the contrary, a shrink filter will output a smaller image than the input. This negotiation process presents the appreciable advantage that a given (multi–dimensional) image can be processed into sub–regions. ITK provides a special filter for it: the *StreamingImageFilter*. It is therefore possible to process (multi-dimensional) images that can not fit the memory, into smaller pieces. Figure 2.5 shows an example of negotiation between filters.

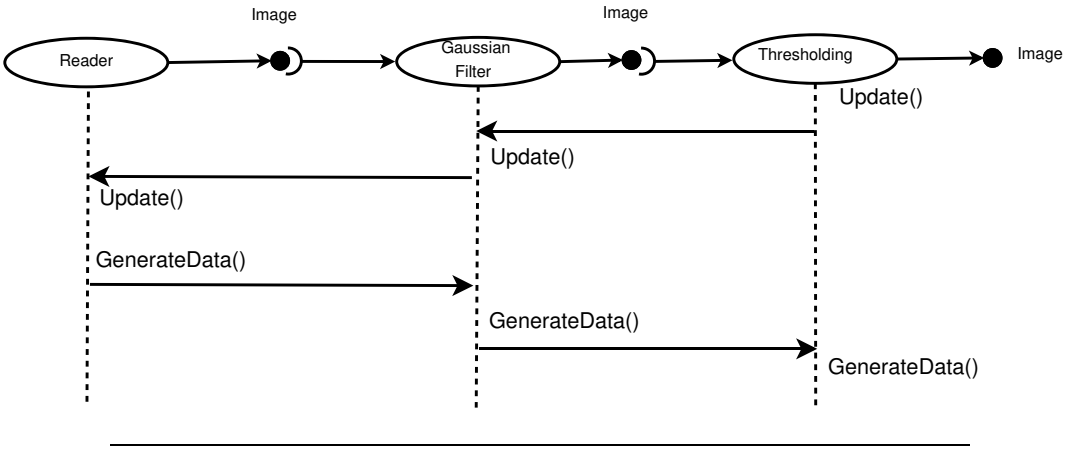
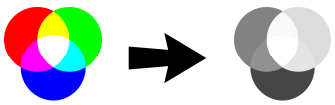


FIGURE 2.5: Example of negotiation between filters in ITK. (Image source: *ITK Software Guide* [69], pp. 199).

2.3.2 Step 0: RGB to luminance filter



- **Aim**: convert a RGB (Red, Green, Blue) image to a gray–level image.
- **Input**: RGB image.
- **Output**: gray–level image.
- **Streaming**: see Section 3.3.

The filters used for the proposed image processing steps only process gray–level images. However, it may happens that some images contain red, green and blue components (R, G and B components). In that case, an RGB image can be converted in a gray-level image by computing its luminance L. Within ITK, the luminance is computed as follows:

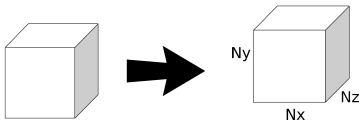
$$L = 0.3R + 0.59G + 0.11B \tag{2.1}$$

Nevertheless, it should be emphasised that equation 2.1 may not be well-suited for getting the wanted features from an RGB image. However, it is believed that most,if not all, X–ray image scan are intensity image and therefore gray–level images. In the extraordinary contrary case, the user can build its own filter pipeline from filters provided by the ITK toolbox for obtaining a satisfactory gray–level image from an RGB image. Then, this filter pipeline can directly be connected to the rest of the pipeline.

TABLE 2.1: Parameters of the *Spacing filter*.

Parameter	Value	Description
N_x	1	Voxel physical length along the x-direction.
N_y	1	Voxel physical length along the y-direction.
N_z	1	Voxel physical length along the z-direction.

2.3.3 Step 0bis: Spacing filter



- **Aim**: add spacing information at the voxels of an image.
- **Input**: RGB or gray–level image.
- **Output**: RGB or gray–level image with spacing information.
- **Streaming**: see Section 3.4.

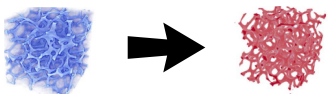
During image acquisition form a sample (by CT-scans or other means), the spatial resolution may not be isotropic. For instance, in the used datasets for this thesis, the resolution[1] in the depth direction (i.e. parallel to the ray emission direction of the scanner) of the sample can be smaller than the resolution in the other perpendicular directions.

Moreover, taking into account the physical spacing between the voxels of a 3D-image is crucial for assessing the mechanical properties of a foam. Indeed, these properties strongly depends on the geometric features of a given foam [56]. Working with a wrong spacing may modify the geometric features of the considered foam (e.g. by reducing the strut length in a given direction) and thus impact the results given by any model using this geometry. Fortunately, ITK provides an automatic mechanism for taking into account inhomogeneous spacing through the *ChangeInformationImageFilter*[2]. Table 2.1 shows the list of expected parameters by the *Spacing filter*.

[1] The resolution in a given direction is defined as the number of voxels in this direction per unit of physical length.

[2] See the *ITK Software Guide* [69] pp. 46 for more details.

2.3.4 Step 1: Threshold filter



- **Aim**: convert a gray–level image into a binary (black and white) image.
- **Input**: gray–level image.
- **Output**: binary image.
- **Streaming**: see Section 3.5.

Thresholding is a basic image transformation method consisting in isolating in a gray-scale image a given foreground object from its background. This is done by checking each pixel value P_i and setting it to 0 if it is below a given threshold T ($P_i < T$) and to 1 otherwise (hence the name of the method). Although simple, the whole difficulty of the method resides in the choice of the threshold T. A simple thersholding procedure with a known threshold T is given as Algorithm 2.

Algorithm 2 Simple thresholding

Require: gray–scale Image I of domain D_I.
Require: $T \in$ range of pixel values.

```
1:  procedure SimpleThresholding(I, T)
2:      i ← 0
3:      for x_i ∈ D_I do
4:          if I(x_i) < T then
5:              I(x_i) ← 0
6:          else
7:              I(x_i) ← 1
8:          end if
9:          i ← i + 1
10:     end for
11: end procedure
```

Though, algorithm 2 reaches quickly its limits and proves to be most of the time unsatisfactory for more complex images (for instance when the illumination varies spatially inside the image [28]). Therefore, considerable efforts have been invested in the so–called automatic thresholding methods, where the threshold T is computed following some criteria. A comprehensive survey of thresholding methods has been conducted by Sezgin and Sankur in 2004 [152]. They categorised the thresholding methods as follow, according to the information they are using:

1. Histogram shape-based methods: maxima, minima and curvatures of a (possibly smoothed) histogram are analysed. For example, Raju and Neelima method [134] belongs to this category. Several histogram shape–based functions have been implemented in ITK by Beare [15].

2. Clustering–based methods: gray-level samples are clustered into two groups (background and foreground). For instance the Otsu's method [126] implemented in ITK by Mosaliganti et al. [115] belongs to this category.

3. Entropy-based methods: entropy of sub–regions of the image are used. An entropy-based function has been implemented in ITK by Beare [15]. More recently, Liang and Cuevas used an entropy measure together with meta-heuristic algorithms for computing multilevel thresholding[3] [94].

4. Object attribute-based methods: shape similarities between objects and coincidence between geometric entities are used.

5. Spatial methods: higher probability distribution functions and sometimes correlation between pixels are used.

6. Locally adaptive methods: the threshold value T is locally adapted at each pixel given some image characteristics. For example, the method developed by Bradley and Roth [28] belongs to this category.

For the foam images considered, it has been noticed that the thresholding method of Ridler and Calvard [139], using the image histogram computed by the ITK class for the switching function, gave all satisfaction. The algorithm developed by Ridler and Calvard can be categorised as an iterative clustering–based method. The algorithm 3 is described here below[4]. Table 2.2 describes its unique parameter and a typical value range for it which depends on the bit depth of the considered image.

The idea behind algorithm 3 is as follow:

1. Construct the histogram of the gray–level image (line 6).

2. Compute a first threshold *thresh* as the average histogram value (lines 7 and 8).

3. Compute two thresholds *mat* and *mbt* as the average histogram values below and above the value of *thresh* (lines 9 to 12).

4. Compute a new threshold value as the average of *mat* and *mbt*, and set *tresh* as this new threshold value (line 13).

5. Repeat from step 3 until the new threshold value does not significantly change from the old one (lines 14 to 21).

It should be emphasised that any of the above thresholding methods can be used if one of them is to be found more accurate and/or convenient to use. Some of them are already implemented inside ITK and can be readily used, the others can be implemented and added to ITK with minimal effort[5].

[3] Multilevel thresholding outputs a grayscale image where not only the foreground plane and the background plane of an image are segmented, but also intermediate planes.

[4] This algorithm comes from `http://www.mathworks.com/matlabcentral/fileexchange/3195-automatic-thresholding`

[5] Depending on the complexity of the chosen method...

Algorithm 3 Ridler & Calvard thresholding

Require: gray-scale Image I of domain D_I.
Require: Nb_Bins: number of histogram bins.
Require: Bin_min: minimum histogram bin value.
Require: Bin_max: maximum histogram bin value.

procedure RIDLERCALVARD(Image)
  $T \leftarrow$ COMPUTETHRESH(I)
  SIMPLETHRESHOLDING(I, T) ▷ See algorithm 2.
end procedure

procedure COMPUTETHRESH(I)
  $Hist \leftarrow$ COMPUTEHISTOGRAM(I).
  $\mu \leftarrow$ COMPUTECUMSUM($Hist$,0,Nb_Bins)
  $thresh \leftarrow$ COMPUTETHRESHSUB($Hist$,0,Nb_Bins,μ)
  $\mu_2 \leftarrow$ COMPUTECUMSUM($Hist$,0,$round(thresh)$)
  $mbt \leftarrow$ COMPUTETHRESHSUB($Hist$, 0, $round(thresh)$, μ_2)
  $\mu_3 \leftarrow$ COMPUTECUMSUM($Hist$, $round(thresh)$, Nb_Bins)
  $mat \leftarrow$ COMPUTETHRESHSUB($Hist$, $round(thresh)$, Nb_Bins, μ_3)
  $newThresh \leftarrow (mat + mbt)/2$

  while $abs(thresh - newThresh) \geq 1$ **do**
    $thresh \leftarrow newThresh$
    $\mu_2 \leftarrow$ COMPUTECUMSUM($Hist$, 0, $round(thresh)$)
    $mbt \leftarrow$ COMPUTETHRESHSUB($Hist$, 0, $round(thresh)$, μ_2)
    $\mu_3 \leftarrow$ COMPUTECUMSUM($Hist$, $round(thresh)$, Nb_Bins)
    $mat \leftarrow$ COMPUTETHRESHSUB($Hist$, $round(thresh)$, Nb_Bins, μ_3)
    $newThresh \leftarrow (mat + mbt)/2$
  end while
  return $thresh$
end procedure

TABLE 2.2: Parameter of the *Threshold filter*.

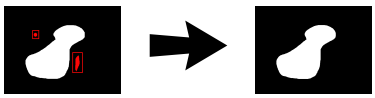
Parameter	Typical value range	Description
Nb_Bins	[256, 65536]	Number of bins in histogram.

24: **procedure** COMPUTEHISTOGRAM(I)
25: $interval \leftarrow \frac{Bin\_max - Bin\_min}{Nb\_Bins - 1}$
26: $Hist \leftarrow$ zero vector of length Nb_Bins
27: **for** $\mathbf{x}_i \in D_I$ **do**
28: $histIndex \leftarrow I(\mathbf{x}_i)/interval$
29: $Hist[histIndex] \leftarrow Hist[histIndex] + 1$
30: **end for**
31: **return** $Hist$
32: **end procedure**

33: **procedure** COMPUTECUMSUM($Hist$, $begin$, end)
34: $sum \leftarrow 0$
35: **for** $begin \leq i < end$ **do**
36: $sum \leftarrow sum + Hist[i]$
37: **end for**
38: **return** sum
39: **end procedure**

40: **procedure** COMPUTETHRESHSUB($Hist$, $begin$, end, μ)
41: $thresh \leftarrow 0$
42: $step \leftarrow \frac{Bin\_max - Bin\_min}{Nb\_Bins - 1}$
43: **for** $begin \leq i < end$ **do**
44: $thresh \leftarrow thresh + i \times step \times Hist[i]$
45: **end for**
46: **return** $thresh/\mu$
47: **end procedure**

2.3.5 Step 1bis: Box filter



- **Aim**: remove small groups of isolated voxels from a binary image.
- **Input**: binary image.
- **Output**: processed binary image with small groups of voxel removed.
- **Streaming**: see Section 3.6.

Origin of the noise in tomographic images

Foam images obtained by X-ray tomography are naturally noisy. The origin of this noise is two-fold. One contribution comes from the electrical noise and round-off error of the detector. The other contribution comes from the X-ray itself which is not strictly monochromatic [72, 165]. Moreover, for foam thin cell walls may appear transparent to the X-rays, especially for polymeric foam which consist in lighter elements than metallic foams.

Indeed, for a narrow beam of mono-energetic photons of incident intensity I_0 passing through a material of mass thickness τ and density ρ, the output intensity I is given by (see Reference [124]):

$$I = I_0 \exp\left(-(\mu/\rho)\tau\right) \tag{2.2}$$

Where the ratio (μ/ρ) is the attenuation coefficient.

The typical wavelength of a X-ray beam is of the order of $\lambda \sim 1nm$, which gives a typical energy of $E = hc/\lambda \sim 3.10^{-2} MeV$. Where, $h \approx 6,626.10^{-34}\, J.s$ is the Planck constant, $c = 299,792,458\, m/s$ is the light speed in vacuum, and $1MeV = 10^6 eV \approx 1,602.10^{-13}\, J$.

For this energy of $3.10^{-2}\, MeV$, NIST tables (see Table 2.3) give the values for the attenuation coefficient for different materials. From this table, it is possible to infer that foam constituted from light elements (hydrogen, oxygen and carbon) will present an attenuation coefficient one order (or more) of magnitude smaller than nickel and aluminium foams. Consequently, X-ray photons are more susceptible to be absorbed by metallic foams, and, therefore, longer scan times are necessary for these type of foams in order to gather enough photons for getting an usable image.

Consequences on foam tomographic images.

The small attenuation coefficients for polymeric foams and the fact that these foams have thin walls produce tomographic images where walls are often missing. Moreover, noisy features have the same gray level intensity of the wanted geometric features [165]; making noise filtering challenging.

For metallic foams, large attenuation coefficients imply longer scanning times in order to obtain an acceptable signal-to-noise ratio for the CT-scan images.

TABLE 2.3: NIST table of the mass attenuation coefficients μ/ρ for a monochromatic photon beam of 3.10^{-2} MeV (typical X-ray energy) for different materials [124].

Material	$(\mu/\rho\ (cm^2/g)$
Hydrogen	$3,57.10^{-1}$
Oxygen	$3,779.10^{-1}$
Carbon	$2,562.10^{-1}$
Aluminium	$1,126.10^{0}$
Nickel	$1,034.10^{1}$

Noise removal

A noise removal technique: *binary opening* The most commonly used tool for noise filtering is the *opening* procedure [125]. A binary opening is a morphological operation that can be used to remove noisy pixels from a black and white image, without disturbing it too much. In consists in a binary erosion, followed by a binary dilation.

The aim of the erosion step is to get rid of the noisy pixels, while the dilation tries to recover as much as possible relevant geometric features erased by the erosion step (see Reference [155], pp.105).

Binary erosion In order to perform an erosion, a *structuring element* is needed. A structuring element is a small connected set of pixels used to probe the considered image [155]. Last but not least, in order to completely define a structuring element, an origin has to be associated to it. Usually, this origin is taken as the centre of the structuring element. Finally, the size and shape of the structuring element have to be chosen accordingly to the geometries of the image objects to be processed.

The technique of binary erosion deals with binary images, i.e. images in which pixels can take only two values (often referred as *black* and *white* pixels or *background* and *foreground* pixels). For an image I, a structuring element B and a connected subset of foreground pixels X, the erosion $E(X,B)$ of set X by the structuring element B can be defined as "*the locus of points* $\mathbf{x}$ *such that* B *is included in* X *when its origin is placed at* $\mathbf{x}$" (P. Soille [155], pp.65):

$$E(X,B) = \{\mathbf{x} \mid B_x \subseteq X\} \tag{2.3}$$

More intuitively, only subsets of connected pixels X in which the structuring element B can fit are kept. The other subsets are discarded (their foreground pixels are set to the value of the background pixels). Figure 2.10b shows a simple example of erosion. Note that the operation of erosion can be generalised to gray-scale images (see Reference [155]) or in term of geodesic distances (see Reference [176]).

Binary dilation As for the binary erosion, the binary dilation deals with binary images and uses a structuring element. For an image I, a structuring element B and a connected subset of foreground pixels X in D_I, the dilation $D(X,B)$ of set X by the structuring element B can be defined as "*the locus of points* $\mathbf{x}$ *such that* B *hits* X *when its origin coincides with* $\mathbf{x}$" (P. Soille [155], pp.68):

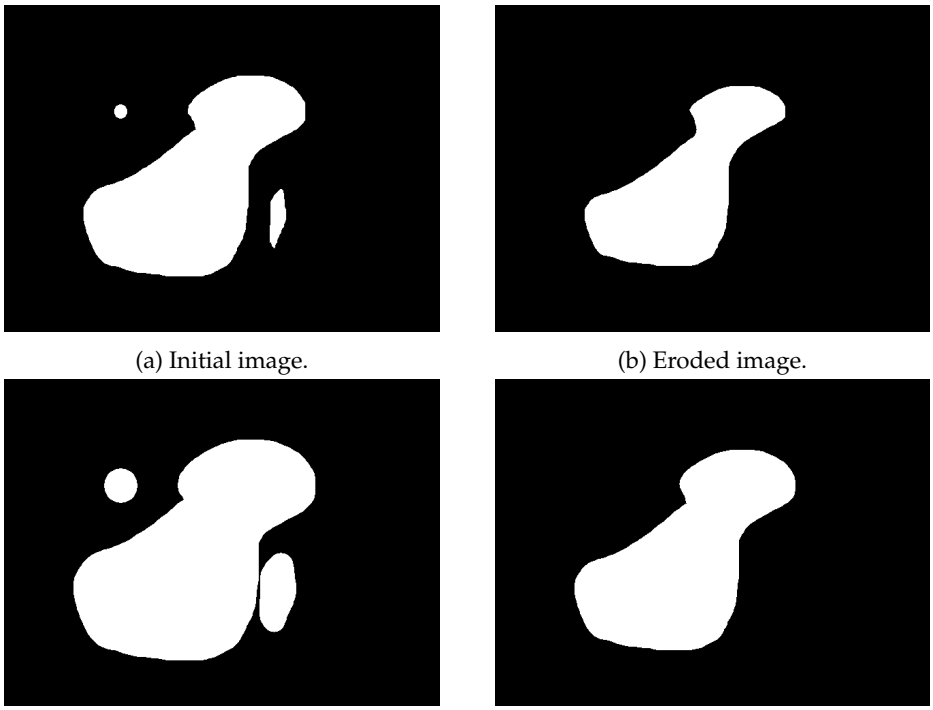
$$D(X,B) = \{\mathbf{x} \mid B_x \cap X \neq \emptyset\} \tag{2.4}$$

More intuitively, any set hitting the structuring element are expanded by the structuring element. Figure 2.10c shows a simple example of dilation. Note that the operation of dilation can be generalised to gray-scale images (see Reference [155]) or in term of geodesic distances (see Reference [176]).

Binary opening The technique of binary opening combines an erosion followed by a dilation. An opening $O(X,B)$ on a connected subset X belonging to an image I with a structuring element B can be defined by equation 2.5.

$$O(X,B) = D\left(E(X,B),B\right) \tag{2.5}$$

The aim of binary opening is thus to try to keep the wanted features of an image while discarding the unwanted ones (often smaller than the wanted features) and is widely used in image processing (as in the *Insight Toolbox* [92], or in *Mavi* [171]). However, erosion and dilation are not inverse of each other. In fact, there is no inverse to both transformations (see Reference [155], pp. 70). As a consequence, even if opening seems to be able to keep quite accurately the wanted geometric feature of an image as it the case for Figure 2.10d, it may fails in other cases (depending on the choice of the structuring element) as discussed later for thin cell walls in Figures 2.13e and 2.13f where thin cell walls are partially lost.



(a) Initial image. (b) Eroded image.

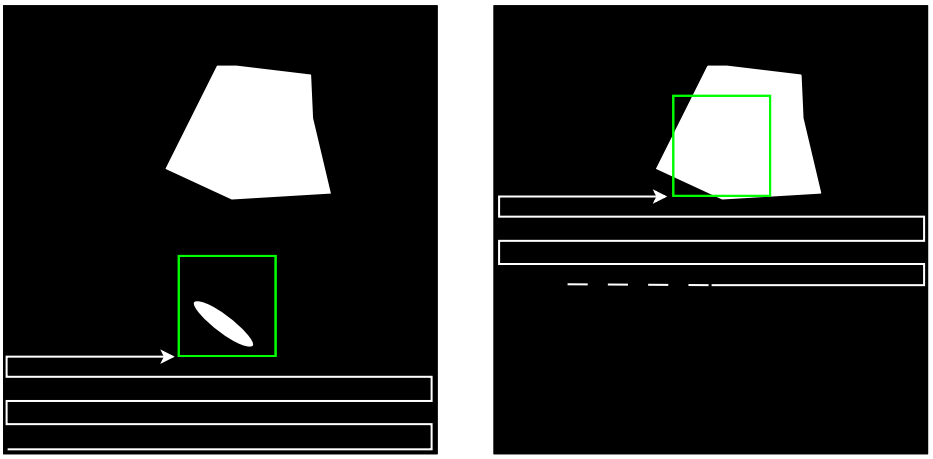
(c) Dilated image. (d) Opening.

FIGURE 2.10: Effect of different morphological operations on an initial binary image (a).

An alternative noise removal technique: *box filter*

As noted in the previous paragraph, the classic opening is not well suited for denoising tomographic images of foams. Though, adaptive techniques exists for computing the opening of a noisy image (see Reference [36] for a survey of the field), which may be suitable; a simpler filtering technique through a box has been preferred here. This technique has been chosen because its simplicity to implement and also its streaming capabilities, i.e. its ability to process images by parts.

The idea of the box filter is as follows: A parallelepipedic box, which size is set by the user, scans each foreground pixel of the image. If a cluster of connected foreground pixels is found to fit entirely inside the box, this cluster is discarded (i.e. its belonging foreground pixels are set as background pixels), otherwise the cluster is kept (figure 2.11).



(a) A cluster of foreground pixels fits into the scanning box and is therefore discarded.

(b) A cluster of foreground pixels does not fit into the scanning box and is therefore kept.

FIGURE 2.11: Intuitive idea of the *box filter*. A box scans the image pixel by pixel and discards any cluster of foreground (white) connected pixels that fits into the box.

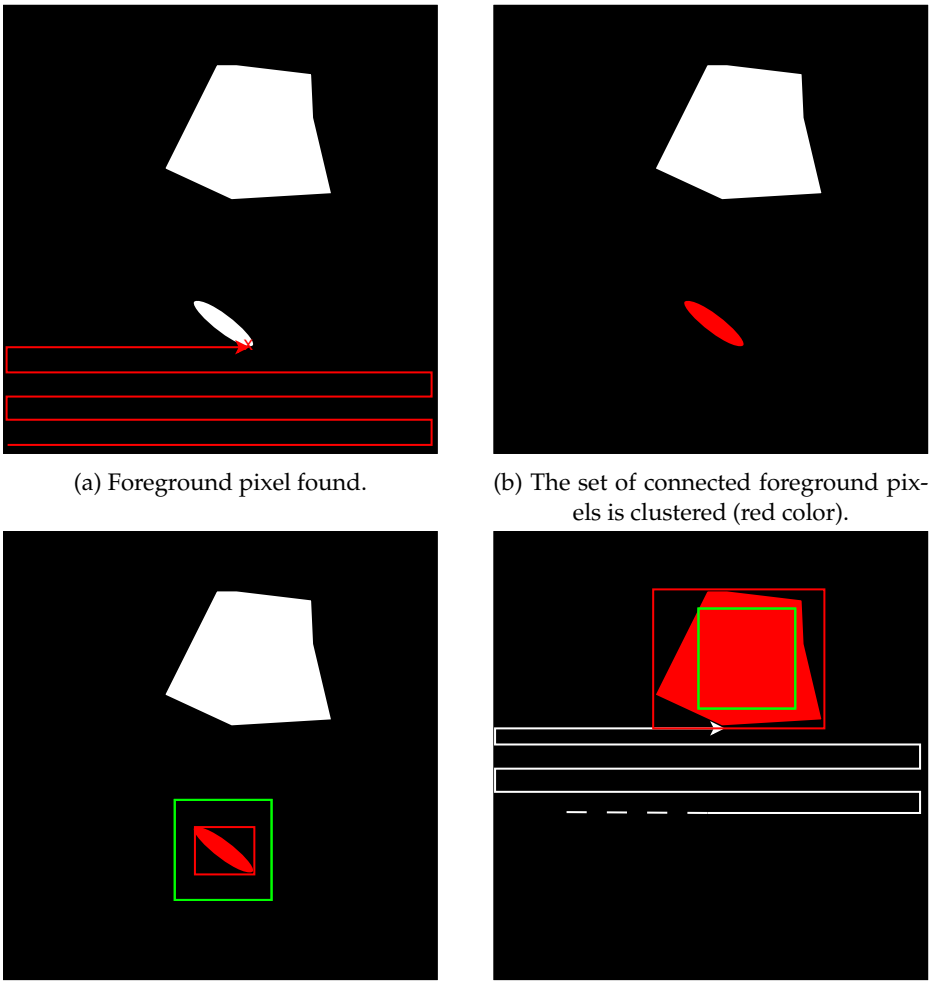
The implementation of this idea is as follow: each time the current scanned pixel turns out to be a foreground pixel, it is marked and, all its connected neighbours are also marked and inserted into a FIFO queue (*First In, First Out*), provided they are foreground pixels and not already marked. The first element in the FIFO queue is marked and then erased, and all unmarked foreground neighbours pixels of the next element (which become the first after the erasure) are marked and inserted at the end of queue. The first (a.k.a. old second element) element is again erased and the process of inserting unmarked foreground neighbours pixels continues until the FIFO queue becomes empty. Step by step a cluster of marked pixels is constructed. Figure 2.12 illustrates the process. Table 2.4 shows the parameters needed by the filter with a typical value range. These values have to be chosen with respect to the maximum size of connected sets of pixels to be discarded.

TABLE 2.4: Parameters of the *Box filter*.

Parameter	Typical value range	Description
B_x	$[5, 20]$	Box size along the x-direction.
B_y	$[5, 20]$	Box size along the y-direction.
B_z	$[5, 20]$	Box size along the z-direction.

During this process, an axis-aligned circumscribing box of the cluster is computed on the fly (the box grows each time a new pixel is added to the FIFO queue). Eventually, the axis-aligned circumscribing box of the cluster is then compared to the user's parallelepipedic box. If the circumscribing box fits inside the parallelepipedic one, the current cluster of pixels is discarded (i.e. all pixels are marked as background pixels), otherwise it is kept. Once the current cluster has been processed, the process continues with the next foreground pixel until no more foreground pixels are to be found or the end of the image is reached. Algorithm 4 describes the process in more details.

Figure 2.13 compares the opening process using a spherical structuring element (two-bottom left images) with the above process of *box filter* (two-bottom right images). It can be seen that the opening tends to smooth and even discard cell walls, while the *box filter* keep the walls intact provided they are not fragmented into small disconnected pieces. The unwanted effect of discarding cell walls by the opening process may perhaps be mitigated by using a more suitable structuring element. However, it is not clear which could be such structuring element. Instead, the *box filter* is simpler to use (no structuring element to choose) and leads to better results.



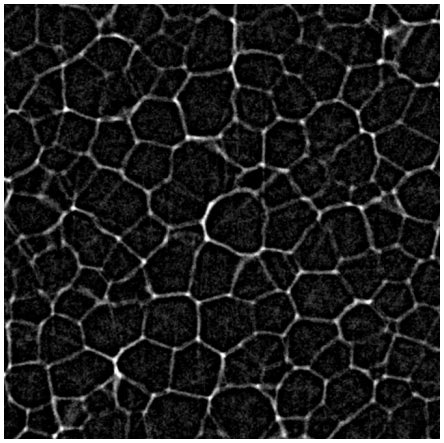
(a) Foreground pixel found.

(b) The set of connected foreground pixels is clustered (red color).

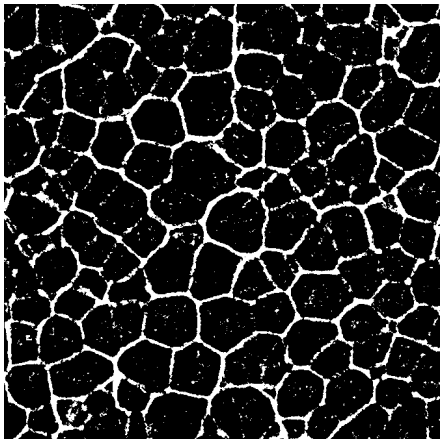
(c) The axis-aligned circumscribing box of the current cluster (in red) is compared to the user's box (in green). If the former fits inside the later, the cluster is discarded.

(d) The process then continues then with the next set of connected foreground pixels.

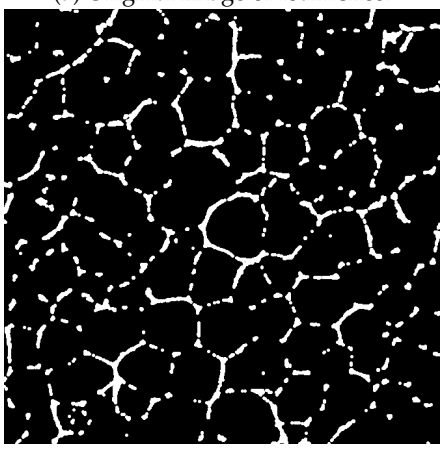
FIGURE 2.12: Illustration of the *box filter* process.



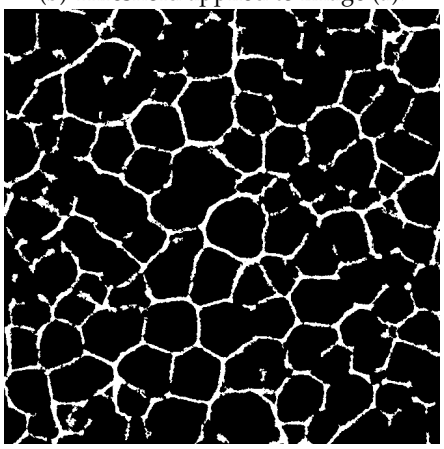
(a) Original image of foam slice.



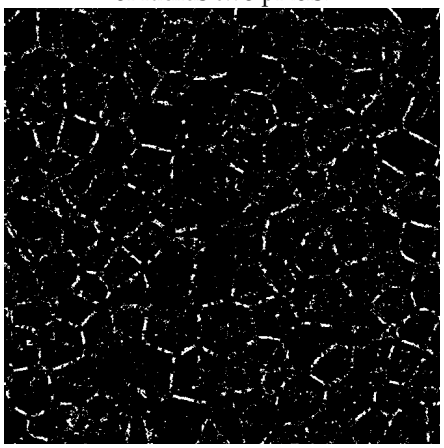
(b) Threshold applied to image (a).



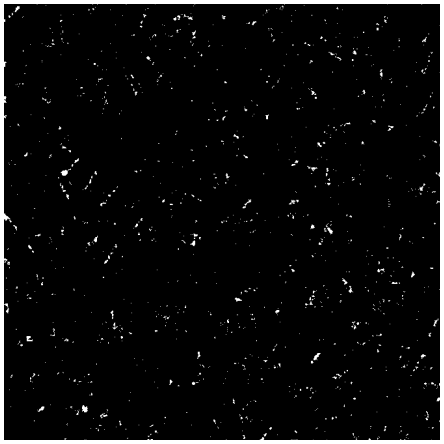
(c) Opening applied to thresholded image (b) with a flat ball structuring element of radius two pixels.



(d) Box filter applied to thresholded image (b) with a box of side ten pixels.



(e) Difference between thresholded image (b) and opening (c).



(f) Difference between thresholded image (b) and box filter (d).

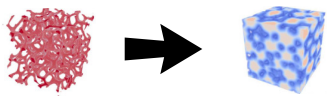
FIGURE 2.13: Noise filtering: opening versus box filter. Opening leads in discarding some thin cell walls (Figure (e)) while they are more preserved with the *box filter* (Figure (f)).

Algorithm 4 Box filter

Require: Binary (black & white) image I of domain D_I.
Require: Axis–aligned Box.
Require: Foreground & Background pixel values.

1: **procedure** BOXFILTER(I, Box, Foreground, Background)
2: $i \leftarrow 0$
3: $clusterNb \leftarrow max\,(Foreground, Background) + 1$
4: **for** $\mathbf{x}_i \in D_I$ **do**
5: **if** $I(\mathbf{x}_i) == Foreground$ **then**
6: $queue \leftarrow \mathbf{x}_i$'s position. $(x_{i,1}, x_{i,2}, \ldots)$
7: $I(\mathbf{x}_i) \leftarrow clusterNb$
8: $clusterBox \leftarrow [x_{i,1}, x_{i,1}, x_{i,2}, x_{i,2}, \ldots]$ ▷ Degenerated box.
▷ Box described by: $[x_{min}, x_{max}, y_{min}, y_{max}, \ldots]$
9: **while** $queue \neq \emptyset$ **do** ▷ $N_I(...)$: neighbourhood of...
10: **for** $\mathbf{x}_j \in N_I(queue.front()) \wedge I(\mathbf{x}_j) == Foreground$ **do**
11: $queue \leftarrow \mathbf{x}_j$.
12: $I(\mathbf{x}_j) \leftarrow clusterNb$
13: **for** $\mu = 0, 1, \ldots, n$ **do** ▷ n: number of dimensions.
14: **if** $clusterBox(2\mu) > x_{j,\mu}$ **then**
15: $clusterBox(2\mu) \leftarrow x_{j,\mu}$
16: **end if**
17: **if** $clusterBox(2\mu + 1) < x_{j,\mu}$ **then**
18: $clusterBox(2\mu + 1) \leftarrow x_{j,\mu}$
19: **end if**
20: **end for**
21: **end for**
22: $queue.pop()$ ▷ Erase position of first pixel in queue.
23: **end while**
24: **if** $clusterBox \subset Box$ **then**
25: **for** $\mathbf{x}_k$ that was in $queue$ **do**
26: $I(\mathbf{x}_k) \leftarrow Background$
27: **end for**
28: **end if**
29: $clusterNb \leftarrow clusterNb + 1$
30: **end if**
31: **end for**
32: **return** I
33: **end procedure**

2.3.6 Step 2: Distance filter



- **Aim**: compute the Euclidean distance transform of a binary image, where white (feature) voxels represent cell walls/struts and are assigned distance zero.
- **Input**: binary image with black and white (feature) voxels.
- **Output**: gray–level image[6] with each voxel value equal to the Euclidean distance to the closest feature voxel.
- **Streaming**: see Section 3.7

In the field of image analysis, the notion of distance between pixels plays a major role [155]. A distance carries with him the notions of balls[7] which themselves provides important topological notions, such as the interior[8], adherence[9] and boundary[10] of a given domain.

Moreover, a distance provides a notion of separations between points and is intimately linked to binary erosion, binary dilation, opening and closing through thresholding (see References [35] and [155], section 3.5.1 pp. 75 for more details). In the modelling of cellular materials, the distance transform of an image (see Appendix B) is widely used (often as input for a watershed transform, see for instance References [149, 165, 138, 33, 38, 103, 95, 169, 32, 156]). Indeed, the maxima of the distance transform are (generally) located at the cell centres, while the zeros of the distance transform allow to identify (partially) cell walls. The *Ellipsoidal Model* (described in Section 4.2) takes advantage of the distance transform for having a first insight about cell centres and cell walls and uses it for initialising an ellipsoid-based algorithm for identifying the cells.

Distance transform algorithms

The first reference sequential algorithm for computing the d_1 and d_∞ distance transforms (DT) is due to A. Rosenfeld and J. L. Pfaltz [143]. This algorithm performs one forward scan and one backward scan for computing the d_1 (or d_∞) distance. It is thus linear in term of number of pixels. The algorithm uses one forward and one backward scan of the image as well as forward and backward neighbourhoods of the current scanned pixel[11]. Although it is not used in this thesis, the algorithm of A. Rosenfeld and J. L. Pfaltz is given by algorithm 5.

[6] Note: here, for visualisation purposes, this gray–level image is coded using a blue-to-red convention instead of a black-to-white one.

[7] For a space E, a closed ball $B[x;r]$ of radius $r > 0$ centred in $x \in E$ is defined as: $B[x;r] = \{y \in E \mid d(x,y) \leq r\}$

[8] For a domain D, $x \in D$ is in the interior $int(D)$ of D if, and only if $\exists r > 0 \mid B[x;r] \subset D$.

[9] For a domain D, $x \in D$ is in the adherence $adh(D)$ of D if, and only if, $\forall r > 0$, $B[x;r] \cap D \neq \emptyset$.

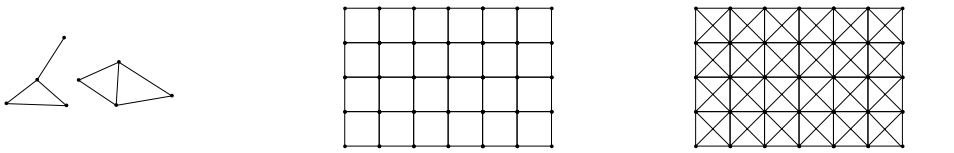
[10] The boundary $fr(D)$ of a domain D is defined as $fr(D) = adh(D) \setminus int(D)$.

[11] The forward (resp. backward) neighbourhood of a current pixel, are its neighbouring pixels coming before (resp. after) him in raster scan order.

Algorithm 5 DT Rosenfeld and Pfalts [143]

Require: Binary image I of domain D_I.
Require: Background pixels to 0.
Require: Foreground pixels to 1.
procedure $DT_{\text{ROSENFELD}}(I)$
  for forward scan of all pixels $\mathbf{x}_i \in D_I$ **do**
    if $I(\mathbf{x}_i) = 1$ **then**
      $I(\mathbf{x}_i) \leftarrow 1 + min\left\{I(\mathbf{y}) \mid \mathbf{y} \in N_I^{\leftarrow}(\mathbf{x})\right\}$
    end if
  end for
  for backward scan of all pixels $\mathbf{x}_i \in D_I$ **do**
    if $I(\mathbf{x}_i) \neq 0$ **then**
      $I(\mathbf{x}_i) \leftarrow min\left\{I(\mathbf{x}_i), 1 + min\left\{I(\mathbf{y}) \mid \mathbf{y} \in N_I^{\rightarrow}(\mathbf{x}_i)\right\}\right\}$
    end if
  end for
end procedure

However, for reconstructions of cellular materials (especially foams), an Euclidean distance measure is often preferred. Indeed, reconstruction algorithms are considerably affected by the distance used in distance transforms (see Reference [125], pp. 27). In the field of reconstruction of cellular materials, the most natural and most used distance is the Euclidean distance, through the use of Voronoï diagrams (see, e.g., References [140, 55, 149, 93]). Moreover, the Euclidean distance is the simplest one (its associated metric is the identity) and derives from a norm as well from a scalar product. Nevertheless, it should be noted that the *power distance* distance is increasingly used through the *Laguerre tessellation* (see, e.g., References [149, 138, 182, 95, 169]). Though, the computation of the Euclidean distance remains mandatory, as the power distance relies on the Euclidean distance by its very definition. Finally, more "esoteric" distances have been introduced by Sonon et al. [156], with the purpose of using them for describing more precisely foam microstructures. However, the properties of the distances introduced by Sonon et al. are not fully understood yet and need to be further investigated.



(a) General network. (b) 4–connected network. (c) 8–connected network.

FIGURE 2.15: Examples of 2–dimensional digitalisation networks.

Algorithms for the Euclidean distance.

Several efforts have been made for computing approximate or exact Euclidean distances in images. Among them can be found:

- Danielsson's algorithm [37]. This algorithm computes two integers L_1 and L_2 given, respectively the horizontal and vertical path lengths separating two pixels. The Euclidean distance between those two pixels is then given by $\sqrt{L_1^2 + L_2^2}$.
- The chamfer transformations proposed by Borgerfors [24] measure the minimal length of all possible paths between two pixels p and q using a weighted version of the digitalisation network of the image.

  For instance, for 2-dimensional images, by measuring the number L_1 of horizontal and the numbver L_2 of vertical pixels separating p from q, their Euclidean distance is given by $\sqrt{L_1^2 + L_2^2}$. If the digitalisation network is a 8-connected network (see Figure 2.15c) weighted by w_1 for its horizontal and vertical edges and by w_2 for its diagonal edges, the chamfer distances become (see Figure 2.16):

  $$d_{w_1,w_2} = L_2 w_2 + (L_1 - L_2) w_1$$

  For this case, the optimal weights are $w_1 = 1$, $w_2 = 1/\sqrt{2} + \sqrt{\sqrt{2} - 1}$. With respect to the exact Euclidean distance, the error $\|d_e(p,q) - d_{w_1,w_2}(p,q)\|$ of this optimal chamfer distance amounts to approximatively $0.06L_1$. Thus, the error grows when the distance between pixels p and q increases.

  It is possible to reduce this error by complexifying the weighting of the digitalisation networks thanks to masks. However, larger masks do not lead to significant accuracy gain [23] and slow down the computation of the chamfer distance (see Reference [125], pp. 27).

- Many other algorithms as Vincent's algorithms based on chain propagations [175] or Saito and Toriwaki based on manipulations of squared distance [145]. A more comprehensive description and comparisons between different Euclidean DT algorithms has been conducted by Fabbri et al. [46].

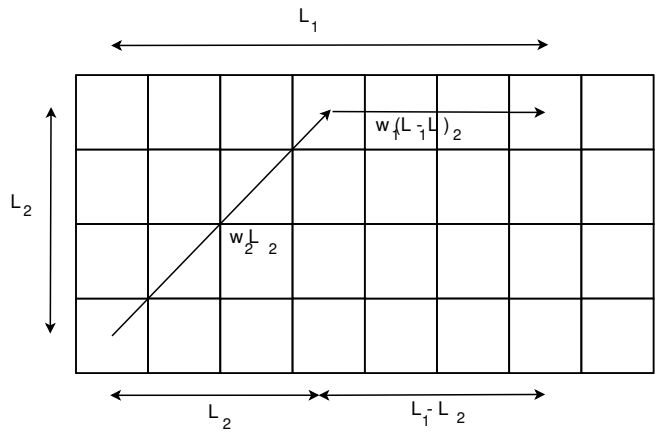


FIGURE 2.16: Schematic example for computing a chamfer distance: weighted shortest path between two pixels along the digitalisation network. Figure inspired from Figure 7 of [24].

Maurer et al. algorithm For the Euclidean distance transform, it has been chosen here to use an algorithm developed by Maurer et al. [111] and for which a parallelised version has been implemented in ITK by Staubs et al. [158]. This algorithm computes efficiently the exact Euclidean distance transform and is linear with respect to the number of pixels in the image. It relies on dimensional reduction and partial Voronoï diagram constructions. The algorithm consists in scanning the image along each of its directions and finding the closest feature pixel for a given pixel. This particularity make the algorithm suitable for streaming a 3D image by slices. Details on how this algorithm works can be found in Appendix B. Figure 2.17 illustrates an Euclidean distance transform computed on a 2D binary image of a foam. It can be seen that local maxima of the distance transform roughly correspond to the centres of cells. In general, unfortunately, local maxima only give a rough first guess of cell centres, and more post-processing is needed for accurately identifying cell centres. This is the topic of the next sections.

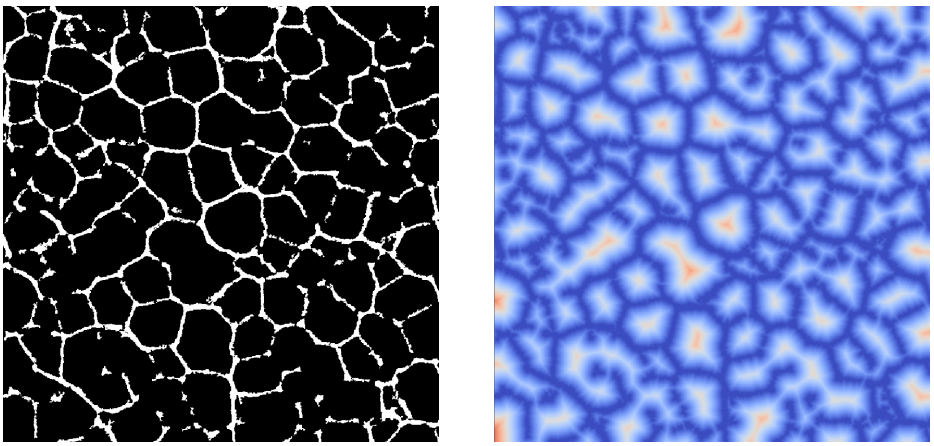
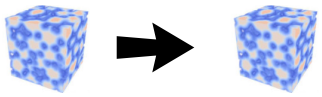


FIGURE 2.17: Original 2D binary image of a foam (left) and Euclidean distance transform using Maurer et al. algorithm [111]. Warmers colours stand for larger distances.

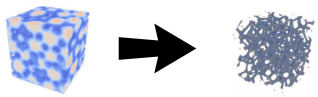
2.3.7 Step 3: Distance post-processing



- **Aim**: smooth a distance transform or perform other post-processing tasks.
- **Input**: gray–level image representing a distance transform.
- **Output**: gray–level image representing the corresponding smoothed distance transform.
- **Streaming**: see Section 3.8

This step is optional. If some post-processing of the distance transform is needed, it can be added here. For instance, a smoothing step may be desirable for avoiding superfluous maxima and oversegmentation [138]. For the datasets used in this thesis, such an optional processing step was not proven necessary and none was used.

2.3.8 Step 4: Local maxima



- **Aim**: given a gray–level image representing the distance transform of a binary foam image, identify local maxima that will be used as seeds for growing ellipsoids.
- **Input**: gray–level image.
- **Output**: gray–level image marked with local maxima.
- **Streaming**: see Section 3.9.

Finding *local maxima* in a gray–scale image is a important basic operation in image analysis for mainly two reasons. Firstly, local maxima (and local minima) often mark relevant distinct objects [18, 155, 33]. Secondly, the ability of finding local maxima (and local minima) can be useful for the development of more advanced algorithms such as *queue-based reconstructions* [142, 162]. Conversely, the presence of superfluous local maxima (or minima) in noisy images can lead to oversegmentation for the watershed transform [155, 149, 138]; which can have been a motivating reason for the development of *H–maxima* (and *H–minima*) transforms [155, 149].

The basic idea of the watershed transformation is the following: let's be a gray-scale image to be seen as the topographical representation of a landscape. Let's drop a water fall on this "landscape". The water will flow towards local minima (a.k.a. "valleys") of the landscape which will constitute "catchment basins". Catchment basins are progressively flooded and, when two different basins reach each-other, a "dam" at their meeting points is "constructed". Figure 2.20 illustrates the concept of watershed transformation seen as a flooding process.

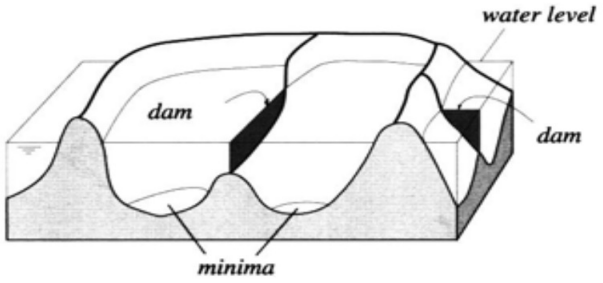


FIGURE 2.20: Gray–scale image seen as the topographic representation of a landscape. The local minima are progressively flooded and constitute "catchment basins". When two catchment basins meet, "dams" separating them are constructed. (Image from P. Soille [155], Figure 9.2).

For the interested reader, some notions related to *mathematical morphology*, and more formal definitions of the watershed and H-maxima transforms are given in Appendix C.

Non-maximum suppression

The technique of *non-maximum suppression* for gray-scale images was first used in the context of edge-thinning in digitalised images [144]. It was then developed and generalised to edge-corner detection and localisation of two–dimensional features [80]. The algorithm used here for the detection of local maxima is a modified version of the algorithm developed by T.Q. Pham [130].

The ITK library [69] provides mainly two filters for finding maxima, namely the *itk::HConvexImageFilter* and the *itk::ValuedRegionalMaximaImageFilter*. The last one from R. Beare and G. Lehman [16] is currently the fastest[12]. Nevertheless, the current implementation of these filters[13] does not allow streaming. For instance, the algorithm of R. Beare and G. Lehman (see Algorithm 6) relies on a flooding approach, where the flooding region cannot be predicted a priori, while the *itk::HConvexImageFilter* relies on a H-convex transformation.

Algorithm 6 R. Beare and G. Lehman algorithm [16] for finding regional maxima inside a grayscale image I.

Require: gray-scale image I.
1: **procedure** BEARELEHMANREGIONALMAXIMA(I)
2: Copy the input image I to an output image J.
3: Visit each pixel of the input image I.
4: **if** The corresponding output value is not maximal (meaning this pixel has not already been visited) then check all the neighbours. **then**
5: **if** Any of the neighbouring gray level are bigger than the current pixel value, then this pixel can not be a regional maxima. **then**
6: Flood fill the region, in the output image, with the same gray level as the current pixel that contains the current pixel, with the minimal value for the pixel type [sic].
7: **end if**
8: **end if**
9: Goto to next input pixel, if any.
10: **return** Output image J.
11: **end procedure**

[12] For ITK version 4.10.1.

[13] Note, though, that the algorithms of R. Beare and G. Lehman computes local maxima, while the *itk::HConvexImageFilter* actually computes *regional* maxima and, therefore, may miss some local maxima.

(a) Original image.

(b) Local maxima computed with the original algorithm.

(c) Local maxima computed with the modified algorithm.

FIGURE 2.21: Comparison of detections of local maxima in a simple gray–scale image (a). Local maxima detected by the original algorithm of T.Q. Pham (b), and by its modified version (c). Note that the original algorithm does not detect plateaus.

Algorithm of T.Q. Pham

The idea of the *scan-line* algorithm of T.Q. Pham is to scan a gray-scale image I of dimension n along a 1D-direction and find local peaks along this direction. Then, the pixels located in the neighbourhoods of the local peaks are scanned and compared against the local peaks in order to determine local maxima.

The *scan-line* algorithm proposed by T.Q. Pham allows streaming using slices that are aligned with the chosen 1D-direction, and then requests only a constant neighbourhood around the peaks. This latest request can easily be handled using the image region negotiation process between filters, in a similar fashion that has been done for the *box filter*.

The original algorithm of T.Q. Pham was initially designed for processing 2-dimensional images and was written in Matlab (see Reference [130]). A modified C++ version of this algorithm for n-dimensional images has been written for the needs of this thesis. The core of the algorithm of T.Q. Pham consisting in scanning lines in 2-dimensional images, the modification for n-dimensional images is straightforward: it only needs to apply the original algorithm to a given 2D slice and then repeat it on the other slices. The modified C++ version has been implemented as a standard streamable ITK filter and, in addition to streaming, plateau detection and the ability to discard saddle points by computing discrete Hessians have been added.

All-in-all, the core of the algorithm of T.Q. Pham remains untouched, while some improvements for getting a streamable C++ version on n-dimensional images and plateaus and saddle points detections were added to it. As a result, the modified version of the algorithm of T.Q. Pham is not faster than the original one (as it performs some additional checks for plateaus and saddle points detections) but it is more general as shown in Figure 2.21.

Description of the modified version of the algorithm of T.Q. Pham. The algorithm searches for local maxima over $(2R+1)^n = \overbrace{(2R+1) \times \ldots \times (2R+1)}^{n}$ neighbourhoods of radius R. Given a grayscale image I of parallelepipedic domain $D_I = [0, D_1] \times \ldots \times [0, D_n]$, a one-dimensional scan-line of I is performed. Let's, without loss of generality, the scan direction be the x-direction (first leading direction).

In order to set notations, let's, for a pixel $\mathbf{x} \in D_I$, write:

- $\tilde{I}_0(\mathbf{x}) = I(x_1, 0, 0, \ldots, 0)$.
- $\tilde{I}_1(\mathbf{x}) = I(x_1, 1, 0, \ldots, 0)$.
- $\tilde{I}_2(\mathbf{x}) = I(x_1, 2, 0, \ldots, 0)$.
- $\ldots$
- $\tilde{I}_{D_1}(\mathbf{x}) = I(x_1, D_1, 0, \ldots, 0)$.
- $\tilde{I}_{D_1+1}(\mathbf{x}) = I(x_1, 0, 1, \ldots, 0)$.
- $\ldots$
- $\tilde{I}_{D_1+D_2}(\mathbf{x}) = I(x_1, 0, D_2, \ldots, 0)$.
- $\ldots$
- $\tilde{I}_{D_1D_2+D_2}(\mathbf{x}) = I(x_1, D_1, D_2, \ldots, 0)$
- $\ldots$
- $\ldots$
- $\tilde{I}_{D_1 \ldots D_n + D_2 \ldots D_n + \ldots + D_n}(\mathbf{x}) = I(x_1, D_1, D_2, \ldots, D_n)$.

For $d \in [0, D_1 \ldots D_n + D_2 \ldots D_n + \ldots + D_n]$ fixed and two consecutive pixels $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(i-1)}$ along the scan line d, let's g be the sign of the finite difference of $\tilde{I}_d$:

$$g : D_I \to \{-1, 0, 1\} \, ; \mathbf{x}^{(i)} \mapsto g(\mathbf{x}(i)) = sgn \left(\tilde{I}_d(\mathbf{x}^{(i)}) - \tilde{I}_d(\mathbf{x}^{(i-1)}) \right). \tag{2.6}$$

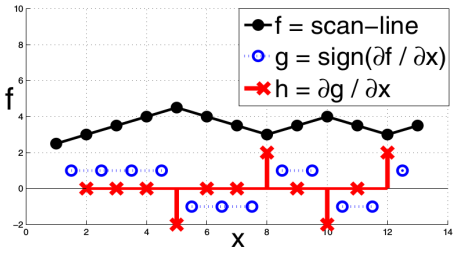
Moreover, let's h be the finite difference of g:

$$h : D_I \to \{-2, -1, 0, 1, 2\} \, ; \mathbf{x}^{(i)} \mapsto h(\mathbf{x}^{(i)}) = g(\mathbf{x}^{(i)}) - g(\mathbf{x}^{(i-1)}). \tag{2.7}$$

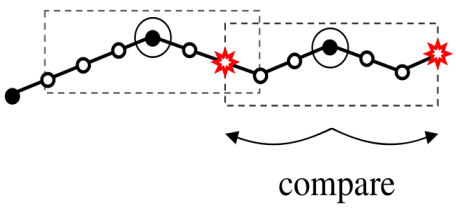
Where:

- $h(\mathbf{x}^{(i)}) = -2$ at local peaks with their corresponding neighbourhood N_I inside D_I ($N_I \subset D_I$).
- $h(\mathbf{x}^{(i)}) = -1$ at other local peaks.
- $h(\mathbf{x}^{(i)}) = +2$ at other local throughs with their corresponding neighbourhood N_I inside D_I ($N_I \subset D_I$).
- $h(\mathbf{x}^{(i)}) = +1$ at other local throughs.
- $h(\mathbf{x}^{(i)}) = 0$ elsewhere.

Figure 2.22a sketches an example of the result obtained with functions g and h along a scan line. A first set S_1 of maxima candidates is computed by scanning the image I along the x-directions and taking pixels $\mathbf{x}$ with strictly negative corresponding Hessian ($h(\mathbf{x}) < 0$). Then a second set S_2 of maxima candidates is obtained by decimating the set S_1. Figure 2.22b illustrates the decimation process where the maxima candidates of set S_1 are selected against their $2(2R+1)$ neighbours along the current scan line. Pixels on a downward slope (hollow pixels) can be immediately ruled out as maxima candidates. Red star–shaped pixels, on the other hand, need to be compared against the circled pixels as they appear after a slope change in their $2(2R+1)$ neighbourhood.



(a) Computation of the sign of the discrete derivative and its Hessian along a line of pixels.



(b) Comparison of the maxima candidates against their $(2R+1)$ $(R = 1)$ neighbours along the scan direction. Hollow pixels does not need to be compared to maxima candidates (circled pixels) as they are located on a downward slope.

FIGURE 2.22: Sketch of the scan-line algorithm of T.Q. Pham along a one–dimensional line of pixels. (Images from Figure 3 of the article of T.Q. Pham [130].)

Eventually, the maxima candidates of set S_2 are compared against their n-dimensional neighbourhoods $N_I(\mathbf{x}) = [x_1 - (2R+1), x_1 + (2R+1)] \times \ldots \times [x_n - (2R+1), x_n + (2R+1)]$ (see Algorithm 10).

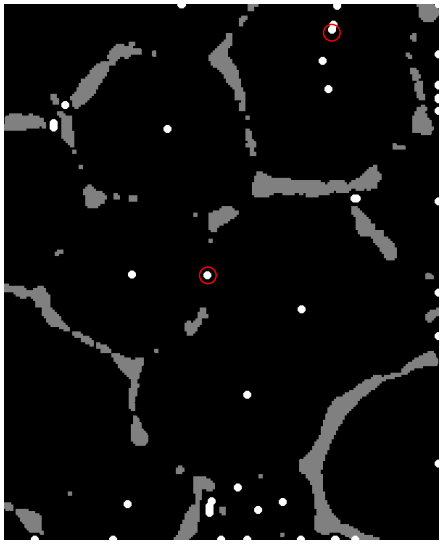
Note on boundaries If the pixel $\mathbf{x}$ lies on a boundary of the image I, part of its neighbourhood $N_I(\mathbf{x})$ does not exist anymore. In order to detect maxima located on the image boundaries, the image is padded over a radius $(2R+1)$ with zeros (the minimum possible value for an unsigned distance).

Hessian-based maxima selection

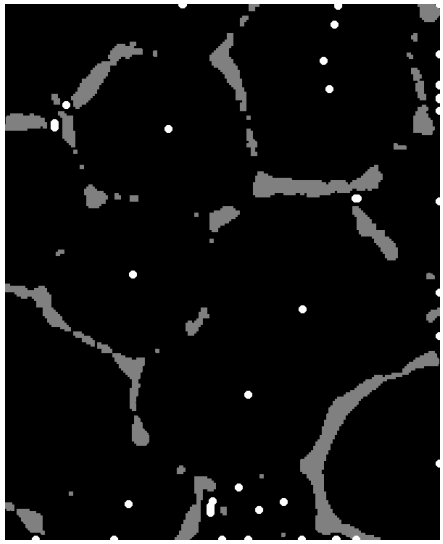
The local maxima found by the algorithm of T.Q. Pham may not all be *genuine* local maxima. Some of these maxima may be saddle points[14]. As the aim is to locate cell centres thanks to the maxima of the image I obtained by an Euclidean distance transform (definition B.2), these saddle points have to be avoided. Indeed, saddle points are located at missing cell wall regions and not at cell centres (see Figure 2.23a).

Therefore, it is needed to detect and discard saddle points. This can be done by looking at the eigenvalues of the Hessian of image I evaluated at each maximum.

[14]For a function $f : \mathcal{R} \to \mathcal{R}; \mathbf{x} \mapsto f(\mathbf{x})$, a point $\mathbf{y} \in dom(f)$ is a *saddle point* if at least one eigenvalue of the Hessian of f evaluated at $\mathbf{y}$ has a sign opposite to another, different, non-zero eigenvalue.



(a) maxima (represented as white disks) and saddle points (represented as white disks circled with red) computed on a 2–dimensional thresholded image of a cellular material.



(b) maxima (represented as white disks) filtered from saddle points and computed on a 2-dimensional thresholded image of a cellular material. Parameters used in algorithm 7: $R_{user} = 5$, $\sigma = 3$, $eigenTol = 10^{-5}$, $tol = 10^{-4}$.

FIGURE 2.23: Filtering of saddle points by computing the eigenvalues of the Hessian evaluated at the maxima candidates.

If all the eigenvalues are negative, then the considered maximum is a *genuine* one, otherwise it is a saddle point and it should be discarded.

In practice, for noisy images, some tolerance $tol > 0$ should be added on the criterion of the negativity of the eigenvalues of the Hessian. In her master–thesis [99], A. Lopez-Reina suggest to use the following criteria:

1. If all eigenvalues $< -tol$, then accept the associated maximum.
2. If $-tol <$ some eigenvalues $< tol$, and the other $< -tol$, then accept the associated maximum.
3. If there is some eigenvalues $\lambda_i > tol$, and if λ_{min} is the smallest eigenvalue such that $\lambda_{min} < 0$, then accept the associated maximum if:

$$\lambda_i \leq tol\ |\lambda_{min}| \tag{2.8}$$

   Reject it otherwise.

The above criteria can be summarised as follow: if λ_{max} is the highest positive eigenvalue (if any) and λ_{min} is the smallest eigenvalue, then:

1. If there is no λ_{max}: accept the associated maximum.
2. Elsewhere, if $!(\lambda_{max} > tol|\lambda_{min}|)$ && $(\lambda_{min} < 0)$ accept the associated maximum, else reject it.

Algorithm 11 describes the procedure for discarding saddle-points within a given tolerance. Figure 2.23b shows filtered maxima found from the same image as Figure 2.23a.

Computation of the Hessians

The selection algorithm 11 of A. Lopez-Reina of the local maxima requires the computation of the corresponding Hessians. Computing these Hessians requires the computations of derivatives in discrete images, which is an ill-posed problem [101]. Derivatives in discrete images are usually computed as *convolutions* of the considered discrete image with a given predefined operator which performs a prior smoothing on the image. One method to compute such prior smoothing is to resort to a *scale-space representation* of the considered image.

The truncated discrete scale-space representation and its Hessian - amongs other discrete derivatives of the scale-space representation - have been implemented by I. Macia [101] in the ITK library and its used in the Ellipsoidal Model for computing the local Hessians à maxima's locations. The following describes in details how scale-space representations are defined, how to compute them in the continuous and discrete case, and, in particular, how to truncate their discrete developments by computing a correct neighbourhood radius $R_{Hessian}$ around each local maxima in order to ensure a correct calculation of the local Hessian within a given tolerance.

Preliminary definitions

Definition 2.1. The modified Bessel function of integer order $I_n : \mathcal{R} \to \mathcal{R}; x \mapsto I_n(x)$ can be defined as:

$$I_n(x) = (-1)^n J_n(ix). \tag{2.9}$$

Where $n \in \mathcal{Z}$, $i \in \mathcal{C}$ is the imaginary number such that $i^2 = -1$ and $J_n = \frac{1}{\pi}\int_0^{\pi} cos(nt - x\, sin(t))\, dt$ is the Bessel function of the first kind.

Definition 2.2. Given two integrable functions $f : \mathcal{R}^n \to \mathcal{R}^p$ and $g : \mathcal{R}^n \to \mathcal{R}^p$, their *convolution* over a range $[a, b]$, with $a \in \mathcal{R}$ and $b \in \mathcal{R}$, is:

$$(f * g)(\mathbf{x}) = \int_a^b f(\mathbf{y})\, g(\mathbf{x} - \mathbf{y})\, d\mathbf{y} = \int_a^b g(\mathbf{y})\, f(\mathbf{x} - \mathbf{y})\, d\mathbf{y}. \tag{2.10}$$

Definition 2.3. A *scale–space kernel* is a function $g : \mathcal{R}^n \times \mathcal{R}^+ \backslash \{0\} \to \mathcal{R}^p; (\mathbf{x}; t) \mapsto g(\mathbf{x}; t)$ satisfying the following axioms [10]:

1. $\forall \mathbf{x} \in \mathcal{R}^n$ and $\forall t > 0$, there exists a function $h : \mathcal{R}^n \to \mathcal{R}^p$ such that $g(\mathbf{x}; t) = t\, h(\mathbf{x}t)$.
2. $\forall \mathbf{x} \in \mathcal{R}$, the kernel g is symmetrical : $g(-\mathbf{x}; t) = g(\mathbf{x}; t)$.
3. $\forall t > 0$, $\int_{-\infty}^{+\infty} g(\mathbf{x}; t)\, d\mathbf{x} = 1$.
4. $\exists p \in \mathcal{Z}$ such that $\frac{\partial^{2p} h(\mathbf{x})}{\partial \mathbf{x}_i^p \mathbf{x}_j^p} \neq 0$; $i, j = 1, \ldots, n$.
5. Given an admissible function $f : \mathcal{R}^n \to \mathcal{R}^p$, the number of maxima (respectively minima) of the convolution $g(\mathbf{x}; t) * f(\mathbf{x})$ increases (respectively decreases) monotonically with t.

J.Badaud et al. [10] showed that, for the above set of axioms, the scale-space kernel is unique and is the Gaussian kernel:

$$g(\mathbf{x};t) = \frac{1}{(2\pi t)^{n/2}}\, exp\left(-\frac{\|\mathbf{x}\|^2}{2t}\right).$$

Definition 2.4. A *scale-space representation* of a function $f : \mathcal{R}^n \to \mathcal{R}^p$ is a function $L : \mathcal{R}^n \times \mathcal{R}^+\backslash\{0\} \to \mathcal{R}^p; (\mathbf{x};t) \mapsto L(\mathbf{x};t)$ of the form:

$$L(\mathbf{x};t) = (g * f)(\mathbf{x};t). \tag{2.11}$$

Where the function $g : \mathcal{R}^n \times \mathcal{R}^+\backslash\{0\} \to \mathcal{R}^p; (\mathbf{x};t) \mapsto g(\mathbf{x};t)$ is a scale-space kernel.

Explicitly, the scale–space representation of an integrable function $f : \mathcal{R}^n \to \mathcal{R}^p$ is:

$$L(\mathbf{x};t) = (g * f)(\mathbf{x};t) = \int_{-\infty}^{+\infty} \frac{1}{(2\pi t)^{n/2}}\, exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2t}\right) f(\mathbf{y})\, d\mathbf{y}. \tag{2.12}$$

Intuitively, $L(\mathbf{x};t)$ corresponds to a "blurred" version of the function $f(\mathbf{x})$, where t is a "blurring" parameter. For $t \to 0$, the "blur" disappears and $lim_{t\to 0}\, L(\mathbf{x};t) = f(\mathbf{x})$, while the "blur" increases with increasing values of t.

If the derivatives of the function f are numerically ill-conditioned (i.e. $\exists \mathbf{x} \in dom(f')$, $\exists M > 0 \mid \forall \epsilon > 0$, $\exists \mathbf{y} \in dom(f')$, $|f'(\mathbf{x}+\epsilon\mathbf{y}) - f'(\mathbf{x})| \geq M$) or not continuous, the derivatives of its scale–space representation $L(\mathbf{x};t)$ present a better behaviour, depending on the value of the parameter t.

Moreover, thanks to the properties of the Gaussian kernel, it is possible to precompute the Hessian of a scale-space representation [101]:

$$\begin{aligned}
\frac{\partial^2 L(\mathbf{x};t)}{\partial x_i \partial x_j} &= \frac{\partial^2}{\partial x_i \partial x_j}\,(g * f)\,(\mathbf{x};t) \\
&= \left(\frac{\partial^2 g(\mathbf{x};t)}{\partial x_i \partial x_j}\right) * f(\mathbf{x}) \\
&= g(\mathbf{x};t) * \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right);\ i,j = 1,\ldots,n
\end{aligned} \tag{2.13}$$

As a consequence, computing the Hessian of the scale-space representation $L(\mathbf{x};t)$ of the function $f(\mathbf{x})$ amounts to compute a "blurred" version of the original Hessian $\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j};\ i,j = 1,\ldots,n$.

Hessian of a discrete image If the above considered function $f(\mathbf{x})$ is a discrete image $I(\mathbf{x})$, a discrete version of the Gaussian kernel is needed. However, as pointed by T. Lindeberg [97], a naïve discretisation of the Gaussian kernel will not, in general, satisfy the scale-space axioms. Furthermore, the interesting property (2.13) will in general be lost.

However, T. Lindeberg [97] proved that a discretisation of the Gaussian kernel preserving the scale-space and the property (2.13) exists for any dimension $n \in \mathcal{N}\backslash\{0\}$. For the one-dimensional case, he suggested to use the following discretisation:

$$T(m;t) = exp(-t)\ I_m(t). \tag{2.14}$$

Where the $I_m(t)$ are the modified Bessel function of integer order.

The corresponding one-dimensional discrete scale-space representation $L_d : \mathcal{Z} \times \mathcal{R}^+\backslash\{0\}; (x;t) \mapsto L_d(x;t)$ for a one-dimensional discrete image $I(x)$ is given by:

$$L_d(x;t) = \sum_{m=-\infty}^{+\infty} T(m;t)\ I(x-m). \tag{2.15}$$

For higher dimensions, the corresponding discrete scale-space representation $L_d : \mathcal{Z}^n \times \mathcal{R}^+\backslash\{0\}; (\mathbf{x};t) \mapsto L_d(\mathbf{x};t)$ for a n-dimensional discrete image $I(\mathbf{x})$ of infinite domain $D_I = \mathcal{Z}^n$ is given by:

$$L_d(\mathbf{x};t) = \sum_{m_1=-\infty}^{+\infty} T(m_1;t) \ldots \sum_{m_n=-\infty}^{+\infty} T(m_n;t)\ I(x_1 - m_1, \ldots, x_n - m_n). \tag{2.16}$$

The discretisation of the Hessian of the scale-space representation $L(\mathbf{x};t)$ is, in turn, given by:

$$\begin{aligned}
\frac{\partial^2 L(\mathbf{x};t)}{\partial x_i \partial x_j} &= \left(\frac{\partial^2 g(\mathbf{x};t)}{\partial x_i \partial x_j}\right) * I(\mathbf{x}) \text{ (Property 2.13).} \\
&= \left(\frac{x_i x_j - \delta_{ij} t}{t^2} g(\mathbf{x};t)\right) * I(\mathbf{x}) \\
&\downarrow \text{ discretise} \\
&= \sum_{m_1=-\infty}^{+\infty} \ldots \sum_{m_n=-\infty}^{+\infty} \frac{m_i m_j - \delta_{ij} t}{t^2} T(m_1;t) \ldots T(m_n;t) I(x_1 - m_1, \ldots, x_n - m_n) \\
&= \frac{\partial^2 L_d(\mathbf{x};t)}{\partial x_i \partial x_j} \\
&(i,j = 1, \ldots, n)
\end{aligned} \tag{2.17}$$

Where $\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ elsewhere} \end{cases}$, is the Kronecker delta.

In practice, the domain D_I of the discrete image I is not infinite. Also, the summations given for the discrete scale-space representation $L_d(\mathbf{x};t)$ 2.16 and for the discrete Hessian of the scale-space representation $\frac{\partial^2 L_d(\mathbf{x};t)}{\partial x_i \partial x_j};\ i,j = 1,\ldots,n$ 2.17 are truncated.

$$L_d^{trunc}(\mathbf{x};t) = \sum_{m1=-M_1}^{M_1} T(m_1;t)\ldots \sum_{m_n=-M_N}^{M_n} T(m_n;t) I(x_1-m_1,\ldots,x_n-m_n). \quad (2.18)$$

And

$$\frac{\partial L_d^{trunc}(\mathbf{x};\mathbf{t})}{\partial x_i \partial x_j} = \sum_{m_1=-M_1}^{M_1} \ldots \sum_{m_n=-M_n}^{M_n} \frac{m_i m_j - \delta_{ij} t}{t^2} T(m_1;t)\ldots T(m_n;t) \times I(x_1-m_1)\ldots I(x_n-m_n); i,j = 1,\ldots,n. \quad (2.19)$$

The bounds $M_1,\ldots,M_n$ are computed such that the discretisation error, defined as the "difference between the area under the discrete Gaussian and the area under the continuous Gaussian", is below a prescribed value[15].

Remark: boundary condition The truncated discrete Hessian of the scale space representation given in equation 2.19 requires for each pixel $\mathbf{x}$ in the domain D_I of the image I, the values of the surrounding pixels in a neighbourhood $N_I(\mathbf{x})$ centred on $\mathbf{x}$ and of size $[-M_1,M_1]\times\ldots\times[-M_n,M_n]$. If $\mathbf{x}$ is located near a boundary of D_I, some pixels $\mathbf{y}\in N_I(\mathbf{x})$ of the requested neighbourhood may *overflow* the domain D_I: i.e. $\exists \mathbf{y}\in N_I(\mathbf{x}) \mid \mathbf{y}\notin D_I$. For such overflowing pixels $\mathbf{y}$, their values are simply set to zero, so they do not contribute to the computation of the convolution.

Neighbourhood computation for the Hessians The selection algorithm 11 of A. Lopez–Reina of the local maxima requires the computation of corresponding Hessians. The computations of these Hessians require themselves the knowledge of pixel values around given neighbourhoods of the local maxima candidates. The extent of these neighbourhood is an integer constant and depends on a Gaussian and its first and second derivatives:

$$g:\mathcal{R}^+\to\mathcal{R}^+; r\mapsto g(r;\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}}\, exp\left(-\frac{r^2}{2\sigma^2}\right). \quad (2.20)$$

$$\frac{\partial g}{\partial r}:\mathcal{R}^+\to\mathcal{R}; r\mapsto \frac{\partial g}{\partial r}(r;\sigma) = \frac{-r}{\sigma^2}\, g(r;\sigma). \quad (2.21)$$

$$\frac{\partial^2 g}{\partial r^2}:\mathcal{R}^+\to\mathcal{R}; r\mapsto \frac{\partial^2 g}{\partial r^2}(r;\sigma) = \frac{r^2-\sigma^2}{\sigma^4}\, g(r;\sigma). \quad (2.22)$$

Where n is the image dimension, $r = \|\mathbf{x}-\mathbf{y}\|$ is the distance between the considered local maximum candidate $\mathbf{x}\in\mathcal{R}^n$ and a pixel $\mathbf{y}\in\mathcal{R}^n$ in its neighbourhood, and $\sigma = \sqrt{t} > 0$ is a given standard deviation.

[15] Citation from ITK's documentation [69], pp. 103.

Note Given the truncated discrete Hessian of the scale-space representation 2.19, the class *itk::DiscreteHessianGaussianImageFunction* of the ITK library computes the convolution region $[-M_1, M_1] \times \ldots \times [-M_n, M_n]$. However, this class does not offer a public access to these values, but only a default threshold value through its public function *GetMaximumKernelWidth()* which is pessimistic[16]. It is in principle possible compute some optimal values $M_1, \ldots, M_n$ by running the algorithm used by the class *itk::DiscreteHessianGaussianImageFunction* several times and minimising some error criterion. However, this algorithm is relatively complex and running it several times requires a fair computational effort.

Instead, the method suggested in this thesis is fast (logarithmic in time) and simple. It consists in computing a radius bound $R \geq max(M_1, \ldots, M_n)$ as small as possible for the needed neighbourhood such that the contribution to the truncated discrete Hessian of the scale–space function 2.19 of pixels located outside the neighbourhood of radius R is negligible. Then, the algorithm used by the class *itk::DiscreteHessianGaussianImageFunction* is only required to be run once with the found radius R as threshold.

From a given pixel $\mathbf{x}$, pixels in the neighbourhood of $\mathbf{x}$ are convolved with the second derivative of the Gaussian $\frac{\partial^2 g}{\partial r^2}$. This second derivative tends towards zero for $r \to \infty$, as the term $g(r;\sigma)$ tends exponentially towards zero by positive values for $r \to \infty$, while the term $\frac{r^2-\sigma^2}{\sigma^4}$ tends only quadratically towards infinity for $r \to \infty$. As a consequence, neighbourhood pixels $\mathbf{y}$ far enough from the considered pixel $\mathbf{x}$ do not contribute significantly to the convolution and may be discarded. Mathematically speaking:

$$\forall \epsilon > 0,\ \exists R > 0 \mid \forall \mathbf{x} \in D_I,\ \left| \int_{\Omega=\{\mathbf{y}\in D_I \mid (\mathbf{x}\pm\mathbf{y})\in D_I\}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} - \int_{\Omega_{R^-}=\{\mathbf{y}\in D_I \mid \|\mathbf{x}-\mathbf{y}\|\leq R\}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} \right| \leq \epsilon \quad (2.23)$$

Where I is an image, D_I its associated domain, and R is the radius of the neighbourhood to consider.

As:

$$\Omega = \{\mathbf{y} \in D_I \mid (\mathbf{x} \pm \mathbf{y}) \in D_I\} = \{\mathbf{y} \in D_I \mid \|\mathbf{x}-\mathbf{y}\| \leq R\} \cup \{\mathbf{y} \in D_I \mid \|\mathbf{x}-\mathbf{y}\| > R\} = \Omega_{R^-} \cup \Omega_{R^+} \quad (2.24)$$

And, trivially:

$$\Omega_{R^-} \cap \Omega_{R^+} = \emptyset \quad (2.25)$$

[16] In ITK 4.10.1 this value is set by default to 30 pixels.

One have:

$$\int_{\Omega} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} = \int_{\Omega_{R^-}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} + \int_{\Omega_{R^+}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} \tag{2.26}$$

And the inequality 2.23 becomes:

$$\forall \epsilon > 0,\ \exists R > 0 \mid \forall \mathbf{x} \in D_I,\ \left| \int_{\Omega_{R^+}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} \right| \leq \epsilon \tag{2.27}$$

The problem to solve here, is thus to find a neighbourhood radius $R > 0$ such that the inequality 2.27 is satisfied. Moreover, a value of R as small as possible should be preferred. Indeed, a high value of R implies the request of a large neighbourhood during the ITK's pipeline execution, which should be avoided in order to save RAM memory usage. For a given $\epsilon > 0$, the problem to solve is thus:

$$\begin{cases} min_{R>0}\ R \\ \text{s.t. } \left| \int_{\Omega_{R^+}} I(\mathbf{y}) \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} \right| \leq \epsilon,\ \forall \mathbf{x} \in D_I \end{cases} \tag{2.28}$$

In the special case where I is an image obtained from a distance transform, I is bounded:

$$\exists M > 0 \mid \forall \mathbf{y} \in D_I,\ I(\mathbf{y}) \leq M. \tag{2.29}$$

For a parallelepipedic domain and an Euclidean distance transform, M is, at worst, the length of the largest diagonal of the domain (this is the maximum possible Euclidean distance between two pixels in the image). Here M is simply approximated by a simple analytical upper bound.

In that case, the problem 2.28 becomes:

$$\begin{cases} min_{R>0}\ R \\ \text{s.t. } \left| \int_{\Omega_{R^+}} \frac{\partial^2 g}{\partial r^2}(\|\mathbf{x}-\mathbf{y}\|;\sigma)\, d\mathbf{y} \right| \leq \frac{\epsilon}{M},\ \forall \mathbf{x} \in D_I \end{cases} \tag{2.30}$$

Which is equivalent to the problem:

$$\begin{cases} min_{R>0} R \\ \text{s.t. } \left| \int_{R}^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\, dr \right| \leq \frac{\epsilon}{M} \end{cases} \tag{2.31}$$

Using the definition of function $g(r;\sigma)$ (2.20) and $\frac{\partial g}{\partial r}(r;\sigma)$ (2.21):

$$\int_{R}^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\, dr = \left[\frac{\partial g}{\partial r}(r;\sigma) \right]_{R}^{+\infty} = \frac{R}{\sigma^2}\, g(R;\sigma) \tag{2.32}$$

For $R \in \mathcal{R}^+$, the integral $\int_{R}^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\, dr$ is always positive, because $exp\left(-\frac{R^2}{2\sigma^2}\right) > 0$. Furthermore, this integral is strictly decreasing for $R > \sigma$.

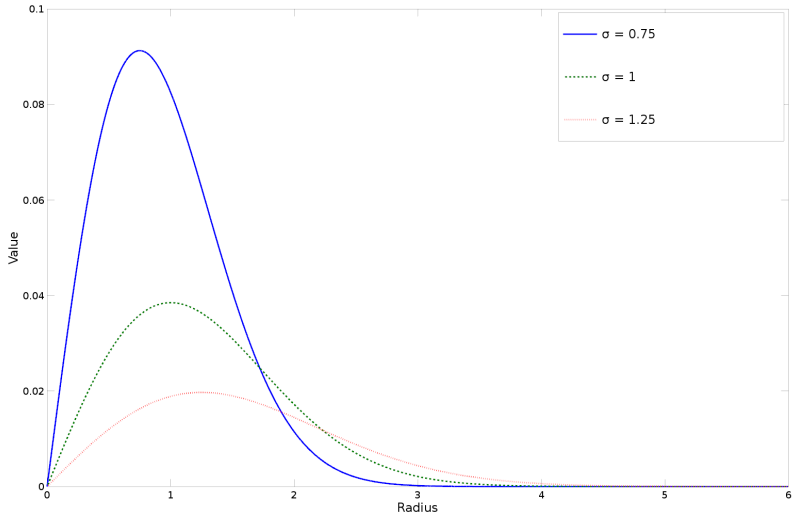


FIGURE 2.24: Values on the interval $[0,6]$ of $G(R;\sigma) = \int_R^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\,dr$ for some values of σ, where $g(r;\sigma) = \frac{1}{(2\pi\sigma^2)^{3/2}}\,exp\left(-\frac{r^2}{2\sigma^2}\right)$ is the 3–dimensional Gaussian of zero mean. For a radius of $R = 4$ pixels, one has: $G(4;0.75) \approx 5.34\ 10^{-7}$, $G(4;1) \approx 8.52\ 10^{-5}$ and $G(4;1.25) \approx 6.22\ 10^{-4}$.

Figure 2.24 illustrates the behaviour of this integral for some values of σ, and qualitatively shows that the values of this integral can be neglected for sufficient big values of R with respect to any finite (strictly positive) value of σ. Quantitatively, using equation 2.22, it may observed that its derivative with respect to R is strictly negative for $R > \sigma$:

$$\frac{\partial}{\partial R}\int_R^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\,dr = \frac{\partial}{\partial R}\left(\frac{R}{\sigma^2}\,g(R;\sigma)\right) = \frac{\sigma^2 - R^2}{\sigma^4}\,g(R;\sigma) \tag{2.33}$$

As a consequence:

$$\left|\int_R^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\,dr\right| = \int_R^{+\infty} \frac{\partial^2 g}{\partial r^2}(r;\sigma)\,dr = G(R;\sigma) \tag{2.34}$$

Where the primitive $G(R;\sigma)$ is explicitly given by the right–hand side of equation 2.32.

Furthermore, as $lim_{z\to+\infty}\, exp(-z^2) = 0$:

$$lim_{R\to+\infty}\, G(R;\sigma) = 0 \tag{2.35}$$

Thus, the problem 2.31 can be stated as:

$$\begin{cases} min_{R>0}\, R \\ \text{s.t. } G(R;\sigma) \leq \frac{\epsilon}{M} \end{cases} \tag{2.36}$$

TABLE 2.5: Parameters of the *Local maxima filter*.

Parameter	Typical value range	Description
R_{user}	$[3, 10]$	Search radius for genuine maxima.
σ	$[1, 5]$	Width of the Gaussian kernel for the Hessian.
$eigenTol$	$[10^{-5}, 10^{-2}]$	Tolerance for criterion on eigenvalues (algo. 11).
tol	$[10^{-5}, 10^{-2}]$	Tolerance for the Hessian radius search (algo. 12).

The problem 2.36 can be solved using a simple bisection method on the function:

$$G(R;\sigma) - \frac{\epsilon}{M} \tag{2.37}$$

Indeed, this function is continuous (combination of continuous functions) and its limit at infinity is $\left(-\frac{\epsilon}{M}\right)$ (consequence of 2.35). The bisection method can thus be applied on the interval $[0, b]$ provided that $G(0;\sigma) = \frac{\sqrt{2\pi}}{(2\pi)^{n/2}\sigma^{n+1}} > \frac{\epsilon}{M}$ and $G(b;\sigma) < \frac{\epsilon}{M}$.

For reasonable values of σ, the interval $[0, D]$, where D is the length of the larger diagonal of the parallelepipedic domain D_I, is a suitable interval. Indeed, $G(0;\sigma)$ is the global maximum[17] of the function $G(R;\sigma)$ over $\mathcal{R}^+$, while the ball $B(\mathbf{x}; R)$ contains the whole image domain I_D for all $\mathbf{x} \in I_D$. The largest possible interval to consider is thus $[0, D]$. Algorithm 12 shows the procedure for solving the problem 2.36.

Remark about the algorithm 12 The stopping condition at line 14 in algorithm 12 is not the usual stopping condition used for a classic bisection algorithm. This is due by the fact that the aim is not to find precisely the root of the function $G(R;\sigma) - \frac{\epsilon}{M}$, but rather to find an integer radius value $R \in \mathcal{N}$ such that the function $G(R;\sigma)$ is smaller than the quantity $tol = \frac{\epsilon}{M}$. As a consequence, if the difference between two consecutive radii values R_{i-1} and R_i is smaller than one - a.k.a $|R_{i-1} - R_i| < 1$ - and are such that $G(R_{i-1};\sigma) - tol > 0$ and $G(R_i;\sigma) - tol < 0$, then $ceil(R_i)$[18] is a suitable integer radius value. Indeed, $ceil(R_i)$ is large enough for the function $G(R_i;\sigma)$ to be negligible, while minimising the extent of the neighbourhood regions over which the Hessians have to be computed. Table 2.5 shows the parameters needed by the modified version of the algorithm 7 of T.Q. Pham.

[17] The functions $exp(-z^2)$ and $erfc(z)$ are strictly decreasing over $\mathcal{R}^+$.

[18] The function $ceil : \mathcal{R} \to \mathcal{Z}; x \mapsto ceil(x)$ maps a real number to its nearest upper integer.

Algorithm 7 Modified version of the algorithm of T.Q. Pham [130]. Given a gray-scale image I of parallelepipedic domain D_I and a provided radius R_{user}, computes the local maxima of I over a neighbourhood of radius R_{user}. Optionally, selects the local maxima based on the eigenvalues of their respective Hessians.

Require: I gray-scale image.
Require: R_{user} radius of the neighbourhood.
Require: σ (optional) width of the Gaussian kernel (see eq. 2.20).
Require: *eigenTol* (optional) tolerance for the eigenvalues (see algo. 11).
Require: *tol* (optional) tolerance for Hessian radius search (see algo. 12).

procedure LOCALMAXIMA(I, R_{user}, σ, *eigenTol*, *tol*)
  if *sigma*, *eigenTol* and *tol* exists **then**
    $D \leftarrow$ biggest diagonal length of D_I.
    $R_{Hessian} \leftarrow$ INTEGRALOFGAUSSIANSECONDDERIVATIVEBISECTION(0, D, σ, *tol*) ▷ See algo 12.
  end if
  $Maxima \leftarrow \emptyset$.
  ▷ Compute sign of first derivative.
  $G \leftarrow \emptyset$
  for pixel line $\tilde{I}_d$, 1D–restriction of I along the x–direction **do**
    for $\mathbf{x}^{(k)} \in dom(\tilde{I}_d)$ **do**
      $G \leftarrow G \cup \left\{ g(\mathbf{x}^{(k)}) \right\}$ ▷ See equ. 2.6.
      ▷ If $\mathbf{x}^{(k-1)} \notin dom(\tilde{I}_d)$, then $\tilde{I}_d(\mathbf{x}^{(k-1)}) =$ min. pixel value.
    end for
    ▷ Find 1D-peaks by computing discrete Hessian h from g.
    $Peaks \leftarrow \emptyset$, $H \leftarrow \emptyset$, $j \leftarrow 0$.
    for $\mathbf{x}^{(k)} \in dom(\tilde{I}_d)$ **do**
      $H \leftarrow H \cup \left\{ h(\mathbf{x}^{(k)}) \right\}$ ▷ See equ. 2.7.
      if $h(\mathbf{x}^{(k)}) == -2 \,||\, h(\mathbf{x}^{(k)}) == -1$ **then**
        $Peaks \leftarrow Peaks \cup \left\{ \left(j, x_2^{(k)}, \ldots, x_n^{(k)} \right) \right\}$.
      end if
      $j \leftarrow j + 1$
    end for
    ▷ Loop over peaks.
    $l \leftarrow$ number of pixels along x–direction.
    $skip \leftarrow \overbrace{(false, \ldots, false)}^{l}$.
    for $\mathbf{p}^{(k)} \in Peaks$ **do**
      if $skip(k)$ **then** continue ▷ Go to the next peak.
      end if

Algorithm 8 Modified algorithm of T.Q. Pham (continued).

▷ Test the current peak against its $(2R_{user}+1)$ neighbourhood.
▷ Right side.
$not1DMaximum \leftarrow false$.
for $j_1 = k, \ldots, k + R_{user}$ **do**
  if $H(j_1) == 0$ **then** break ▷ We are on a slope.
  end if
end for
▷ Test if the slope is going upwards.
$\mathbf{y} \leftarrow \mathbf{p}^{(k)}$, $y_1 \leftarrow j_1$.
for $j_2 = j_1, \ldots, k + R_{user}$ **do**
  if $\tilde{I}_d(\mathbf{p}^{(k)}) < \tilde{I}_d(\mathbf{y})$ **then** ▷ Current peak not a 1D-local maximum.
    $not1DMaximum \leftarrow true$.
    break.
  end if
  $skip[j_2] \leftarrow true$ ▷ Skip future smaller pixel values.
end for
if $not1DMaximum$ **then** continue ▷ Go to the next peak.
end if
▷ Left side.
for $j_1 = k - 1, \ldots, k - R_{user}$ **do**
  if $H(j_1) == 0$ **then** break ▷ We are on a slope.
  end if
end for
▷ Test if the slope is going upwards.
$\mathbf{y} \leftarrow \mathbf{p}^{(k)}$, $y_1 \leftarrow j_1$.
for $j_2 = j_1, \ldots, k - R_{user}$ **do**
  if $\tilde{I}_d(\mathbf{p}^{(k)}) < \tilde{I}_d(\mathbf{y})$ **then** ▷ Current peak not a 1D–local maximum.
    $not1DMaximum \leftarrow true$.
    break.
  end if
  ▷ No $skip$ here, as previous pixels already scanned.
end for
if $not1DMaximum$ **then** continue ▷ Go to the next peak.
end if

Algorithm 9 Modified algorithm of T.Q. Pham (continued).

▷ Test if selected 1D–maxima are true nD–maxima.
$N_I(\mathbf{p}^{(k)}) \leftarrow [p_1^{(k)} - R_{user}, p_1^{(k)} + R_{user}] \times \ldots \times [p_n^{(k)} - R_{user}, p_n^{(k)} + R_{user}]$.
if IsMaximum(I, $\mathbf{p}^{(k)}$, $N_I(\mathbf{p}^{(k)})$) **then** ▷ Algo. 10.
  if $R_{Hessian}$ exists **then**
    $H_k \leftarrow$ Hessian of the discrete scale–space representation of I evaluated at $\mathbf{p}^{(k)}$ with $t = \sigma^2$ and $M_1, \ldots, M_n = R_{Hessian}$ ▷ See equ. 2.19.
    $\Lambda_k \leftarrow$ eigenvalues of the Hessian H_k.
    if SelectMaximumOnHessianEigenvalues(Λ_k, *eigenTol*) **then** ▷ See algo. 11.
      $Maxima \leftarrow Maxima \cup \left\{\mathbf{p}^{(k)}\right\}$.
    end if
  else
    $Maxima \leftarrow Maxima \cup \left\{\mathbf{p}^{(k)}\right\}$.
  end if
  ▷ Check for plateau in the $(2R_{user} + 1)$ neighbourhood.
  $\mathbf{y} \leftarrow \mathbf{p}^{(k)}$.
  for $j = k, \ldots, l$ **do**
    $y_1 \leftarrow j$.
    if $\tilde{I}_d(\mathbf{p}^{(k)}) == \tilde{I}(\mathbf{y})$ **then** ▷ Plateau found.
      $N_I(\mathbf{y}) \leftarrow [y_1 - R_{user}, y_1 + R_{user}] \times \ldots \times [y_n - R_{user}, y_n + R_{user}]$.
      if IsMaximum(I, $\mathbf{y}$, $N_I(\mathbf{y})$) **then** ▷ See algo. 10.
        if $R_{Hessian}$ exists **then**
          $H_{\mathbf{y}} \leftarrow$ Hessian of the discrete scale–space representation of I evaluated at $\mathbf{y}$ with $t = \sigma^2$ and $M_1, \ldots, M_n = R_{Hessian}$ ▷ See equ. 2.19.
          $\Lambda_{\mathbf{y}} \leftarrow$ eigenvalues of the Hessian $H_{\mathbf{y}}$.
          if SelectMaximumOnHessianEigenvalues($\Lambda_{\mathbf{y}}$, *eigenTol*) **then** ▷ See algo. 11.
            $Maxima \leftarrow Maxima \cup \left\{\mathbf{p}^{(k)}\right\}$.
          end if
        else
          $Maxima \leftarrow Maxima \cup \{\mathbf{y}\}$.
        end if ▷ Compute Hessian option.
      end if ▷ Plateau pixel is maximum check.
    end if ▷ Plateau check.
  end for ▷ Loop for plateau check.
end if ▷ nD-maximum check.
end for ▷ Loop on peaks.
end for ▷ Loop on pixel lines.
return *Maxima*.
end procedure

Algorithm 10 Given a maximum pixel candidate $\mathbf{x}$ in a gray-scale image I and a neighbourhood $N_I(\mathbf{x})$ around $\mathbf{x}$, tests $\mathbf{x}$ is a local maximum for the neighbourhood $N_I(\mathbf{x})$.

Require: I grayscale image of domain D_I.
Require: $\mathbf{x} \in D_I$ maximum candidate.
Require: $N_I(\mathbf{x})$ neighbourhood around $\mathbf{x}$.

procedure ISMAXIMUM(I, $\mathbf{x}$, $N_I(\mathbf{x})$)
  for $\mathbf{y} \in N_I(\mathbf{x})$ **do**
    if $I(\mathbf{y}) > I(\mathbf{x})$ **then**
      ▷ If $\mathbf{y} \notin D_I$, then $I(\mathbf{y}) =$ minimum pixel value.
      return false.
    end if
  end for
  return true.
end procedure

Algorithm 11 Accept or reject a given maximum pixel candidate based on the eigenvalues of the Hessian evaluated at this maximum candidate.

Require: *tol* tolerance for the eigenvalues.
Require: $\Lambda = \{\lambda_i\}_{i=1,\dots,n}$, the eigenvalues of the Hessian evaluated at the considered maximum.

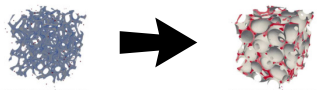
procedure SELECTMAXIMUMONHESSIANEIGENVALUES(Λ, *tol*)
  $\lambda_{min} \leftarrow \lambda_1$, $\lambda_{max\_pos} \leftarrow 0$.
  for $i = 1, \dots, n$ **do**
    if $(\lambda_i > 0)$ && $(\lambda_{max\_pos} < \lambda_i)$ **then**
      $\lambda_{max\_pos} = \lambda_i$.
    end if
    if $\lambda_{min} > \lambda_i$ **then**
      $\lambda_{min} = \lambda_i$.
    end if
  end for
  return $!(\lambda_{max\_pos} > tol\ |\lambda_{min}|)$ && $(\lambda_{min} < 0)$.
end procedure

Algorithm 12 Algorithm for solving problem 2.36 for determining the minimal integer radius of the convolution region needed for the computation of the Hessians so that the error stays below a given tolerance.

Require: tol error tolerance and σ, standard deviation for the Gaussian.
Require: Interval $[a_0, b_0]$ such that $(G(a_0;\sigma) - tol)(G(b_0;\sigma) - tol) < 0$

procedure INTEGRALOFGAUSSIANSECONDDERIVATIVEBISECTION(a_0, b_0, σ, tol)
    $i \leftarrow 0$.
    $R_i \leftarrow \frac{a_i + b_i}{2}$.
    repeat
        if $G(R_i;\sigma) - tol > 0$ **then**
            $a_{i+1} \leftarrow R_i$.
            $b_{i+1} \leftarrow b_i$.
        else
            $a_{i+1} \leftarrow a_i$.
            $b_{i+1} \leftarrow R_i$.
        end if
        $R_{i+1} \leftarrow \frac{a_{i+1} + b_{i+1}}{2}$.
        $i \leftarrow i + 1$.
    until $|R_{i-1} - R_i| < 1$
    return $R_{Hessian} \leftarrow ceil(R_i)$.
end procedure

2.3.9 Step 5: Parent ellipsoids



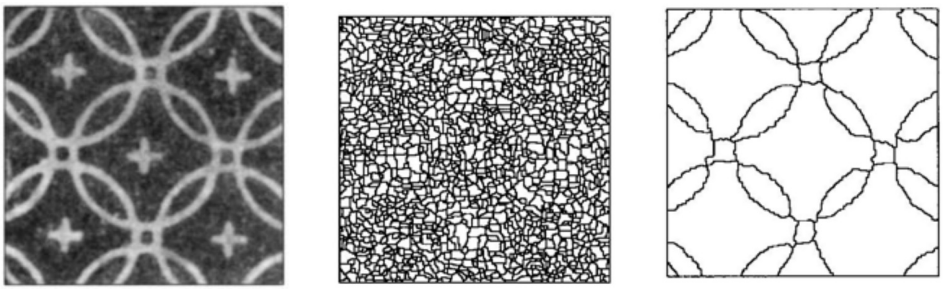
- **Aim**: from found local maxima, fit ellipsoids to cells. Cluster and merge overlapping ellipsoids so to obtain one fitted ellipsoid per cell.
- **Input**: local maxima and threshold image.
- **Output**: parent ellipsoids, each fitted to one particular cell.
- **Streaming**: see Section 3.10.

The oversegmentation problem

As the watershed transformation is a key and extensively used transformation, many efforts have been invested in designing fast (parallel) and accurate watershed algorithms ([21, 155, 159, 82, 162]. In this thesis, the watershed transformation is avoided and its *flooding basins* philosophy is replaced by *growing ellipsoids*. The advantage is that the input image can be locally referenced (i.e., only a small part of the image needs to be loaded into memory) and ellipsoids can be grown in parallel.

The watershed transform is often subject to oversegmentation: irrelevant small local minima (often generated by the noise in the image) generate catchment basins and induce the segmentation of the considered image in too many small zones (see, e.g., Figure 2.26). As a consequence, relevant features (as cells) may be fractured into subcomponents. Avoiding oversegmentation for the watershed transformation is a hard task. Several solutions may be used, depending on the nature of the considered image: simple smoothing [138], pre-processing by a H-maxima/minima transformation [176, 155], pre-identifying markers [155], etc. It should be noted that seemingly even the H-maxima/minima transformation appears to be insufficient for certain cases. Indeed, adaptive H-maxima transformations techniques have emerged [57, 125, 149, 33] and added to image analysis softwares such as *MAVI* [171].

Clearly, even if some superfluous local maxima are discarded on the basis of the eigenvalues of their associated Hessians, most of them are still present at this stage (Figure 2.27b). From local maxima, even superfluous ones, ellipsoids are associated. As is, this will lead to an oversegmentation problem. This oversegmentation problem is tackled here by merging overlapping ellipsoids above a given intersection volume ratio. Ellipsoids are associated to local maxima as follows: The ellipsoids are initially set as unit spheres centred at the local maxima positions (Figure 2.27c). They are then grown by an optimisation procedure described in Reference [39] (Figure 2.27d). This optimisation procedure iteratively grows ellipsoids inside associated polyhedra. At each iteration, the associated polyhedra are constructed from their current associated ellipsoids and the surrounding obstacles (here, the voxels corresponding to the cells boundaries in the considered CT image). Once the polyhedra are constructed, the ellipsoids are grown by one more increment. This process is repeated until the relative volume variation of each ellipsoid drops below a prescribed threshold. Figure 2.28 sketches in two dimensions how the algorithm works.



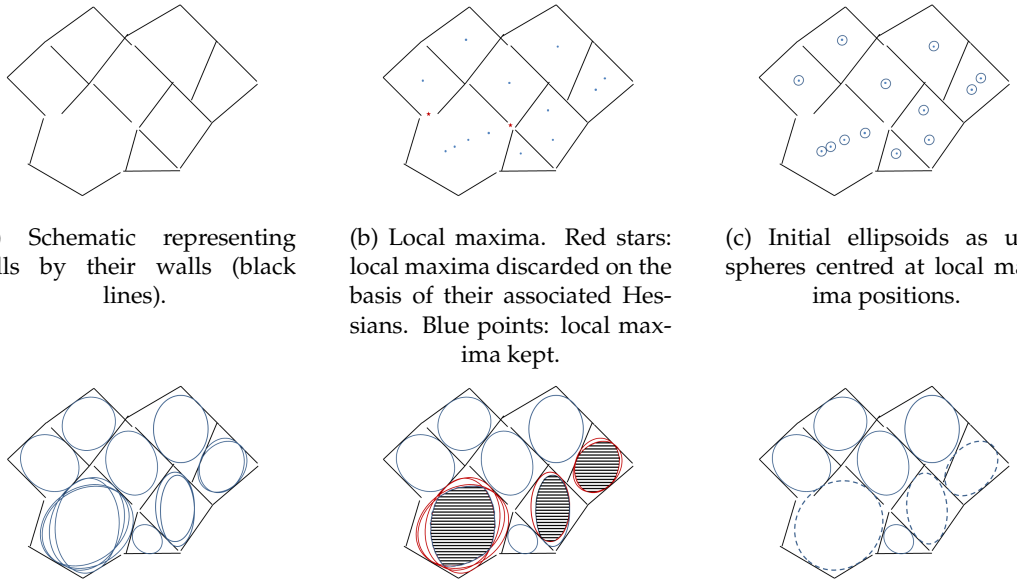
(a) Original image to be segmented (identification of the rings).

(b) Oversegmented image after watershed transformation.

(c) Segmented image after pre-filtering and the watershed transformation.

FIGURE 2.26: Example of a 2–dimensional image oversegmentation by the watershed transformation and its watershed transformation after pre–filtering. (Images for P. Soille [155], Figure 9.9, sub-figures a, c and f).

Once the ellipsoids are fully grown, overlapping volumes between pairs of intersecting ellipsoids are computed. If a given overlapping volume is bigger than a prescribed percentage of both ellipsoids volumes, both ellipsoids are clustered together (Figure 2.27e). Clustered ellipsoids are then merged inside a minimum volume covering ellipsoid (Figure 2.27f). These clustering and merging stages allow to tackle the oversegmentation problem without using the expensive H–maxima transform.



(a) Schematic representing cells by their walls (black lines).

(b) Local maxima. Red stars: local maxima discarded on the basis of their associated Hessians. Blue points: local maxima kept.

(c) Initial ellipsoids as unit spheres centred at local maxima positions.

(d) Grown ellipsoids using algorithm in Reference [39].

(e) Clustering of overlapping ellipsoids.

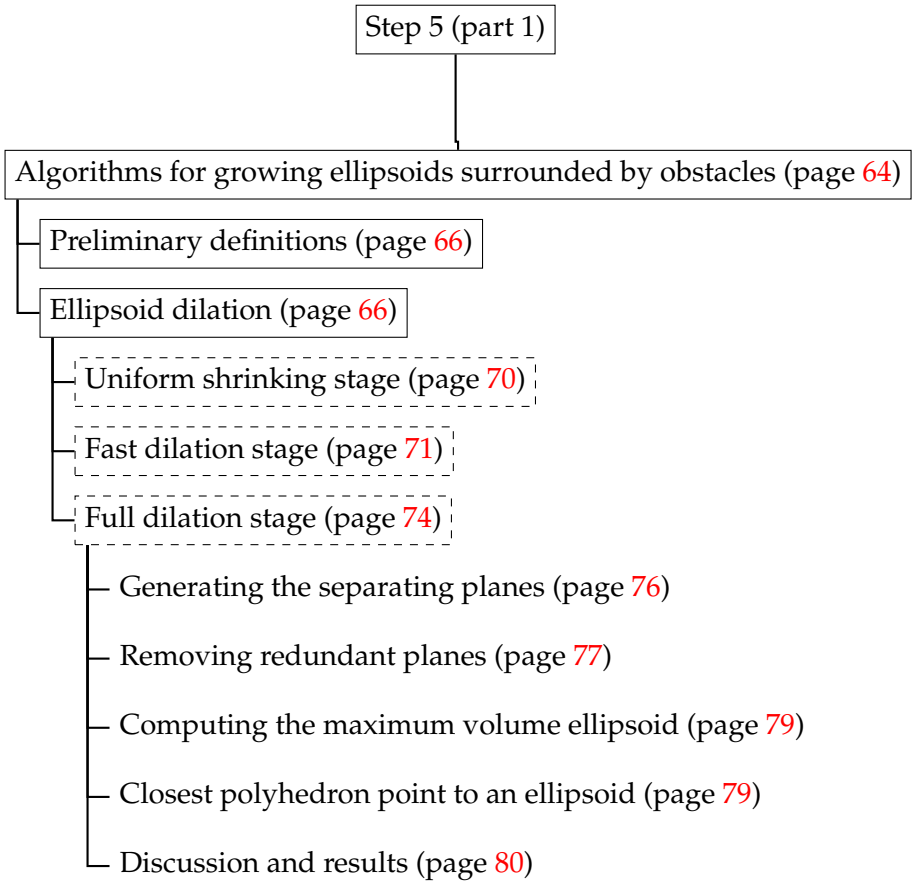
(f) Merging of clustered ellipsoids into a minimum volume covering ellipsoid (dashed blue curves).

FIGURE 2.27: Schematic representing the process of growing ellipsoids and clustering and merging groups of overlapping ellipsoids.

Step 5 (part 1) of the proposed image analysis procedure depicted in Figure 2.1b is organized into two parts:

1. Growing of parent ellipsoids.
2. Clustering and merging of overlapping parent ellipsoids.

Part 1 is structured as follows:



Step 5 (part 1): Algorithms for growing ellipsoids surrounded by obstacles.

This section describes more precisely how the algorithm of R. Deits et al. [39] operates for dilating ellipsoids. It should be noted that the algorithm of R. Deits et al. is preceded by a "fast dilatation stage". This stage also grows ellipsoids given some surrounding obstacles. It is however far less optimal than the algorithm of R. Deits et al. in the sense that it provides ellipsoids of lower volumes, but it is designed to be fast. The aim of the "fast dilatation stage" is to initialise the algorithm of R. Deits et al. with pre-grown ellipsoids and reduce the number of iterations necessary for it to converge.

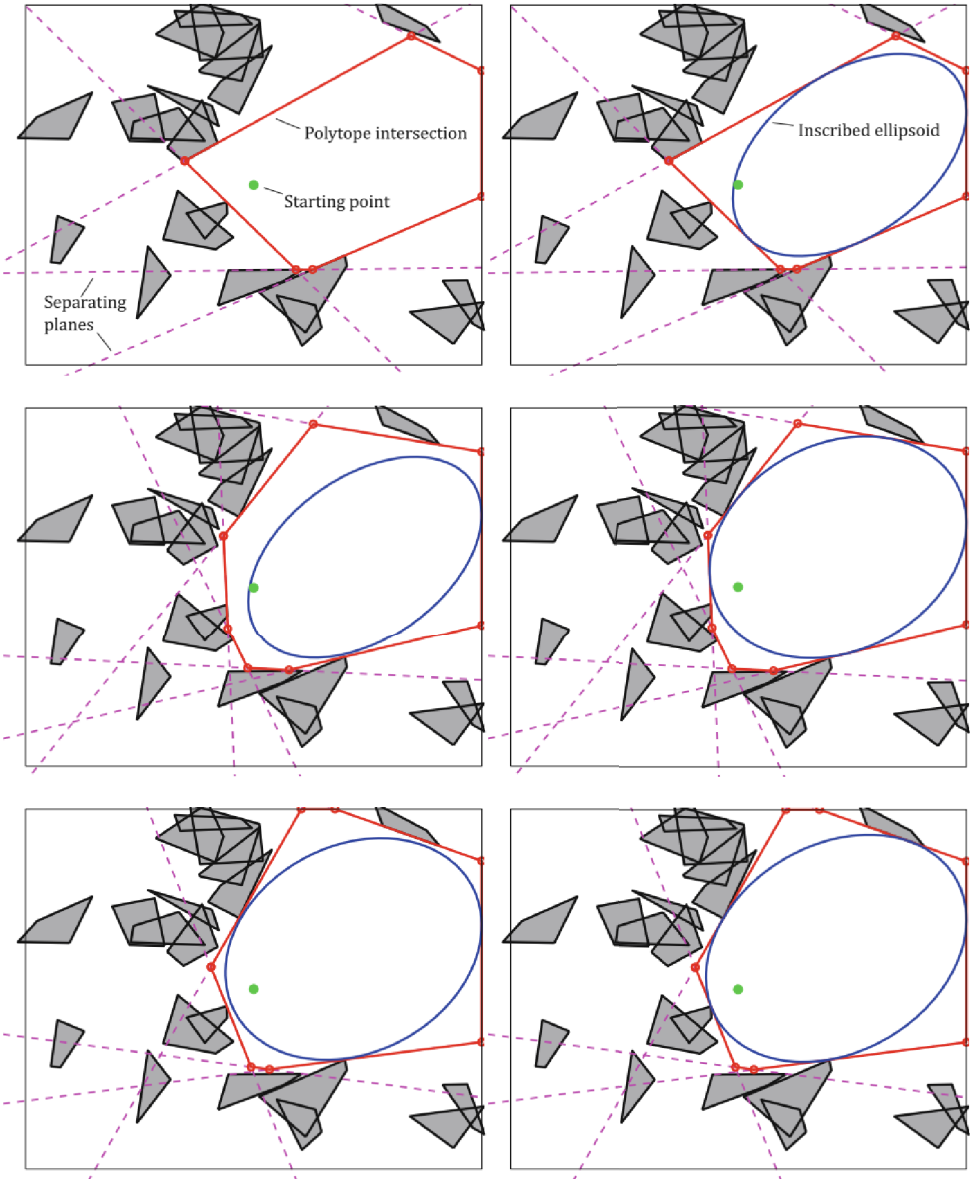


FIGURE 2.28: A demonstration of the algorithm for growing ellipses in a planar environment consisting of 20 uniformly randomly placed convex obstacles and a square boundary. Each row above shows one complete iteration of the algorithm: on the left, the hyperplanes are generated, and their polyhedra intersection is computed. On the right, the ellipse is inflated inside the polyhedra. After three iterations, the ellipse has ceased to grow, and the algorithm has converged. (Figure from [39]).

Preliminary definitions. Ellipsoids in $\mathcal{R}^n$ can be represented as follows:

Definition 2.5. An ellipsoid $\mathcal{E}$ in a n-dimensional affine Euclidean space can be represented as:

$$\begin{aligned}\mathcal{E} &= \{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x}-\mathbf{c})^t M(\mathbf{x}-\mathbf{c}) \leq 1\} \\ &= \{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x}-\mathbf{c})^t GG^t(\mathbf{x}-\mathbf{c}) \leq 1\} \\ &= \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1\} \\ &= \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + E\mathbf{y},\ \|\mathbf{y}\| \leq 1\}\end{aligned}$$

Where $\mathbf{c} \in \mathcal{R}^n$ is the centre of the ellipsoid and $M \in \mathcal{R}^{n\times n}$, $M = G^tG$ is a symmetric definite positive matrix (see Reference [133] for more details).

$G \in \mathcal{R}^{n\times n}$ is the lower Cholesky decomposition of M.

Often, M is called the *shape matrix* of the ellipsoid $\mathcal{E}$ and $E = G^{-t}$ is referred as the *scaling matrix* of the ellipsoid $\mathcal{E}$.

Definition 2.6. Given definition 2.5, the *surface* $\partial\mathcal{E}$ of an ellipsoid $\mathcal{E}$ in an n-dimensional affine Euclidean space can be represented as:

$$\partial\mathcal{E} = \{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x}-\mathbf{c})^t GG^t(\mathbf{x}-\mathbf{c}) = 1\}$$

Definition 2.7. Given definition 2.5, the *interior* $int\mathcal{E}$ of an ellipsoid $\mathcal{E}$ in an n-dimensional affine Euclidean space can be represented as:

$$int\mathcal{E} = \{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x}-\mathbf{c})^t GG^t(\mathbf{x}-\mathbf{c}) < 1\}$$

Definition 2.8. Given definition 2.5, the *uniform scaling of factor* α of an ellipsoid $\mathcal{E}$ is an ellipsoid $\mathcal{E}_\alpha$ that can be represented as:

$$\mathcal{E}_\alpha = \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq \alpha\}$$

Where $\alpha \in \mathcal{R}^+\backslash\{0\}$ is the scaling factor.

If $\alpha < 1$, the uniform scaling will be referred as a *uniform shrink*.
If $\alpha > 1$, the uniform scaling will be referred as a *uniform dilation*.

Ellipsoid dilation. The aim of ellipsoid dilation is, given an initial ellipsoid $\mathcal{E}^0$ located inside a given cell, to *fit* the ellipsoid to the cell. More precisely, the aim is to find the maximum volume ellipsoid inside a given cell where the wall's pixels are seen as obstacles for the ellipsoid (see Figure 2.29). Dilated ellipsoids will model the cell shapes and anisotropies.

The dilation of an initial ellipsoid $\mathcal{E}^0$ is performed in a two-fold fashion. Firstly, a *fast dilation*, possibly preceded by a *uniform shrink*, is performed. Secondly, a *full dilation* computes the final, fitted, ellipsoid $\mathcal{E}^*$. The initial ellipsoid $\mathcal{E}^0$ is a sphere centred at the position of its associated maximum with radius value of the value of the distance transform at that maximum. Algorithm 13 describes the whole process. Note that algorithm 13 calls other algorithms which will be described in details hereafter.

Algorithm 13 Given an initial ellipsoid $\mathcal{E}^0$ and an image I, iteratively fits the ellipsoid to its corresponding cell.

Require: Discrete digitalised image I of domain D_I.
Require: Pixel value $p \in Im(I)$ figuring the value associated to the cell walls.
Require: Ellipsoid $\mathcal{E}^0 \subset \mathcal{R}^n$.
Require: $\mathcal{O}_{artif}$ a set of artificial obstacles (may be empty).

1: **procedure** EllipsoidExpand(I, p, $\mathcal{E}^0$, $\mathcal{O}_{artif}$)
2: $k \leftarrow 0$.
3: $\tilde{B}^k \leftarrow$ AxisAlignedBox($\mathcal{E}^k$) ▷ Bounding box, algo. 27.
4: $f \leftarrow 2$. ▷ Box dilation factor.
5: $B^k \leftarrow$ DilateAxisAlignedBox($\tilde{B}^k$, f) ▷ Dilate box by a factor f, algo. 14.
6: $B^k \leftarrow B^k \cap D_I$ ▷ Crop box by the image domain.

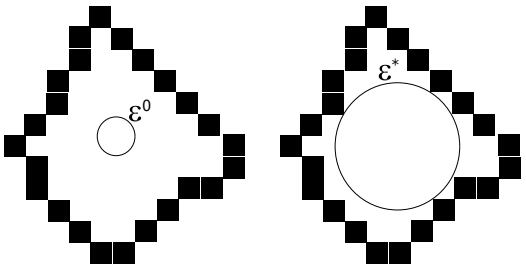
7: Scan for feature pixels in B^k: $\mathcal{O} \leftarrow \left\{\mathbf{v} \in B^k \mid I(\mathbf{v}) = p\right\}$.
8: $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{O}_{artif}$. ▷ Add artificial obstacles (if any).
9: Get scaling matrix E^k and centre $\mathbf{c}^k$ from ellipsoid $\mathcal{E}^k$.
10: $(\mathbf{v}^*, is_inside) \leftarrow$ClosestPixelToEllipsoid($E^k$, $\mathbf{c}^k$, $\mathcal{O}$) ▷ Algo. 15.

11: **if** *is_inside* **then**
12: $\mathcal{E}^k \leftarrow$ ShrinkEllipsoid($\mathcal{E}^k$, $\mathbf{v}^*$) ▷ Algo. 16.
13: **end if**

14: $\tilde{\mathcal{E}}^k \leftarrow$ FastEllipsoidExpand($\mathcal{O}$, $\mathcal{E}^k$) ▷ Algo. 17.
15: $\mathcal{E}^{k+1} \leftarrow$ FullEllipsoidExpand($\mathcal{O}$, $\tilde{\mathcal{E}}^k$) ▷ Algo. 18.

16: $\tilde{B}^{k+1} \leftarrow$ AxisAlignedBox($\mathcal{E}^{k+1}$) ▷ Algo. 27.
17: $B^{k+1} \leftarrow$ DilateAxisAlignedBox($\tilde{B}^{k+1}$, f) ▷ Algo. 14.
18: $B^{k+1} \leftarrow B^{k+1} \cap D_I$.

19: **if** $B^k \subset B^{k+1}$ or $\left|Vol\left(\mathcal{E}^{k+1}\right) - Vol\left(\mathcal{E}^k\right)\right| \leq Vol\left(\mathcal{E}^k\right) . tol$ **then**
20: **return** $\mathcal{E}^{k+1}$
21: **else**
22: $k \leftarrow k + 1$
23: Go back to line 7
24: **end if**
25: **end procedure**



(a) Initial ellipsoid $\mathcal{E}^0$. (b) Final ellipsoid $\mathcal{E}^*$ after fitting.

FIGURE 2.29: 2–dimensional sketch of the fitting of an ellipsoid inside a cell. Black squares represent pixel belonging to the cell walls.

Remark about algorithm 13. At lines 3 and 16 this algorithm uses axis–aligned boxes that surround ellipsoids[19]. Ellipsoids can not grow outside theses boxes. The aim of theses boxes is two-fold.

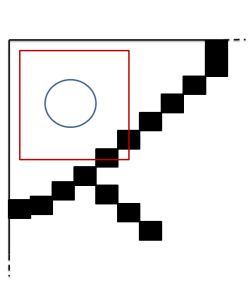
First, their role is to avoid that ellipsoids at located near the image boundaries grow to an infinite volume because the lack of obstacles (absence of feature voxels in the image on its boundaries). As illustrated in Figure 2.30, if an ellipsoid reaches one or more side of a box, the said box is dilated by a factor two (lines 5 and 17 in the algorithm) in order to allow the ellipsoid to continue its growth. However, the box is always cropped to the size of the image (lines 6 and 18), effectively forbidding any ellipsoid to grow outside the image.

Second, in case of streaming (see Section 3.10), if a surrounding box is dilated outside the boundaries of the current considered image slice, and if that boundary is not also a boundary of the whole image, the ITK filter implementing this algorithm can request a bigger slice through the ITK pipeline in order to have more available data for growing the ellipsoid associated to the corresponding box.

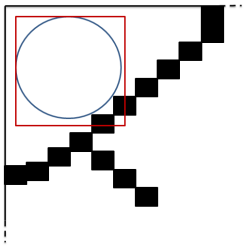
Algorithm 14 Dilates around its centre an axis-aligned box B by a dilation factor f.

Require: Axis-aligned box $B \subset \mathcal{R}^n$.
Require: Dilation factor $f \in \mathcal{R}$.
1: **procedure** DILATEAXISALIGNEDBOX(B, f)
2: **for** $i = 1, \ldots, n$ **do**
3: $temp \leftarrow \frac{1}{2}\left(s_i^+(1+f) + s_i^-(1-f)\right)$.
4: $s_i^- \leftarrow \frac{1}{2}\left(s_i^-(1+f) + s_i^+(1-f)\right)$.
5: $s_i^+ \leftarrow temp$.
6: **end for**
7: **return** $B = [s_1^-, s_1^+] \times \ldots \times [s_n^-, s_n^+]$.
8: **end procedure**

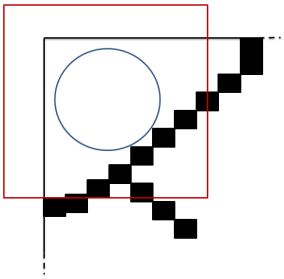
[19] The construction of such axis–aligned boxes from ellipsoids is described on page 94.



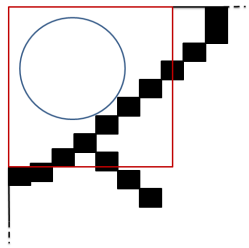
(a) Initial ellipsoid within its surrounding box with red lines in the upper–left corner of an image.



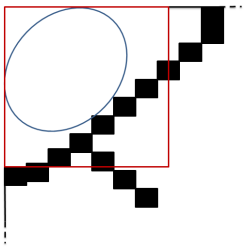
(b) Ellipsoid grown inside its cell and surrounding box.



(c) Surrounding box is dilated (Algorithm 14).



(d) Surrounding box is cropped by the image boundaries.



(e) Ellipsoid is grown again inside its cell and dilated surrounding box.

FIGURE 2.30: Schematic of the growth of an ellipsoid near the boundaries of an image. Image boundaries represented with black lines, ellipsoid represented with a blue line and its surrounding axis–aligned box with red lines.

Algorithm 15 Given an ellipsoid $\mathcal{E}$ and a set of pixels $\mathcal{O}$, computes the closest pixel to the ellipsoid $\mathcal{E}$.

Require: Ellipsoid $\mathcal{E} = \mathcal{E}(E, \mathbf{c}) \subset \mathcal{R}^n$.
Require: Set of point–wise obstacles $\mathbf{O} = \{\mathbf{v}_j\}_{j=1,\dots,m}$.

procedure CLOSESTPIXELTOELLIPSOID(E, $\mathbf{c}$, $\mathcal{O}$)
  $distances \leftarrow \emptyset$.
  for $\mathbf{v}_j \in \mathcal{O}$ **do**
    $\tilde{\mathbf{v}}_j \leftarrow E^{-1}(\mathbf{v}_j - \mathbf{c})$.
    $distances \leftarrow distances \cup \|\tilde{\mathbf{v}}_j\|$.
  end for
  $min\_distance \leftarrow min(distances)$.
  closest_point $\leftarrow \mathbf{v} \in \mathcal{O}$ associated to the minimal distance $min\_distance$.
  $is\_inside \leftarrow (min\_distance < 1)\ ?\ true : false$.
  return closest_point, is_inside.
end procedure

Remark about algorithm 15. Among a set of pixels $\mathcal{O}$, algorithm 15 simply finds the closest one to a given ellipsoid $\mathcal{E} = \mathcal{E}(E, \mathbf{c})$ and determines if this pixel lies inside the ellipsoid. This computation is performed by using the transform:

$$T_{\mathcal{E}} : \mathcal{R}^n \to \mathcal{R}^n; \mathbf{x} \mapsto T_{\mathcal{E}}(\mathbf{x}) = E^{-1}(\mathbf{x} - \mathbf{c}) \tag{2.38}$$

This transform reduces the ellipsoid $\mathcal{E}$ to the unit sphere centred at the origin. Algorithm 15 simply apply this transform to the pixels of the set $\mathcal{O}$ and look for the closest one to the origin. If the Euclidean distance of this closest pixel to the origin is less than one, then the considered pixel lies inside the ellipsoid.

Uniform shrinking stage. Before any dilation, it is already possible that an initial ellipsoid $\mathcal{E}^0$ overlaps an obstacle (pixel belonging to a cell wall, see Figure 2.31). Though improbable[20], this occurrence can not be ruled-out.

Let's be $\mathcal{E} = \mathcal{E}(E, \mathbf{c}) \subset \mathcal{R}^n$ an ellipsoid represented by a centre $\mathbf{c} \in \mathcal{R}^n$ and a scaling matrix $E \in \mathcal{R}^{n \times n}$, and a point $\mathbf{v} \in \mathcal{E}$. Then, a uniformly shrunk ellipsoid $\mathcal{E}_\alpha = \mathcal{E}_\alpha(\tilde{E}, \mathbf{c})$ that does not overlap the point $\mathbf{v}$ is such that:

$$\mathbf{v} = \mathbf{c} + \tilde{E}\mathbf{y} = \mathbf{c} + \alpha E\mathbf{y}. \tag{2.39}$$

With $\|\mathbf{y}\| = 1$.
The segment joining the centre $\mathbf{c}$ to the point $\mathbf{v}$ is intersecting the surface of $\mathcal{E}_\alpha$ at:

$$\mathbf{y} = \frac{\mathbf{v} - \mathbf{c}}{\|\mathbf{v} - \mathbf{c}\|}. \tag{2.40}$$

Then, it is possible to find the shrinking factor α such that $\mathbf{v}$ will lie on the surface of $\mathcal{E}_\alpha$:

$$\begin{aligned}
& \mathbf{v} = \mathbf{c} + \alpha E \tfrac{\mathbf{v}-\mathbf{c}}{\|\mathbf{v}-\mathbf{c}\|} \\
\Leftrightarrow \quad & (\mathbf{v} - \mathbf{c}) = \alpha E \tfrac{\mathbf{v}-\mathbf{c}}{\|\mathbf{v}-\mathbf{c}\|} \\
\Leftrightarrow \quad & E^{-1}(\mathbf{v} - \mathbf{c})\|\mathbf{v} - \mathbf{c}\| = (\mathbf{v} - \mathbf{c})\alpha \\
\Rightarrow \quad & \left(E^{-t}(\mathbf{v} - \mathbf{c})\right)^t (\mathbf{v} - \mathbf{c})\|\mathbf{v} - \mathbf{c}\| = \|\mathbf{v} - \mathbf{c}\|^2\alpha \\
\Leftrightarrow \quad & \alpha = \frac{\left(E^{-t}(\mathbf{v}-\mathbf{c})\right)^t(\mathbf{v}-\mathbf{c})}{\|\mathbf{v}-\mathbf{c}\|}
\end{aligned} \tag{2.41}$$

In the computation of the shrinking factor α, it is needed to compute the denominator $\|\mathbf{v} - \mathbf{c}\|$ which gives the distance between the cell wall pixel $\mathbf{v}$ and the ellipsoid's centre $\mathbf{c}$. This denominator is unlikely to be zero (in practice, this case never occurs) because, by construction, the centre of an initial ellipsoid $\mathcal{E}^0$ is located as far as possible from any cell wall pixel.

[20] Ellipsoid centres are located at the maxima of the distance transform of the considered image. I.e., ellipsoid centres are located as far as possible from cell walls.

(a) Initial ellipsoid $\mathcal{E}$ (black line).

(b) Uniformly shrinked ellipsoid $\tilde{\mathcal{E}}$ (black line).

FIGURE 2.31: Uniform shrinking of an ellipsoid $\mathcal{E}$ containing a pixel (black square) so that the new ellipsoid $\tilde{\mathcal{E}}$ does not contain the pixel anymore.

Algorithm 16 Given an ellipsoid $\mathcal{E}$ and a point $\mathbf{v} \in int\mathcal{E}$, computes an uniform shrunk ellipsoid $\mathcal{E}_\alpha$ such that $\mathbf{v} \in \partial\mathcal{E}_\alpha$.

Require: Ellipsoid $\mathcal{E} = \mathcal{E}(E, \mathbf{c}) \subset \mathcal{R}^n$.
Require: Point $\mathbf{v} \in int\mathcal{E}$.
1: **procedure** SHRINKELLIPSOID($\mathcal{E}$, $\mathbf{v}$)
2: **if** $\|\mathbf{v} - \mathbf{c}\| \leq \epsilon$ **then**
3: **return** Error. ▷ Division by (almost) zero. Unlikely to occurs.
4: **else**
5: $\alpha \leftarrow \frac{\left(E^{-t}(\mathbf{v}-\mathbf{c})\right)^t(\mathbf{v}-\mathbf{c})}{\|\mathbf{v}-\mathbf{c}\|}$.
6: $E_\alpha \leftarrow \alpha E$.
7: **return** Ellipsoid $\mathcal{E}_\alpha = \{\mathbf{x}\mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + E_\alpha\mathbf{y},\ \|\mathbf{y}\| \leq 1\}$.
8: **end if**
9: **end procedure**

Fast dilation stage. The fast dilation stage consists only in lengthen the semi-axes of the considered ellipsoid, till it is no more possible to increase the ellipsoid volume without hitting the surrounding obstacles. This stage will not find the maximum volume ellipsoid possible, but is faster than the full dilation stage and it will be used as an initialisation for the latter.

In practice, during the first iterations the semi-axis lengths $\{r_i\}_{i,\ldots,n}$ are uniformly increased by a factor α till the current ellipsoid $\mathcal{E}^k$ *hits* a cell wall pixel $\mathbf{v}$ (i.e. $\mathbf{v} \in \mathcal{E}^k$). In that case, the last iteration k is cancelled and a penalisation factor $tempf_i$ for each semi-axis length r_i, $i = 1, \ldots, n$ is computed. The penalisation factors are simply given as (the absolute values of) the projections of the vector connecting the ellipsoid's centre $\mathbf{c}$ to the hit point $\mathbf{v}$ on each semi-axis $\mathbf{u}_i$, $i = 1, \ldots, n$; where $\mathbf{u}_i$ is the ith eigenvector of the shape matrix M^0 of the initial ellipsoid[21] $\mathcal{E}^0$.

$$tempf_i(\mathbf{v}) = \left\| r_i \, (\mathbf{v} - \mathbf{c})^t \mathbf{u}_i \right\|, \ i = 1, \ldots, n \tag{2.42}$$

The penalisation factor 2.42 is then weighted by the penalisation factors over all the semi-axes $\mathbf{u}_i$, $i = 1, \ldots, n$:

$$f_i(\mathbf{v}) = \frac{tempf_i(\mathbf{v})}{\sum_{j=1}^{n} tempf_j(\mathbf{v})}, \ i = 1, \ldots, n \tag{2.43}$$

[21] As the fast dilation stage only changes the radii of the semi-axes and not their orientations, their corresponding eigenvectors are constant throughout the iterations.

At iteration k, if the ellipsoid $\mathcal{E}^k$ hits a cell wall pixel $\mathbf{v}^m$, while having already hit $(m-1)$ cell wall pixels $\mathbf{v}^1, \ldots, \mathbf{v}^{m-1}$ during the preceding iterations, its weighted penalisation factor for all the m cell wall pixels is given as:

$$f_i^m = f_i^{m-1} + \frac{tempf_i(\mathbf{v}^m)}{\sum_{j=1}^{n} tempf_j(\mathbf{v}^m)},\ i = 1, \ldots, n \tag{2.44}$$

Where f_i^{m-1} is defined recursively by equation 2.44 and f_i^1 is defined by equation 2.43 with $\mathbf{v} = \mathbf{v}^1$.

A non-recursive definition of f_i^m is given by:

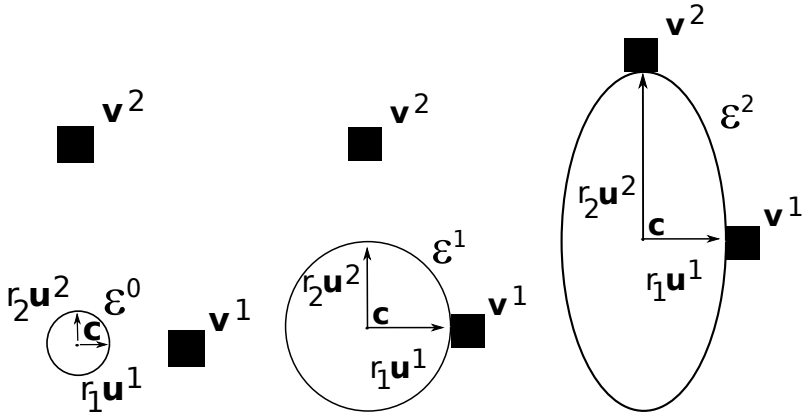
$$f_i^m = \sum_{k=1}^{m} \frac{tempf_i(\mathbf{v}^k)}{\sum_{j=1}^{n} tempf_j(\mathbf{v}^k)},\ i = 1, \ldots, n \tag{2.45}$$

Finally, at iteration k, the semi-axis lengths $\{r_i\}_{1,\ldots,n}$ are updated as follow:

$$r_i^{k+1} = r_i^k + \alpha^m \left(1 - \frac{f_i^m}{m}\right) r_i^k \tag{2.46}$$

Where α^m is a rate factor and $\left(1 - \frac{f_i^m}{m}\right)$ is the final penalisation term.

By construction, $\frac{f_i^m}{m}$ is bounded by the interval $[0, 1]$. Each time a cell wall pixel $\mathbf{v}$ is hit by the growing ellipsoid, its projections relatively to the ellipsoid's centre $\mathbf{c}$ on the semi-axes are computed. These projections are added to the weighted penalisation factors f_i^m which increase towards one. Conversely, the term $\left(1 - \frac{f_i^m}{m}\right)$ will decrease towards zero. The more the vector $(\mathbf{v} - \mathbf{c})$ is aligned with a semi-axis $\mathbf{u}_j$, the more its corresponding radius r_j will be penalised. Similarly, the more cell wall pixels are hit, the more the radii $\{r_i\}_{1,\ldots,n}$ will be penalised. Figure 2.32 sketches the effect of the penalisation terms over the semi-axis lengths $\{r_i\}_{1,\ldots,n}$ for a simple 2-dimensional case.



(a) Initial ellipsoid $\mathcal{E}^0$ with semi–axes $r_1\mathbf{u}^1$ and $r_2\mathbf{u}^2$.

(b) Semi-axes $r_1\mathbf{u}^1$ and $r_2\mathbf{u}^2$. are uniformly increased in length till the ellipsoid hits the first cell wall pixel $\mathbf{v}^1$.

(c) Only the semi–axis $r_2\mathbf{u}^2$ is lengthened, as the semi–axis $r_1\mathbf{u}^1$ is entirely penalised by $\mathbf{v}^1$.

FIGURE 2.32: 2–dimensional sketch example of a fast ellipsoid dilation. Cell wall pixels are represented by black squares.

Algorithm 17 Given a initial ellipsoid $\mathcal{E}^0$ and a list of cell wall pixels $\mathcal{O}$, increases as much as possible the semi–axes lengths of $\mathcal{E}^0$ such that $\mathcal{E}^0 \cap \mathcal{O} = \emptyset$.

Require: Ellipsoid $\mathcal{E}^0 \subset \mathcal{R}^n$.
Require: List of obstacles $\mathcal{O} = \left\{\mathbf{v}_j\right\}_{j=1,\dots,m}$.

procedure FASTELLIPSOIDEXPAND($\mathcal{O}$, $\mathcal{E}^0$)
  $m \leftarrow 1, k \leftarrow 0$.
  $\alpha^m \leftarrow 0.5$. ▷ Rate parameter.
  $f_i^m \leftarrow 0,\ i = 1, \dots, n$. ▷ Penalisation factors.
  repeat
    Get the semi–axis lengths $\{r_i\}_{i=1,\dots,n}$ and eigenvectors $\{\mathbf{u}_i\}_{i=1,\dots,n}$ of $\mathcal{E}^k$.
    $r_i^{k+1} \leftarrow r_i^k + \alpha^m \left(1 - \frac{f_i^m}{m}\right) r_i^k,\ i = 1, \dots, n$ ▷ dilation step.
    $\Delta r^k = \max_{i=1,\dots,n} \left(r_i^{k+1} - r_i^k\right)$.
    Set $\mathcal{E}^{k+1}$ as $\mathcal{E}^k$ with radii $r_i^{k+1},\ i = 1, \dots, n$.
    Extract scaling matrix E^{k+1} and ellipsoid centre $\mathbf{c}$ from ellipsoid $\mathcal{E}^{k+1}$
    $(\mathbf{v}^*, is\_inside) \leftarrow$ CLOSESTPIXELTOELLIPSOID(E^{k+1}, $\mathbf{c}$, $\mathcal{O}$) ▷ Algo. 15.

    if *is_inside* **then**
      $\alpha^{m+1} \leftarrow \alpha^m / 2$.
      $tempf_i(\mathbf{v}^*) = \left\| r_i^k (\mathbf{v}^* - \mathbf{c})^t \mathbf{u}_i \right\|,\ i = 1, \dots, n$.
      $f_i^{m+1} \leftarrow f_i^m + \frac{tempf_i(\mathbf{v}^*)}{\sum_{j=1}^n tempf_j(\mathbf{v}^*)},\ i = 1, \dots, n$.
      $m \leftarrow m + 1$.
      Continue. ▷ Go back at the beginning of the loop.
    end if
    $k \leftarrow k + 1$.
  until $\Delta r^k \leq 1$ ▷ Stop when the radius increment is less or equal to one pixel.
  return Ellipsoid $\mathcal{E}^k$.
end procedure

Full dilation stage. This stage is the one which really *fits* the considered initial ellipsoid $\mathcal{E}^0$ to its corresponding cell. Starting from the initial ellipsoid $\mathcal{E}^0$ and a set of cell wall pixels $\mathcal{O}$, it computes the maximum volume ellipsoid $\mathcal{E}^*$ such that $\mathcal{E}^* \cap \mathcal{O} = \emptyset$. To this end, it uses the algorithm of R. Deits and R. Tedrake [39] which iteratively dilates the ellipsoid inside a growing set of polyhedra P^k, $k = 1, \ldots$. The polyhedra P^k, $k = 1, \ldots$ are themselves constructed from the current considered ellipsoid at a given iteration and from the set of cell wall pixels $\mathcal{O}$.

Algorithm 18 Given a initial ellipsoid $\mathcal{E}^0$ and a list of cell wall pixels $\mathcal{O}$, finds the maximum volume ellipsoid $\mathcal{E}^*$ such that $\mathcal{E}^* \cap \mathcal{O} = \emptyset$.

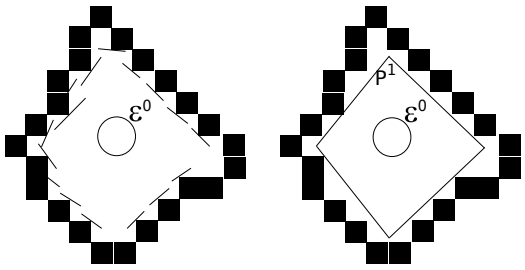
Require: Ellipsoid $\mathcal{E}^0 \subset \mathcal{R}^n$.
Require: List of obstacles $\mathcal{O} = \left\{\mathbf{v}_j\right\}_{j=1,\ldots,m}$.
1: **procedure** FULLELLIPSOIDEXPAND($\mathcal{O}$, $\mathcal{E}^0$)
2: Extract scaling matrix E^0 and ellipsoid centre $\mathbf{c}$ from ellipsoid $\mathcal{E}^0$
3: **return** DEITSMAXVOLELLIPSOID(E^0, $\mathbf{c}$, $\mathcal{O}$) ▷ Algo. 21.
4: **end procedure**

Ellipsoid dilations inside a given cell is performed thanks to the algorithm of R. Deits and R. Tedrake [39]. This algorithm was originally designed for determining large, obstacle-free, convex (ellipsoidal) regions where a bipedal robot can safely step and move. As pointed by R. Deits and R. Tedrake themselves, it has been found that this algorithm could be used for a totally different context: namely here determining a maximum volume ellipsoid located inside a given cell.

Here, the list of obstacles $\mathcal{O}$ is simply a list of feature pixels $\{\mathbf{v}_k\}_{1,\ldots,K}$ (i.e., pixels identified as cell struts/walls) lying in a given neighbourhood V of an initial ellipsoid $\mathcal{E}^0$.

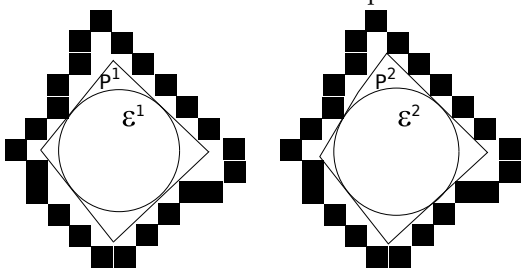
The algorithm of R. Deits and R. Tedrake first consists in generating separating planes between the obstacles and the initial ellipsoid $\mathcal{E}^0$. Then, "redundant" separating planes are discarded. The remaining planes define a polyhedron P^1 containing the initial ellipsoid $\mathcal{E}^0$. From this initial ellipsoid $\mathcal{E}^0$, a maximum volume inscribed ellipsoid $\mathcal{E}^1 \subset P^1$ is computed. Given this new ellipsoid $\mathcal{E}^1$, a new set of separating planes is generated from it. Then redundant separating planes are discarded from this set and a new polyhedron P^2 is determined from the remaining separating planes. From $\mathcal{E}^1$, a new maximum volume inscribed ellipsoid $\mathcal{E}^2 \subset P^2$ is computed. This process is repeated until the relative volume difference between two consecutive ellipsoids $\mathcal{E}^i$ and $\mathcal{E}^{i+1}$ is below a given threshold. Figure 2.33 gives a sketch of the algorithm of R. Deits and R. Tedrake in two dimensions.

The main advantage of the algorithm of R. Deits and R. Tedrake lies in the fact that redundant planes are removed. Indeed, the problem of computing a maximum volume ellipsoid inscribed in a given polyhedron P defined by planes is a constrained convex optimisation problem, where each plane corresponds to a constraint. Complexity bounds for the optimisation algorithms solving this kind of optimisation problems are mainly driven by the number of constraints. Typically, the complexity bounds with respect to the number of constraints m for these algorithm are of the order $O\left(m^{3.5}\, ln\left(\frac{m}{e}\right)\right)$, where $e \in]0,1[$ is a constant [189]. Therefore, the less constraints (a.k.a. the less separating planes) the better.



(a) Generation of separating planes.

(b) Removing of redundant separating planes and computation of a first polyhedron P^1 from ellipsoid $\mathcal{E}^0$.



(c) Dilation of ellipsoid $\mathcal{E}^0$ to ellipsoid $\mathcal{E}^1$ inside polyhedron P^1.

(d) Computation of a new polyhedron P^2 from ellipsoid $\mathcal{E}^1$ and dilation of ellipsoid $\mathcal{E}^1$ to ellipsoid $\mathcal{E}^2$.

FIGURE 2.33: Sketch of the algorithm of R. Deits and R. Tedrake applied to a two–dimensional image of a cell. Pixels representing the cell walls are shown as black squares, separating planes and polyhedra as black lines.

As cell walls may be constituted by a large number of pixels, each representing one obstacle and each associated to a separating plane, the number of constraints may rapidly become overwhelming for the considered optimisation algorithms. But, since most of the separating planes in the algorithm of R. Deits and R. Tedrake are redundant, this dramatically improves the efficiency of the stage where maximum volumes inscribed ellipsoids are computed.

Given an initial ellipsoid $\mathcal{E}^0$ and a list of convex obstacles $\{\zeta_j\}_{j=1,\dots,K}$, the algorithm of R. Deits and R. Tedrake finds a polyhedron P^* whose boundaries are touching the obstacles and a maximum volume inscribed ellipsoid $\mathcal{E}^* \subset P^*$.

Generating the separating planes. Given an ellipsoid $\mathcal{E}$ and a convex obstacle ζ_j, the aim is to find a separating plane H_j which touches the obstacle ζ_j and separates the ellipsoid $\mathcal{E}$ from the obstacle.

The plane H_j can be represented as:

$$H_j = H_j(\hat{\mathbf{n}}_j, b_j) = \left\{ \mathbf{x} \in \mathcal{R}^n \mid \hat{\mathbf{n}}_j^t \mathbf{x} - b_j = 0 \right\} \tag{2.47}$$

Where $\hat{\mathbf{n}}_j \in \mathcal{R}^n$ is the normal vector and $b_j \in \mathcal{R}$ a shift term.

The ellipsoid $\mathcal{E} = \mathcal{E}(E, \mathbf{c})$ can be represented as:

$$\mathcal{E}(E, \mathbf{c}) = \left\{ \mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{c})^t E^{-t} E^{-1} (\mathbf{x} - \mathbf{c}) \leq 1 \right\} \tag{2.48}$$

Where $E = G^{-t} \in \mathcal{R}^{n \times n}$ is the scaling matrix and $\mathbf{c} \in \mathcal{R}^n$ its centre.

The algorithm of R. Deits and R. Tedrake aims to solve the following optimisation problem:

$$\begin{array}{ll} maximise_{E, \mathbf{c}, \{\mathbf{n}_j, b_j\}_{j=1,\dots,K}} & log\, det\, E \\ \text{s.t.} & \mathbf{n}_j^t \mathbf{v}_k \geq b_j \; \forall \mathbf{v}_k \in \zeta_j, \; j = 1, \dots, K \\ & sup_{\|\mathbf{y}\| \leq 1} \, \mathbf{n}_i^t (\mathbf{c} + E)\mathbf{y} \leq b_i, \; \forall i = 1, \dots, K \end{array} \tag{2.49}$$

Given the closest point $\mathbf{x}^* \in \zeta_j$ to $\mathcal{E}(E, \mathbf{c})$, R. Deits and R. Tedrake take the normal vector $\hat{\mathbf{n}}_j$ of the separating plane H_j as the gradient vector of the ellipsoid's barrier function at $\mathbf{x}^*$:

$$\begin{aligned} \mathbf{n}_j &= \nabla_{\mathbf{x}} \left[(\mathbf{x} - \mathbf{c})^t E^{-t} E^{-1} (\mathbf{x} - \mathbf{c}) \right]_{\mathbf{x} = \mathbf{x}^*} \\ &= 2 E^{-t} E^{-1} (\mathbf{x}^* - \mathbf{c}) \end{aligned} \tag{2.50}$$

$$\hat{\mathbf{n}}_j = \frac{\mathbf{n}_j}{\|\mathbf{n}_j\|} \tag{2.51}$$

The closest point $\mathbf{x}^* \in \zeta_j$ is found by solving an optimisation problem, using the algorithm of Mattingley et al. [110]. See Appendix D for more details. Note that if the considered obstacle ζ_j is reduced to a point, there is no need to solve an optimisation problem.

Finally, the separating plane H_j is imposed to pass through $\mathbf{x}^*$:

$$b_j = \hat{\mathbf{n}}_j^t \mathbf{x}^* \tag{2.52}$$

Algorithm 19 describes the procedure.

Algorithm 19 Compute a separating plane $H_j = H_j(\hat{\mathbf{n}}_j, b_j)$ between an ellipsoid $\mathcal{E}$ and a convex polyhedral obstacle ζ_j. R. Deits and R. Tedrake ([39]).

Require: Ellipsoid $\mathcal{E}(E, \mathbf{c}) \subset \mathcal{R}^n$.
Require: Convex polyhedral obstacle $\zeta_j \subset \mathcal{R}^n$.
procedure DEITSTANGENTPLANE(E, $\mathbf{c}$, ζ_j)
  if ζ_j not a point **then**
    Find the closest point $\mathbf{x}^* \in \zeta_j$ to $\mathcal{E}$: $\mathbf{x}^* \leftarrow \text{CVX}(E, \mathbf{c}, \zeta_j)$. ▷ Algorithm 41.
  else
    $\mathbf{x}^* \leftarrow \zeta_j$.
  end if
  $\mathbf{n}_j \leftarrow 2E^{-t}E^{-1}(\mathbf{x}^* - \mathbf{c})$.
  $\hat{\mathbf{n}}_j \leftarrow \frac{\mathbf{n}_j}{\|\mathbf{n}_j\|}$.
  $b_j \leftarrow \hat{\mathbf{n}}_j^t \mathbf{x}^*$.
  return $\hat{\mathbf{n}}_j$, b_j. ▷ Quantities defining the plane H_j.
end procedure

Removing redundant planes. As already stated, iterating over all the obstacles will generate a large number of separating planes. For already mentioned algorithmic efficiency reasons, it is preferable to keep the number of separating planes as small as possible. To this aim, *redundant* planes are removed.

Given an obstacle ζ_j, it is possible to compute its corresponding separating plane $H_j = H_j(\hat{\mathbf{n}}_j, b_j)$ relatively to a given ellipsoid $\mathcal{E}$. If, for another obstacle ζ_k, $k \neq j$, it holds:

$$\hat{\mathbf{n}}_j^t \mathbf{y} - b_j \geq 0, \ \forall \mathbf{y} \in \zeta_k \tag{2.53}$$

Then, the computation of the separating plane corresponding to ζ_k can be skipped, as the obstacle ζ_k is already separated from ellipsoid $\mathcal{E}$ by the separating plane H_j. In that case, the separating plane H_k associated to the obstacle ζ_k is called *redundant*.

In practice, as the considered obstacles are convex polyhedra, equation 2.53 need only to be checked on the vertices of ζ_k. Algorithm 20 describes the procedure for computing the non-redundant separating planes.

Remark about algorithm 20. This algorithm computes the separating planes between obstacles and an ellipsoid $\mathcal{E}$. The next step, involves computing a maximum volume ellipsoid belonging to the intersection of the half-spaces $\{HS_j\}_{j=1,\dots,m}$ defined by the separating planes. Namely:

$$P = \cap_{j=1}^{m} HS_j = \{\mathbf{x} \in \mathcal{R}^n \mid A\mathbf{x} \leq \mathbf{b}\} \tag{2.54}$$

Where the matrix $A \in \mathcal{R}^{m \times m}$ and the vector $\mathbf{b} \in \mathcal{R}^m$ are computed by algorithm 20; while the half–spaces $\{HS_j\}_{j=1,\dots,m}$ are defined by:

$$HS_j = \left\{\mathbf{x} \in \mathcal{R}^n \mid \hat{\mathbf{n}}_j^t \mathbf{x} \leq b_j\right\}, \ j = 1, \dots, m \tag{2.55}$$

Algorithm 20 Compute the non-redundant separating planes between a given ellipsoid $\mathcal{E}$ and a list of convex polyhedral obstacles $\mathcal{O} = \{\zeta_j\}_{j=1,\dots,K}$. R. Deits and R. Tedrake ([39]).

Require: Ellipsoid $\mathcal{E}(E, \mathbf{c}) \subset \mathcal{R}^n$.
Require: List of convex polyhedral obstacles $\mathcal{O}$.

procedure DEITSSEPARATINGPLANES(E, $\mathbf{c}$, $\mathcal{O}$)
  $\mathcal{O}_{excluded} \leftarrow \emptyset$.
  $\mathcal{O}_{remaining} \leftarrow \mathcal{O}$.
  $i \leftarrow 1$.
  while $\mathcal{O}_{remaining} \neq \emptyset$ **do**
    $\zeta^* \leftarrow$ closest obstacle in $\mathcal{O}$ to ellipsoid $\mathcal{E}$.
    $(\hat{\mathbf{n}}_i, b_i) \leftarrow$ DEITSTANGENTPLANE(E, $\mathbf{c}$, ζ^*) ▷ Algorithm 19.
    for $\zeta_k \in \mathcal{O}_{remaining}$ **do**
      if $\hat{\mathbf{n}}_i^t \mathbf{x}_j - b_i \geq 0$, $\forall \mathbf{x_j} \in \zeta_k$ **then**
        $\mathcal{O}_{remaining} \leftarrow \mathcal{O}_{remaining} \backslash \zeta_k$.
        $\mathcal{O}_{excluded} \leftarrow \mathcal{O}_{excluded} \cup \zeta_k$.
      end if
    end for
    $i \leftarrow i + 1$.
  end while
  $A \leftarrow \begin{pmatrix} \hat{\mathbf{n}}_1^t \\ \hat{\mathbf{n}}_2^t \\ \vdots \end{pmatrix}$, $\mathbf{b} \leftarrow \begin{pmatrix} b_1 \\ b_2 \\ \vdots \end{pmatrix}$. ▷ Non-redundant plane parameters associated to obstacles $\mathcal{O} \backslash \mathcal{O}_{excluded}$.
  return A, $\mathbf{b}$.
end procedure

However, there is no guarantee that the polyhedron P is bounded[22]. If the polyhedron P is unbounded, so can also be the corresponding maximum volume ellipsoid and the algorithm of R. Deits et al. will never converge. In order to avoid this unbounded case, a bounding box is added to the constraints computed by the algorithm 20. In practice, this bounding box is, at most, of the size of the *largest possible region* of the considered image.

Computing the maximum volume ellipsoid Given the algorithm 20, computing a bounded polyhedron given a set of polyhedral convex obstacles $\mathcal{O} = \{\zeta_j\}_{j=1,\dots,K}$ and an initial ellipsoid $\mathcal{E}^0$, it is now possible possible to iteratively find a maximum volume ellipsoid $\mathcal{E}^*$ which is constrained by the set of obstacles $\mathcal{O}$. Algorithm 21 describes how such a maximum volume ellipsoid $\mathcal{E}^*$ is computed. Figure 2.33 sketches the algorithm for a 2-dimensional case.

Algorithm 21 Given a set of polyhedral convex obstacles $\mathcal{O} = \{\zeta_j\}_{j=1,\dots,K}$ and an initial ellipsoid $\mathcal{E}^0$, computes iteratively a maximum volume ellipsoid constrained by the obstacles. R. Deits and R. Tedrake ([39]).

Require: Ellipsoid $\mathcal{E}^0(E^0, \mathbf{c}^0) \subset \mathcal{R}^n$.
Require: Set of convex polyhedral obstacles $\mathcal{O}$.
procedure DEITSMAXVOLELLIPSOID(E^0, $\mathbf{c}^0$, $\mathcal{O}$)
  $i \leftarrow 0$.
  repeat
    $(A^{i+1}, \mathbf{b}^{i+1}) \leftarrow$ DEITSSEPARATINGPLANES(E^i, $\mathbf{c}^i$, $\mathcal{O}$) ▷ Compute polyhedron P^i, algorithm 20.
    $(E^{i+1}, \mathbf{c}^{i+1}) \leftarrow$ MAXVOLELLIPSOIDINPOLYHEDRON(A^{i+1}, $\mathbf{b}^{i+1}$, $\mathbf{c}^i$) ▷ Compute the maximum volume ellipsoid inside polyhedron P^{i+1}, algorithm 42.
    $i \leftarrow i + 1$.
  until $(detE^i - detE^{i-1})/detE^{i-1} < tol$
  return A_i, $\mathbf{b}_i$, E^i, $\mathbf{c}^i$.
end procedure

Closest polyhedron point to an ellipsoid. The algorithm of R. Deits and R. Tedrake requires to find the closest point belonging to a convex polyhedron to a given ellipsoid. The present section describes how such a point can be found.

Let's ζ_j be a convex obstacle with vertices $\{\mathbf{v}_{j,i}\}_{i=1,\dots,m}$ such that ζ_j is the convex hull of $\{\mathbf{v}_{j,i}\}_{i=1,\dots,m}$. Let's also $\mathcal{E} \subset \mathcal{R}^n$ be an ellipsoid of centre $\mathbf{c} \in \mathcal{R}^n$ and non–singular scaling matrix $E = G^{-t} \in \mathcal{R}^{n \times n}$ represented by:

$$\mathcal{E}(E, \mathbf{c}) = \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + E\mathbf{y}, \|\mathbf{y}\| \leq 1\} \tag{2.56}$$

[22]For instance, this case may occur if there is not enough separating planes, or if all the obstacle are located to "one side" of the ellipsoid.

In order to find the closest point $\mathbf{x}^* \in \zeta_j$ to the ellipsoid $\mathcal{E}$, it is convenient to construct the transform that maps the ellipsoid $\mathcal{E}$ to the unit ball centred at the origin:

$$\tilde{\mathcal{E}} = \{\tilde{\mathbf{x}} \in \mathcal{R}^n \mid \|\tilde{\mathbf{x}}\| \leq 1\} \tag{2.57}$$

In that case, the transformed obstacle is:

$$\tilde{\zeta}_j = \text{ConvexHull}\left\{\tilde{\mathbf{v}}_{j,1}, \ldots, \tilde{\mathbf{v}}_{j,m}\right\} \tag{2.58}$$

$$\tilde{\mathbf{v}}_{j,k} = E^{-1}(\mathbf{v}_{j,k} - \mathbf{c}),\ k = 1, \ldots, m \tag{2.59}$$

Indeed, from the definition 2.56 of an ellipsoid, one can see that:

$$\mathbf{x} = \mathbf{c} + E\mathbf{y},\ \|\mathbf{y}\| \leq 1 \tag{2.60}$$

$$\Leftrightarrow \mathbf{y} = E^{-1}(\mathbf{x} - \mathbf{c}) \tag{2.61}$$

Thus, the transform:

$$T_{\mathcal{E}} : \mathcal{R}^n \to \mathcal{R}^n, \mathbf{x} \mapsto T_{\mathcal{E}}(\mathbf{x}) = E^{-1}(\mathbf{x} - \mathbf{c}) \tag{2.62}$$

maps an ellipsoid $\mathcal{E}$ to a unit ball centred at the origin.

The closest point $\tilde{\mathbf{x}}^*$ of $\tilde{\zeta}_j$ to the unit ball $\tilde{\mathcal{E}}$ is then simply the closest point of $\tilde{\zeta}_j$ to the origin. Mathematically, this problem can be expressed as [39]:

$$\begin{cases} arg\ min_{\tilde{\mathbf{x}} \in \mathcal{R}^n, \mathbf{w} \in \mathcal{R}^m}\ \|\tilde{\mathbf{x}}\|^2 = \tilde{\mathbf{x}}^t\tilde{\mathbf{x}} \\ \quad \sum_{k=1}^{m} \tilde{\mathbf{v}}_{j,k} w_k = \tilde{\mathbf{x}} \\ \text{s.t.}\ \ \sum_{k=1}^{m} w_k = 1 \\ \quad \mathbf{w} \succeq \mathbf{0} \end{cases} \tag{2.63}$$

An algorithm for solving the problem 2.63 is given in appendix D, while the mean of computing the maximum volume ellipsoid inside a polyhedron is given in appendix E.

Discussion

It should be emphasised that the idea of modelling the geometry of a foam as the result of a growth process, as presented here, is not new. The Voronoï and Laguerre tessellations can be defined as the result of a growth process of spheres [136]. For growth of spheres, the idea has been explicitly exploited by, e.g., K. Mader et al. ([103]). However, their approach only deals with spheres, ignoring thus the possible effects of anisotropy of foams on their mechanical behaviours. Ellipsoids are indeed able to capture cell anisotropies, which is considered as a necessity for assessing the mechanical behaviour of foams [56, 164, 30, 6]. Although sets of growing spheres are able to capture some foam anisotropies [103], it is believed that ellipsoids lead to better results by capturing more precisely local cell features [160, 150, 151]. Growth of ellipsoids has been proposed in Reference [8], and, in the context of polycrystalline microstructures reconstruction in Reference [160].

More recently, algorithms implementing ellipsoid growths for fitting cell have been presented by A. Alpers et al. [4] and improved by O. Sedivy et al. [150, 151]. These algorithms try to optimise an objective function encoding the positions and shape factors of all ellipsoids. The objective function is designed so to try to maximise ellipsoids volumes while avoiding voxels representing boundaries between crystal grains. Obviously, the optimisation problem becomes rapidly high-dimensional as the number of parameters to consider grows linearly with the number of ellipsoids. This issue has been tackled in Reference [151] with an heuristic approach. In this approach, only a subset of the parameters are optimised at each step, using simulated annealing.

Though faster, the algorithm proposed in Reference [151] still requires on the order of several million of iterations to converge. On the contrary, the optimisation procedure used in this thesis for the *Ellipsoidal Model* relies on the algorithm proposed by R. Deits et al. [39]. This algorithm maximises the volumes of the considered ellipsoids independently. As a consequence, it breaks down the high-dimensional optimisation problem of A. Alpers in a set of independent low-dimensional optimisation problems. Indeed, since each ellipsoid volume is independently maximised, only nine parameters (three for the position of the centre and six for the shape factors) need to be optimised. Therefore, the number of ellipsoids (and thus cells) that can be processed can be much higher than with the approaches of A. Alpers. and O. Sedivy. Moreover, the algorithm presented by R. Deits can be trivially parallelised. A drawback, though, of the approach of R. Deits. is that there is no guarantee that ellipsoids will not overlap. However, this drawback is not a concern here as subsequent steps do not depends on this peculiarity. Therefore, the approach proposed by R. Deits for growing ellipsoids has been chosen as it can process a large number of ellipsoids while being less computationally expensive than the approach proposed by O. Sedivy.

Results

Figure 2.34 illustrates the obtained ellipsoids for a 3D CT-scan image of size $670 \times 670 \times 670$ voxels obtained from a closed polypropylene foam provided by Erwan Plougonven (Department of Chemical Engineering / PEPs - Products, Environment, and Processes at University of Liège). It can been seen that indeed some cells contain several ellipsoids. This issue is tackled in the next section.

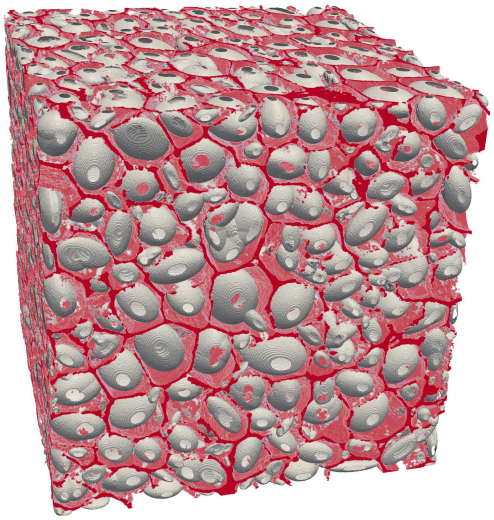


FIGURE 2.34: Red: walls of a $670 \times 670 \times 670$ 3D CT–scan image of a polypropylene foam provided by E. Plougonven, Department of Chemical Engineering at University of Liège. Gray: computed ellipsoids using the algorithm of R. Deits et al. [39]. It can be noticed that some cells contain several ellipsoids.

2.3.10 Step 5 (part 2): Clustering and merging of ellipsoids

For the purpose of identifying unequivocally each cell by one ellipsoid, it is needed to determine superfluous ellipsoids associated to superfluous local maxima. In order to merge them correctly, it is required to cluster together ellipsoids belonging to a same cell. Once clusters of ellipsoids identified, it is then possible to merge ellipsoids belonging to a same cluster and identify cells.

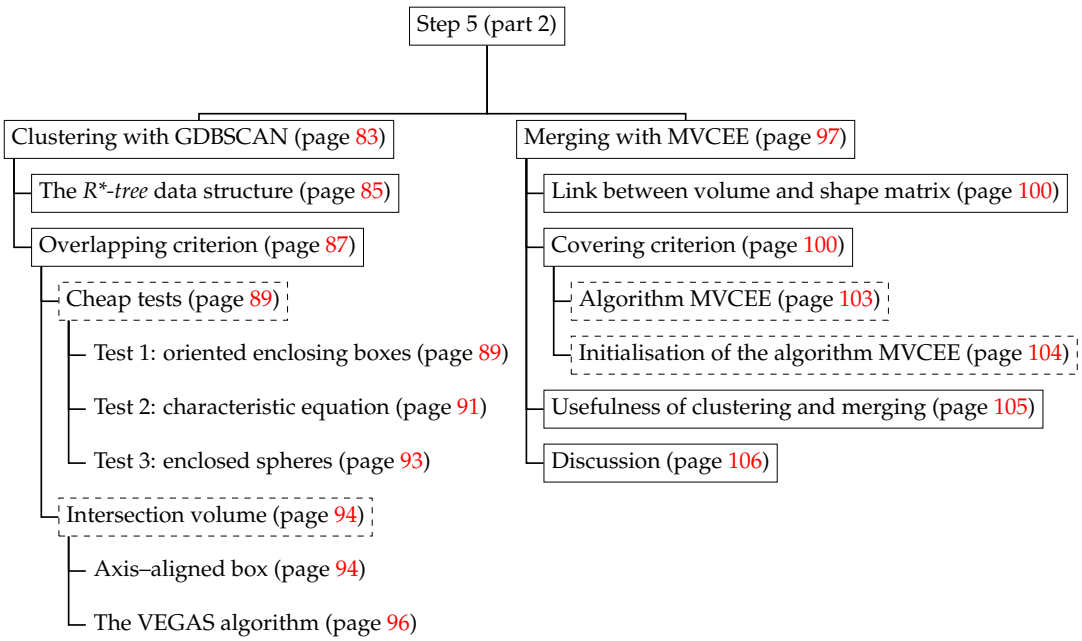
The criterion for deciding if two ellipsoids belong to a same cluster (and thus a same cell) is as follows: if two ellipsoids "significanly" overlap, then they are clustered together. Otherwise, they are identify as belonging to two different clusters. The idea behind is that, since ellipsoids were growth inside cells, ellipsoids belonging to a same cell (even from different starting local maxima) are more likely to overlap than ellipsoids grown in two different cells. Thus the criterion for associated two ellipsoids to the same cluster is based on their relative intersection volume. More precisely, two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ potentially associated to the same cell, are clustered together if their associated volume satisfy one of the relations 2.64.

$$\frac{Vol\left(\mathcal{E}_1 \cap \mathcal{E}_2\right)}{Vol\left(\mathcal{E}_1\right)} \geq \tau \text{ or } \frac{Vol\left(\mathcal{E}_1 \cap \mathcal{E}_2\right)}{Vol\left(\mathcal{E}_2\right)} \geq \tau \tag{2.64}$$

Where $\tau \in [0, 1]$ is the relative intersection volume threshold. When τ is set close to zero, any slightly intersecting pair of ellipsoids will be clustered together; while for τ set close to one, only pair of almost overlapping ellipsoids will be clustered together. In general, a suitable guess value for this parameter will be $\tau \approx 0.5$.

Step 5 (part 2) of the proposed image analysis procedure depicted in Figure 2.1b is organized in two main branches:

1. Clustering of ellipsoids identified as belonging to a common cell thanks to criterion 2.64.
2. Merging of ellipsoids belonging to a same cluster.



The *GDBSCAN* algorithm

The *Generalised Density Based SCAN* algorithm (*GDBSCAN*) of J. Sander et al. [147] is a generalisation of the *Density Based SCAN* algorithm (*DBSCAN*) of M. Ester et al. [45]. Its aim is to cluster a set of spatially distributed objects (here ellipsoids). The final purpose is to obtain a one-to-one correspondence between ellipsoids and cells by merging together ellipsoids identified as belonging to the same cluster.

Naturally, the *GDBSCAN* algorithm is not the only existing algorithm for performing clustering. As a matter of fact, there exists a huge variety of algorithms for performing this task. Two main classes of clustering algorithm can be distinguished [75]: partitioning and hierarchical algorithms. Partitioning algorithms, as the well-known *KNN* (K-Nearest Neighbours) algorithm [102], their weighted variants [11] and CLARANS (Clustering Large Application based on RANdomized Search [98]), try to optimise an objective function from an initial partition by determining cluster centres.

Hierarchical algorithms, such as the *GDBSCAN* algorithm of J. Sander et al. [147] or the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm of T. Zhang et al. [188], try to create a hierarchical tree of the objects to cluster by grouping them for a given criterion. Then, each level of the constructed tree gives a clustering of the objects.

Among all the above algorithms (and many others), the *GDBSCAN* algorithm has been chosen here for the following reasons[23]:

- It deals with spatially extended objects.
- It is versatile on its neighbourhood criterion, accepting as well spatial criteria (e.g. separation distance), as binary predicates (e.g. overlapping versus non-overlapping), and non-spatial criteria (e.g. colour).
- It has been designed to cluster "dense" (in some sense) sets of objects. This behaviour is desired, as ellipsoids covering the same cell will appear "denser" than their surrounding.
- Finally, it take into account "noise"[24].

On input, the *GDBSCAN* algorithm takes the grown ellipsoids obtained from previous section. In order to perform clustering, the *GDBSCAN algorithm* needs a neighbourhood criterion. The chosen neighbourhood criterion is here given by an overlapping threshold as specified by condition 2.64. Eventually, the data storage and query of the ellipsoids is provided by a R*-tree data structure [17].

At first sight, it may sound odd to use an algorithm such as *GDBSCAN* along a R*-tree data structure for performing clustering. Indeed, a R*-tree data structure already performs some kind of clustering by hierarchically sorting objects in a tree following an overlapping of enclosing boxes criterion (see Appendix G for details). Therefore one may ask what is the point of using an algorithm such as *GDBSCAN*. A first difference is that the neighbourhood criterion of a R*-tree data structure is *fixed*: ellipsoids (or any other objects) are clustered following the distances separating their enclosing boxes. The *GDBSCAN* algorithm may accept other clustering criteria. A second difference is that a R*-tree data structure will classify all the objects it gets as input, while the *GDSCAN* algorithm may *exclude* some of them (the "noisy" ones). In this respect the *GDBSCAN* algorithm is much more versatile and tunable than a simple R*-tree data structure and is able to achieve better classification results.

The GDBSCAN algorithm mechanism is as follows: it chooses the first unclassified object (i.e. an object not yet belonging to any cluster) and look at is neighbouring objects[25]. Neighbours satisfying the overlapping criterion 2.64 are set into the same cluster as the object. Then, neighbours are in their turn inspected to see if there is any of their neighbours satisfying the criterion 2.64. This is repeated until no neighbours to inspect remains. Then, the next unclassified object is selected for constructing the next cluster by inspecting its neighbours, until no unclassified object remains. The interested reader can find a more detailed description of the algorithm in Appendix F.

[23] Disclaimer: it does not mean that other algorithms than *GDBSCAN* are not suitable for the performing clustering of ellipsoids with noise, or even that the *GDBSCAN* algorithm is the best at this task. It just means that the *GDBSCAN* algorithm does the job well enough.

[24] The concept of noise in the context of clustering is given in Appendix F.

[25] As mentioned, neighbour querying is efficiently performed by sorting the objects in a R*-tree data structure.

The *R*-tree* data structure The *R*-tree* data structure used by the GDBSCAN algorithm[26] is a generalisation of the *R–tree* data structure of A. Guttman [59]. It is used in order to efficiently perform neighbour queries needed by the GDBSCAN algorithm. The following describes the *R*-tree* data structure as presented by N. Beckmann et al. in their article [17]. Figure 2.35 shows a schematic of how a tree data structure can be constructed from axis–aligned boxes containing objects (either ellipsoids, either other axis-aligned boxes).

The *R*-tree* algorithm of N. Beckmann et al. [17] builds a tree of axis-aligned boxes which combines optimisation (minimisation) of volume, overlap and margin. The *R*-tree* algorithm generalises the *R-tree* algorithm of A. Guttman [59] in the sense that the latter only optimises the volume of the axis-aligned boxes in the tree.

"Child nodes" axis-aligned boxes are constructed by the algorithm 27 and contain each one ellipsoid $\mathcal{E}$. Other "non–child nodes" axis-aligned boxes contain other axis-aligned boxes.

The *R*-tree* algorithm tries to minimise the following quantities:

1. *Volume of a "non–child" axis-aligned box*: allows to decide which paths in the tree should be traversed at a higher node.
2. *Overlap between axis-aligned boxes*: allows to decrease the number of paths to be traverses.
3. *Margin of a "non–child" axis-aligned box*. The *margin* of an axis-aligned box is the sum of the length of its edges. For a given volume, the optimal axis-aligned box is a cube. Minimising the *margin* will improves the structure of the tree by generating parent nodes with smaller volumes.

Moreover, as for R-trees, the shape of a R*-tree is driven by two other parameters M and m, with $2 \leq m \leq M/2$, and where:

1. M is the maximum number of child nodes that a node can have, except the root.
2. m is the minimum number of child nodes that a node can have, except the root.

Here, the following values have been chosen: $m = 2$ and $M = 8$. It has been indeed observed that, for each cell in average, there is between 1 and 10 associated ellipsoids.

Finally, a *R*-tree* satisfies the following properties [17]:

- The root has at least two children unless it is a child.
- Every non-child node has between m and M children unless it is the root.
- Every leaf node contains between m and M entries unless it is the root.
- All leaves appear on the same level.

[26] Note: other data structures can also be used by the GDBSCAN algorithm.

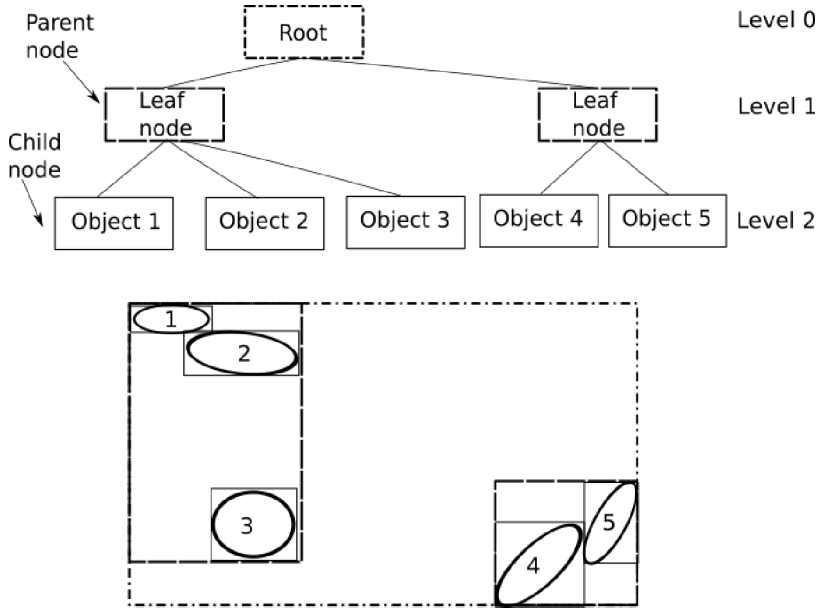


FIGURE 2.35: Schematic of a tree data-structure obtained from some ellipses in two dimensions.

The most important two sub-algorithms of the *R*-tree* algorithm when inserting a new entry are:

1. *ChooseSubTree*: find on every level the best subtree to accomodate the new entry.
2. *Split*: if *ChooseSubTree* ends in a node filled with the maximum M of allowed children, *Split* distributes $M + 1$ child nodes over two new parent nodes.

The *Split* algorithm 48 computes volume-values, overlap-values and margin-values over $M - 2m + 2$ distributions of the $M + 1$ entries. For $k = 1, \ldots, M - 2m + 2$, the first group contains the first $(m - 1) + k$ entries and the second group the remaining entries. The associated values to the kth distribution is then computed as follow:

- volume-value = volume[bb(first group)] + volume[bb(second group)].
- overlap-value = volume[bb(first group) $\cap$ bb(second group)].
- margin-value = margin[bb(first group)] + margin[bb(second group)].

Where bb denotes the bounding box of a set of boxes.

Detailed algorithms for handling a *R*-tree* data structure can be found in Appendix G.

Computation of the overlapping criterion (2.64)

In order to evaluate the overlapping criterion 2.64, the GDBSCAN algorithms needs to evaluate two quantities. Namely the volume of an ellipsoid $Vol(\mathcal{E}_i)$ and the volume of the intersection of two ellipsoids $Vol\left(\mathcal{E}_i \cap \mathcal{E}_j\right)$. While the volume of an ellipsoid can be computed analytically[27], the intersection volume of two ellipsoids can only be evaluated numerically.

As computing the intersection volume of two ellipsoids is rather expensive, the algorithm evaluating the criterion 2.64 (referenced hereafter as algorithm 22) first performs three "cheap" tests consisting in testing if the two ellipsoids are intersecting, and, if applicable, computing some upper and lower bounds for the criterion 2.64. If none of these "cheap" tests are passed, then a more expensive volume intersection algorithm is used. It should be noted that using the intersection volume between objects as (partial) criterion for determining whether they belong to the same cell or not is not a new idea. For instance, this criterion has been exploited by K. Mader et al [103], for removing what they call "artificial seeds" (roughly corresponding to our ellipsoids in this thesis).

The first "cheap" test is testing if the two ellipsoids are colliding or not. If not, there is no intersection volume and the criterion 2.64 is not satisfied (except if $\tau = 0$). The collision test is based on Reference [180] which is based of the signs of the roots of a "characteristic equation" constructed in homogeneous coordinates from the centres and shape matrices of the two considered ellipsoids. Details on how this "characteristic equation" is computed are given hereafter.

The second "cheap" test consists in computing the volume intersection $VolBoxInter$ of the boxes enclosing the two considered ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$. As the volume $VolBoxInter$ is bigger or equal to the intersection volume of the ellipsoids, there is no chance for the inequalities 2.64 to be satisfied if both ratios:

$$R_i = VolBoxInter/Vol(\mathcal{E}_i),\ i = 1,2 \tag{2.65}$$

are smaller than τ: $R_i < \tau,\ i = 1,2$.

The third "cheap" test resides in calculating the intersection volume of two enclosed spheres $S_1 \subset \mathcal{E}_1$ and $S_2 \subset \mathcal{E}_2$. As the intersection volume $Vol\left(S_1 \cap S_2\right)$ of the spheres S_1 and S_2 is smaller than the intersection volume $Vol\left(\mathcal{E}_1 \cap \mathcal{E}_2\right)$ of ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$, it is possible to infer a sufficient condition to satisfy criterion 2.64.

Finally, if all "cheap" tests fail, an expensive test is performed. The expensive test directly evaluates the intersection volume of two ellipsoids using a Monte-Carlo method. Algorithm 22 shows how these tests are used to determine if criterion 2.64 is satisfied.

[27] $Vol\left(\mathcal{E}_i\right) = 4\pi/3\,abc$, where a, b and c are the lengths of the semi-axes of the ellipsoid. These lengths can be computed from the eigenvalues of the shape matrix of the ellipsoid.

Algorithm 22 Predicate *NPred* for determining if two objects (ellipsoids) belong to the same cluster.

Require: $\mathcal{E}_1$ first ellipsoid belonging to the current cluster.
Require: $\mathcal{E}_2$ candidate ellipsoid for belonging to the same cluster as $\mathcal{E}_1$.
Require: τ overlapping volume ratio criterion as given in condition 2.64.
Require: M_{max} maximum number of integration points for estimating the overlapping volume.
Require: *tol* convergence tolerance for algorithm 28.

procedure NPRED($\mathcal{E}_1$, $\mathcal{E}_2$, τ, M_{max}, *tol*)
  ▷ Cheap test 1.
  if Not COLLISIONTEST($\mathcal{E}_1$, $\mathcal{E}_2$) **then** ▷ Algorithm 25.
    return false.
  end if
  ▷ Cheap test 2.
  $B_1 = \text{AXISALIGNEDBOX}(\mathcal{E}_1)$ ▷ Algorithm 27.
  $B_2 = \text{AXISALIGNEDBOX}(\mathcal{E}_2)$ ▷ Algorithm 27.
  $VolBoxInter = Vol\left(B_1 \cap B_2\right)$
  $R_1 = VolBoxInter/Vol(\mathcal{E}_1)$
  $R_2 = VolBoxInter/Vol(\mathcal{E}_2)$ ▷ Upper bounds for inequalities 2.64.
  if $R_1 < \tau$ and $R_2 < \tau$ **then return** false
  end if
  ▷ Cheap test 3.
  Extract r_1, r_2 the smallest semi–axes lengths of, respectively, $\mathcal{E}_1$ and $\mathcal{E}_2$.
  Extract $\mathbf{c}_1, \mathbf{c}_2$ the centres of, respectively, $\mathcal{E}_1$ and $\mathcal{E}_2$.
  Construct the spheres S_1, S_2 of respective centres $\mathbf{c}_1$ and $\mathbf{c}_2$ and radii r_1 and r_2.
  $d = \|\mathbf{c}_1 - \mathbf{c}_2\|_2$
  if $(r_1 + r_2) > d$ **then** ▷ Spheres S_1, S_2 are intersecting.
    if $r_1 \geq d + r_2$ or $r_2 \geq d + r_1$ **then** ▷ One sphere contains the other.
      return true
    else
      $VolSphereInter = \frac{\pi}{12d}\left(r_1 + r_2 - d\right)^2 \left(d^2 + 2d\left(r1 + r2\right) - 3\left(r1 - r2\right)^2\right)$
      $R_1 = VolSphereInter/Vol(\mathcal{E}_1)$
      $R_2 = VolSphereInter/Vol(\mathcal{E}_2)$ ▷ Lower bounds for inequalities 2.64.
      if $R_1 \geq \tau$ or $R_2 \geq \tau$ **then return** true
      end if
    end if
  end if
  ▷ Expensive test.
  $\{R_1, R_2\} = \text{ELLIPSOIDOVERLAPPINGRATIOS}(\mathcal{E}_1, \mathcal{E}_2, M_{max}, tol)$ ▷ Algorithm 28.
  return $R_1 \geq \tau$ or $R_2 \geq \tau$
end procedure

First "cheap" tests: collision detection between two ellipsoids. Given two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ their collision detection is performed in two steps. A first test involving enclosing boxes is achieved. This test is cheap from a computational point of view but it may fail as it represents only a necessary condition. If the first test fails, a second test, more computationally expensive, is performed. This test involves computing the roots of a so called "characteristic equation".

Test 1: oriented enclosing boxes. The two considered ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ are enclosed in oriented boxes: OB_i such that $\mathcal{E}_i \subset OB_i$, $i = 1, 2$. If the two boxes do not intersect ($OB_1 \cap OB_2 = \emptyset$), then there is no possible collision between $\mathcal{E}_1$ and $\mathcal{E}_2$. If the two boxes overlaps ($OB_1 \cap OB_2 \neq \emptyset$), then the two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ may or may not collide and a second test is necessary.

An oriented enclosing box is computed using the semi-axes lengths r_i; $i = 1, \ldots, n$ and directions $\mathbf{v}_i$; $i = 1, \ldots, n$ of the considered ellipsoid. These semi-axis lengths and vectors can be computed for the singular value decomposition (SVD) of the matrix G used in definition 2.5. Indeed, the semi-axis lengths $\{r_i\}_{i=1,\ldots,n}$ and directions $\{\mathbf{v}_i\}_{i=1,\ldots,n}$ can be extracted from the shape matrix M of the considered ellipsoid through eigenvalue decomposition:

$$M = U\Sigma^2 U^t \tag{2.66}$$

Where:

- $U \in \mathcal{R}^{n\times n}$ is an orthonormal matrix containing along its columns the semi-axis directions $\{\mathbf{v}_i\}_{i=1,\ldots,n}$.
- $\Sigma \in \mathcal{R}^{n\times n}$ is a diagonal matrix such that $\Sigma_{ii} = \lambda_i = 1/r_i > 0$; $i = 1, \ldots, n$, gives the semi-axis lengths.

As $GG^t = M$ and if $G = U_G \Sigma_G V_G^t$ is the SVD of G, then:

$$\begin{aligned} GG^t &= \left(U_G\Sigma_G V_G^t\right)\left(U_G\Sigma_G V_G^t\right)^t \\ &= \left(U_G\Sigma_G V_G^t\right)\left(V_G\Sigma_G U_G^t\right) \\ &= U_G\Sigma_G^2 U_G^t \quad \left(V_G^T V_G = Id\right) \\ &= M \\ &= U\Sigma^2 U^t \end{aligned}$$

$\Leftrightarrow U_G = U$ and $\Sigma_G = \Sigma$ (because the eigenvalues are all positives).

Therefore, the semi-axis directions can be obtained from the left singular vectors of G and their corresponding lengths from the singular values of G.

Construction of an oriented box. Thanks to the directions vectors $\{\mathbf{v}_i\}_{i=1,\ldots,n}$ and lengths $\{r_i\}_{i=1,\ldots,n}$, it is then possible to construct an oriented box enclosing the considered ellipsoid $\mathcal{E}$. Indeed, in the reference frame defined by point $\mathbf{c}$ and orthonormal vectors $\{\mathbf{v}_i\}_{i=1,\ldots,n}$, such a box is axis-aligned with the vectors $\{\mathbf{v}_i\}_{i=1,\ldots,n}$ and can be defined as $OB(\mathcal{E}) = [-r_1, r_1] \times \ldots \times [-r_n, r_n]$ in the eigenspace of the associated ellipsoid (see Figure 2.36 for a 2D example). Such an oriented enclosing box is constructed using the *GeometricTools* toolbox developed by P. J. Schneider and D H. Eberly [131].

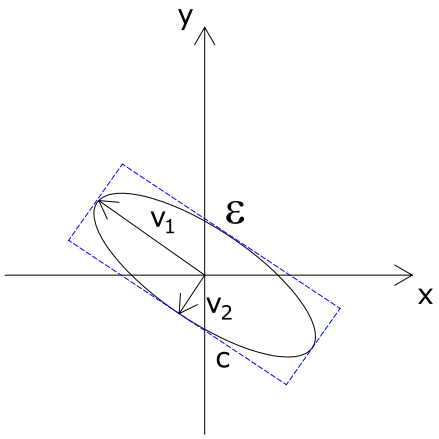


FIGURE 2.36: Oriented enclosing box of an ellipse.

In the framework of the *GeometricTools* toolbox, oriented boxes in $\mathcal{R}^n$ are defined by a centre $\mathbf{C} \in \mathcal{R}^n$, n right-handed orthonormal axes $\mathbf{A}_1, \ldots, \mathbf{A}_n \in \mathcal{R}^n$, and three corresponding extents $a_1, \ldots, a_n > 0$. Then, an oriented box in $\mathcal{R}^n$ can be represented as:

$$OB = \left\{ \mathbf{x} \in \mathcal{R}^n \mid \mathbf{C} + \sum_{i=1}^{n} y_i \mathbf{A}_i;\ |y_i| < |a_i|,\ i = 1, \ldots, n \right\} \tag{2.67}$$

Algorithm 23 Oriented enclosing box for ellipsoid

Require: Ellipsoid $\mathcal{E} = \left\{ \mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{c})^t G G^t (\mathbf{x} - \mathbf{c}) \leq 1 \right\}$

1: **procedure** ENCLOSINGBOX($\mathcal{E}$)
2: Compute SVD of $G = U_G \Sigma_G V_G^t$.
3: Extract left singular vectors $\{\mathbf{v}\}_{i=1,\ldots,n}$ from the columns of U_G.
4: Extract radii $\{r_i = 1/\lambda_i\}_{i=1,\ldots,n}$ from Σ_G.
5: Define oriented box in the eigenspace as $OB(\mathcal{E}) = [-r_1, r_1] \times \ldots \times [-r_n, r_n]$.
6: Alternatively, define oriented box by its centre $\mathbf{C} = \mathbf{c}$, axes $\mathbf{A}_i = \mathbf{v}_i$, $i = 1, \ldots, n$, and extents $a_i = r_i$, $i = 1, \ldots, n$.
7: **end procedure**

Separation of two oriented boxes in $\mathcal{R}^3$. In order to determine if two oriented boxes intersect or not, the procedure detailed in the documentation of *GeometricTools* toolbox [131] is applied, and is only given in Algorithm 24 for the sake of completeness.

Algorithm 24 Oriented box separation test [131]

Require: Two oriented boxes OB_1 and OB_2 in $\mathcal{R}^3$ defined by their respective centres $\mathbf{C}_1$ and $\mathbf{C}_2$, axes $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ and $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$, and corresponding extents $a_1, a_2, a_3 > 0$ and $b_1, b_2, b_3 > 0$.

procedure ORIENTEDBOXSEPARATIONTEST(OB_1, OB_2)
  $\mathbf{D} \leftarrow \mathbf{C}_2 - \mathbf{C}_1$
  for $\mathbf{L}$ is one of $\mathbf{A}_i, \mathbf{B}_i,\ (i = 1,2,3)$ or $\mathbf{A}_i \wedge \mathbf{B}_j,\ (i,j = 1,2,3)$ **do**
    $R \leftarrow |< \mathbf{L}|\mathbf{D} >|$
    $R_1 \leftarrow \sum_{i=1}^{3} a_i\ sgn\left(< \mathbf{L}|\mathbf{A}_i >\right) < \mathbf{L}|\mathbf{A}_i >$
    $R_2 \leftarrow \sum_{i=1}^{3} b_i\ sgn\left(< \mathbf{L}|\mathbf{B}_i >\right) < \mathbf{L}|\mathbf{B}_i >$
    if $R > R_1 + R_2$ **then**
      return True. ▷ Separation found. Boxes do not intersect.
    end if
  end for
  return False. ▷ Boxes are intersecting.
end procedure

Where:

$\wedge : \mathcal{R}^3 \times \mathcal{R}^3 \to \mathcal{R}^3, (\mathbf{A}, \mathbf{B}) \mapsto \mathbf{A} \wedge \mathbf{B}$ is the cross product.

$$sgn : \mathcal{R} \to \{-1, 0, 1\}, x \mapsto sgn(x) = \begin{cases} -1 \text{ if } x < 0 \\ 0 \text{ if } x = 0 \\ 1 \text{ elsewhere} \end{cases}$$

$< \ldots | \ldots >: \mathcal{R}^3 \times \mathcal{R}^3 \to \mathcal{R}, (\mathbf{A}, \mathbf{B}) \mapsto < \mathbf{A}|\mathbf{B} >$ is the Euclidean dot product.

Test 2: characteristic equation If the first test fails, to determine if the two ellipsoids do not collide, the following test is used. Before stating it, some definitions and properties about homogeneous coordinates are needed.

Ellipsoid in homogeneous coordinates Proposition 2.1 allows to define an ellipsoid in homogeneous coordinates and links it to definition 2.5.

Proposition 2.1. Given the representation of an ellipsoid $\mathcal{E} = \left\{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{c})^t M (\mathbf{x} - \mathbf{c}) \leq 1\right\}$ in the n-dimensional Euclidean affine space, its representation in homogeneous coordinates is $\mathcal{E} = \left\{\mathbf{x}_h \mid \mathbf{x}_h^t A_h \mathbf{x}_h \leq 0\right\}$.

Where: $A_h = \begin{pmatrix} M & -M\mathbf{c} \\ -\mathbf{c}^t M & \mathbf{c}^t M \mathbf{c} - 1 \end{pmatrix}$ and $\mathbf{x}_h = (\mathbf{x}^t, 1)^t$.

Proof.

$$\begin{aligned}
& \mathbf{x}_h^t A_h \mathbf{x}_h && \leq 0 \\
\Leftrightarrow \quad & \begin{pmatrix} \mathbf{x}^t & 1 \end{pmatrix} \begin{pmatrix} M & -M\mathbf{c} \\ -\mathbf{c}^t M & \mathbf{c}^t M \mathbf{c} - 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} && \leq 0 \\
\Leftrightarrow \quad & \begin{pmatrix} \mathbf{x}^t M - \mathbf{c}^t M & -\mathbf{x}^t M \mathbf{c} + \mathbf{c}^t M \mathbf{c} - 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} && \leq 0 \\
\Leftrightarrow \quad & \mathbf{x}^t M \mathbf{x} - \mathbf{c}^t M \mathbf{x} - \mathbf{x}^t M \mathbf{c} + \mathbf{c}^t M \mathbf{c} - 1 && \leq 0 \\
\Leftrightarrow \quad & (\mathbf{x} - \mathbf{c})^t M \mathbf{x} - (\mathbf{x} - \mathbf{c})^t M \mathbf{c} && \leq 1 \\
\Leftrightarrow \quad & (\mathbf{x} - \mathbf{c})^t M (\mathbf{x} - \mathbf{c}) && \leq 1
\end{aligned}$$

Which is definition 2.5. □

It is therefore possible to define the representation of an ellipsoid in homogeneous coordinates using definition 2.9.

Definition 2.9. The representation of a n-dimensional ellipsoid $\mathcal{E}$ in the homogeneous coordinates is:
$\mathcal{E} = \{\mathbf{x}_h \in \mathcal{R}^{n+1} \mid \mathbf{x}_h^t A_h \mathbf{x}_h \leq 0\}$

Where $\mathbf{x}_h = (\mathbf{x}, w)^t$ is written in homogeneous coordinates and $A_h \in \mathcal{R}^{(n+1)\times(n+1)}$ is a symmetric matrix.

Definition 2.10. (From W. Wang et al. [180])
Given two ellipsoids with homogeneous representation $\mathcal{E}_1 = \{\mathbf{x}_h \mid \mathbf{x}_h^t A_h \mathbf{x} \leq 0\}$ and $\mathcal{E}_2 = \{\mathbf{x}_h \mid \mathbf{x}_h^t B_h \mathbf{x} \leq 0\}$, their *characteristic polynomial* is defined as:

$f(\lambda) = det(\lambda A_h + B_h)$,
and $f(\lambda) = 0$ is called the *characteristic equation*.

The following property is the base of the second test for testing if two ellipsoids collide or not. Its complete proof can be found in [180].

Proposition 2.2. (From W. Wang et al. [180])
Let's A_h and B_h be two matrices representing respectively two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ in homogeneous coordinates. Then:

1. $\mathcal{E}_1$ and $\mathcal{E}_2$ are disjoint if and only if $f(\lambda) = 0$ has two distinct positive roots.
2. $\mathcal{E}_1$ and $\mathcal{E}_2$ touch each other externally if and only if $f(\lambda) = 0$ has a positive double root.

The test involving the characteristic equation $f(\lambda) = 0$ is thus the following: The homogeneous matrices A_h and B_h from ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ are computed. The root of the characteristic equation $f(\lambda) = 0$ are then found as the eigenvalues of the real eigenvalue problem $-A_h^{-1} B_h \mathbf{z}_h = \lambda \mathbf{z}$[28]. Indeed, if A_h is non–singular (i.e. ellipsoid $\mathcal{E}_1$ is non–degenerated), $det(\lambda A_h + B_h) = 0 \Leftrightarrow det(A_h) det(A_h^{-1} B_h + \lambda I) = 0 \Leftrightarrow det(-A_h^{-1} B_h - \lambda I) = 0$[29]. Then, if two roots are positive and distinct, the two ellipsoids do not collide. Otherwise they collide (possibly in only one point). Algorithm 25 summarises all the steps used for determining if two ellipsoids collide or not.

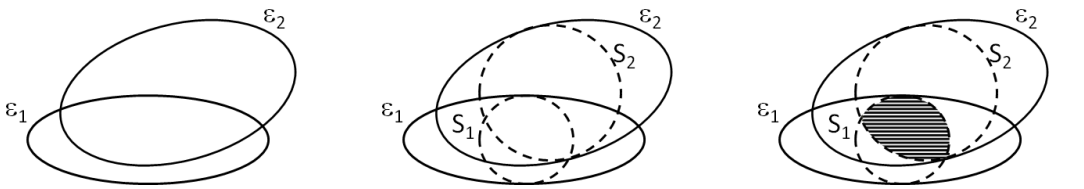
[28] The corresponding eigenvalues can be computed thanks to the routine *dgeev* of Lapack [5].

[29] It should be noted that, although A_h and B_h are symmetric matrices, $-A_h^{-1} B_h$ is generally not, because matrix–matrix multiplication is generally not commutative.

Algorithm 25 Two ellipsoids collision

Require: Two ellipsoids $\mathcal{E}_1, \mathcal{E}_2 \subset \mathcal{R}^3$.

procedure COLLISIONTEST($\mathcal{E}_1$, $\mathcal{E}_2$)
  ▷ Test 1: enclosing boxes
  Compute enclosing boxes $OB_1 \supset \mathcal{E}_1$ and $OB_2 \supset \mathcal{E}_2$. ▷ Algorithm 23
  if $OB_1 \cap OB_2 = \emptyset$ **then** ▷ Algorithm 24
    return false.
  else
    ▷ Test 2: characteristic equation
    Compute the homogeneous matrices A_h and B_h of $\mathcal{E}_1$ and $\mathcal{E}_2$. ▷ Prop. 2.1
    Compute the four eigenvalues $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in \mathcal{C}$ of the eigenvalue problem $-A_h^{-1} B_h \mathbf{z} = \lambda \mathbf{z}$
    if $\exists \lambda_i, \lambda_j \in \mathcal{R}$ and $\lambda_i, \lambda_j > 0$ **then** ▷ Prop. 2.2.
      return false.
    else
      return true.
    end if
  end if
end procedure



(a) Initial ellipsoids. (b) Enclosed corresponding spheres. (c) Computation of the intersection volume of the enclosed spheres.

FIGURE 2.37: Schematics illustrating test 3 (see text).

Test 3: volume intersection of enclosed spheres If the two first tests fail, this third "cheap" test resides in calculating the intersection volume of two enclosed spheres $S_1 \subset \mathcal{E}_1$ and $S_2 \subset \mathcal{E}_2$ (see Figure 2.37). These two spheres S_1 and S_2 are centred at their corresponding ellipsoid centres $\mathbf{c}_1$ and $\mathbf{c}_2$ and have radii equal to the smallest semi–axes lengths r_1 and r_2 of their corresponding ellipsoids (see Figure 2.37b). The intersection volume of S_1 with S_2 (see Figure 2.37c) is then given by the formula[30] 2.68.

$$VolSphereInter = \frac{\pi}{12d} \left(r_1 + r_2 - d\right)^2 \left(d^2 + 2d\left(r1 + r2\right) - 3\left(r1 - r2\right)^2\right) \tag{2.68}$$

Where $d = \|\mathbf{c}_1 - \mathbf{c}_2\|_2$ is the Euclidean distance separating the two spheres centres $\mathbf{c}_1$ and $\mathbf{c}_2$.

As $VolSphereInter$ is smaller or equal than the intersection volume of the ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$, at least one of the inequalities 2.64 is satisfied if $VolSphereInter / Vol(\mathcal{E}_1) \geq \tau$ or $VolSphereInter / Vol(\mathcal{E}_2) \geq \tau$ is satisfied.

[30] Source: http://mathworld.wolfram.com/Sphere-SphereIntersection.html.

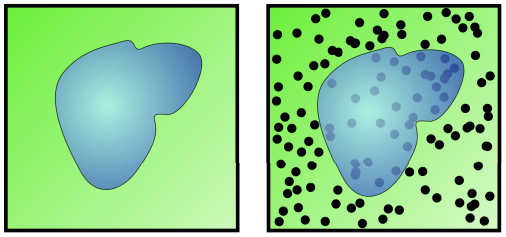


FIGURE 2.38: Schematic illustrating the principle of the Monte-Carlo method, see text (images from Wikipedia https://fr.wikipedia.org/wiki/Méthode_de_Monte-Carlo).

Computation of the intersection volume of two ellipsoids. Criterion 2.64 requires the computation of the intersection volume of two ellipsoids. In general, this volume is not an ellipsoid and can not be readily computed (i.e. with an analytical formula). Instead, a Monte-Carlo technique is here used in order to estimate this volume.

The principle of the Monte-Carlo for estimating an area/volume/hyper-volume of a geometrically non-trivial object is as follows (see Figure 2.38): given a non-trivial object $O_{complex}$ (in blue in the figure) and an enclosing object $O_{enclosing}$ (in green in the figure) whose area/volume can be computed analytically; sample N random points over the enclosing object. Count the number X of points inside the non-trivial object. Then the an estimation of the area/volume of the non-trivial object is given in equation 2.69.

$$Area\left(O_{complex}\right) \approx \frac{N-X}{X} Area\left(O_{enclosing}\right) \tag{2.69}$$

Construction of an enclosing axis-aligned box. In order to compute the intersection volume between two ellipsoids, the so-called *VEGAS* algorithm of G.P. Lepage [129] is used here. This algorithm has been implemented by M. Booth in the *GNU Scientific Library* (GSL) [52]. Like any Monte-Carlo algorithm, the algorithm of G.P. Lepage requires an integration domain separable along each dimension for which the volume is easily (i.e. analytically) computable. This requirement is indeed mandatory for estimating the volume of two intersecting ellipsoids using an equation of the form 2.69[31]. It is therefore necessary to construct such a domain from two ellipsoids.

This domain is obtained as the intersection of two axis-aligned boxes[32] from two colliding ellipsoids. An axis-aligned box $B(\mathcal{E})$ containing an ellipsoid $\mathcal{E}$ is constructed from the orthogonal projections of the ellipsoids on the axes of the canonical frame of reference. These projections are computed as suggested by S. Pope and the following paragraphs are essentially a rewriting of the section 12 of its report [133].

[31] Typically, it is only needed to replace the word *Area* by the word *Volume*.
[32] Which is itself an axis-aligned box.

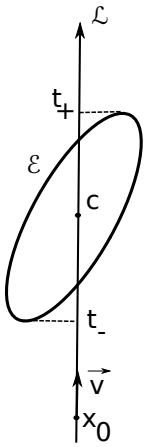


FIGURE 2.39: Schematic example of an orthogonal projection of an ellipse $\mathcal{E}$ on a line $\mathcal{L}$. (Figure taken from S. Pope [133] and slightly modified).

Definition 2.11. A *line* $\mathcal{L} \subset \mathcal{R}^n$ parametrised by $t \in \mathcal{R}$ and defined as:

$$\mathcal{L} = \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{x}_0 + t\mathbf{v}\} \tag{2.70}$$

Where $\mathbf{x}_0 \in \mathcal{R}^n$ is a point on the line $\mathcal{L}$ and $\mathbf{v} \in \mathcal{R}^n$ is a unit–length vector.

Definition 2.12. For any point $\mathbf{x} \in \mathcal{R}^n$, its *orthogonal projection* on the line $\mathcal{L}$ is a point on $\mathcal{L}$ with associated parameter t such that:

$$t = \mathbf{v}^t (\mathbf{x} - \mathbf{x}_0) \tag{2.71}$$

Considering now an ellipsoid $\mathcal{E} = \mathcal{E}(\mathbf{c}, G) \subset \mathbf{R}^n$ of centre $\mathbf{c} \in \mathcal{R}^n$ and its associated matrix $G \in \mathcal{R}^{n \times n}$ as given by definition 2.5, the orthogonal projections of the points of ellipsoid $\mathcal{E}$ on the line $\mathcal{L}$ can be obtained through the parameter t, with:

$$t = \mathbf{v}^t \left(\mathbf{c} + G^{-t}\mathbf{y} - \mathbf{x}_0\right), \ \|\mathbf{y}\| \leq 1 \tag{2.72}$$

$$= t_0 + \mathbf{w}^t\mathbf{y}, \ \|\mathbf{y}\| \leq 1 \tag{2.73}$$

Where $t_0 = \mathbf{v}^t (\mathbf{c} - \mathbf{x}_0)$ and $\mathbf{w} = G^{-1}\mathbf{v}$.

Given equation 2.73, it can be found that the orthogonal projection of ellipsoid $\mathcal{E}$ on the line $\mathcal{L}$ corresponds to parameters:

$$t_\pm = t_0 \pm \|\mathbf{w}\| \tag{2.74}$$

Figure 2.39 illustrates the above equations, while algorithm 26 implements them.

Algorithm 26 Orthogonal projection of an ellipsoid on a line (S. Pope [133]).

Require: Line $\mathcal{L} = \mathcal{L}(\mathbf{x}_0, \mathbf{v}) = \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{x}_0 + t\mathbf{v},\ \|\mathbf{v}\| = 1\}$.
Require: Ellipsoid $\mathcal{E} = \mathcal{E}(\mathbf{c}, G) = \left\{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1\right\}$.
procedure EllipsoidOnLineProjection($\mathcal{L}$, $\mathcal{E}$)
  $t_0 = \mathbf{v}^t\,(\mathbf{c} - \mathbf{x}_0)$.
  $\mathbf{w} = G^{-1}\mathbf{v}$.
  $t_\pm = t_0 \pm \|\mathbf{w}\|$.
  return $\mathbf{x}_\pm = \mathbf{x}_0 + t_\pm\mathbf{v}$.
end procedure

From the orthogonal projections of an ellipsoid $\mathcal{E}$ on the axis of the canonical reference frame, it is then possible to construct an enclosing axis-aligned box of the ellipsoid, as described by the algorithm 27. From algorithm 27 it should be noted that the returned box is indeed centred at the centre of the enclosed ellipsoid, even if this centre is not the origin.

Algorithm 27 Axis-aligned bounding box for an ellipsoid.

Require: Ellipsoid $\mathcal{E} = \mathcal{E}(\mathbf{c}, G) = \left\{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1\right\}$.
procedure AxisAlignedBox($\mathcal{E}$)
  for $i = 1, \ldots, n$ **do**
    Construct $\mathbf{e}_i$, the ith canonical vector which is zero everywhere, except in the ith position where it is one.
    Construct the line $\mathcal{L}_i = \mathcal{L}_i(\mathbf{c}, \mathbf{e}_i)$ passing through the ellipsoid's centre $\mathbf{c}$ and oriented following $\mathbf{e}_i$.
    $x_i^\pm$ = EllipsoidOnLineProjection($\mathcal{L}_i$, $\mathcal{E}$) ▷ Algorithm 26.
  end for
  return box as $B(\mathcal{E}) = [x_1^-, x_1^+] \times \ldots \times [x_n^-, x_n^+]$.
end procedure

Computation of the volume intersection Thanks to the VEGAS algorithm 56 (see Appendix H for details) it is now possible to compute the volume intersection $Vol(\mathcal{E}_1 \cap \mathcal{E}_2)$ of two colliding ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$. As the VEGAS algorithm only estimates the integral of a given function, the following algorithm 28 uses repeatedly the VEGAS algorithm with an increasing number of integration points until convergence. On convergence, the algorithm 28 returns the overlapping ratios R_1 and R_2 as defined in equations 2.64.

Algorithm 28 Overlapping ratios of two colliding ellipsoids.

Require: Two colliding ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$.
Require: Number of initial integration points M_{init}.
Require: Convergence tolerance tol.

procedure EllipsoidOverlappingRatios($\mathcal{E}_1$, $\mathcal{E}_2$, M_{init}, tol)
  $M \leftarrow M_{init}$
  $f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$ as defined in equation H.1
  $AAB(\mathcal{E}_i) \leftarrow$ AxisAlignedBox($\mathcal{E}_i$) $i = 1, 2$ ▷ Algoritm 27.
  $\Omega \leftarrow AAB(\mathcal{E}_1) \cap AAB(\mathcal{E}_2)$ ▷ Integration domain.
  $N \leftarrow M/n^3$.
  $Vol(\mathcal{E}_1 \cap \mathcal{E}_1) \leftarrow$ VEGAS($f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, M, Ω, N) ▷ Algorithm 56.
  $R_i \leftarrow Vol(\mathcal{E}_1 \cap \mathcal{E}_1)/Vol(\mathcal{E}_i)$, $i = 1, 2$
  repeat
    $R_{old,i} \leftarrow R_i$, $i = 1, 2$
    $M \leftarrow 2M$
    $N \leftarrow M/n^3$
    $Vol(\mathcal{E}_1 \cap \mathcal{E}_1) \leftarrow$ VEGAS($f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, M, Ω, N) ▷ Algorithm 56.
    $R_i \leftarrow Vol(\mathcal{E}_1 \cap \mathcal{E}_1)/Vol(\mathcal{E}_i)$, $i = 1, 2$
  until $|R_i - R_{old,i}| \leq tol|R_i|$, $i = 1, 2$
  return $\{R_1, R_2\}$
end procedure

Merging of clustered ellipsoids

From here, the GDBSCAN algorithm has clustered (hopefully correctly) "sufficiently" overlapping ellipsoids, so that there is an unequivocally correspondence between clusters of ellipsoids and the cells of the considered foam. Ellipsoids clustered together are merged into one ellipsoid using the *Minimum Volume Covering Ellipsoid of Ellipsoids* (MVCEE) algorithm given in Reference [185].

Essentially, the MVCEE algorithm works as follows:

1. Given a set of ellipsoids $\{\mathcal{E}_i\}_{1\leq i\leq n}$ and an initial covering ellipsoid $\tilde{\mathcal{E}}\left(\tilde{\mathbf{c}}, \tilde{M}\right)$ of centre $\tilde{\mathbf{c}}$ and shape matrix $\tilde{M}$ (Figure 2.40a), the furthest point $\mathbf{x}_1$ belonging to $\{\mathcal{E}_i\}_{1\leq i\leq n}$ from the centre $\tilde{\mathbf{c}}$ is computed (Figure 2.40b).

2. Then, the centre $\tilde{\mathbf{c}}$ and the shape matrix $\tilde{M}$ of the covering ellipsoid $\tilde{\mathcal{E}}$ are updated so to minimise the distance of $\mathbf{x}_1$ to the boundary of $\tilde{\mathcal{E}}$ (Figure 2.40c).

3. Afterwards, a new furthest point $\mathbf{x}_2$ in $\{\mathcal{E}_i\}_{1\leq i\leq n}$ from centre $\tilde{\mathbf{c}}$ is computed (Figure 2.40d).

4. A new update of the centre $\tilde{\mathbf{c}}$ and the shape matrix $\tilde{M}$ is performed by minimising the distances to $\mathbf{x}_1$ and $\mathbf{x}_2$ to the boundary of $\tilde{\mathcal{E}}$. This process is repeated by adding new furthest points $\mathbf{x}_3, \ldots, \mathbf{x}_m$ until the ellipsoid does not vary any more (within given threshold) when adding new furthest points.

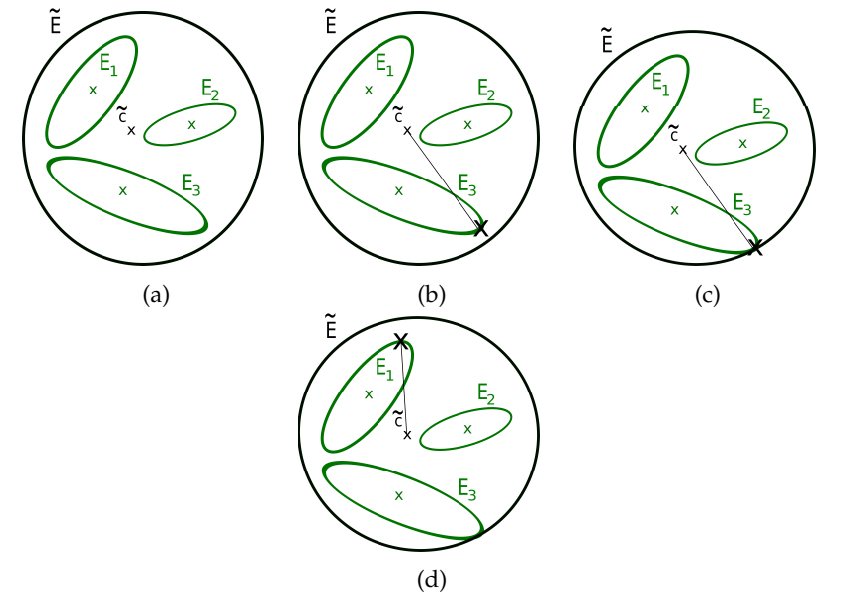


FIGURE 2.40: Schematic sketching how the MVCEE algorithm [185] is operating. See text for details.

Figure 2.41 shows a three-dimensional example of the MVCEE of three ellipsoids, while Figure 2.42 shows the obtained ellipsoids for a 3D CT-scan image of size $670 \times 670 \times 670$ voxels obtained from a closed polypropylene foam provided by Erwan Plougonven (Department of Chemical Engineering / PEPs - Products, Environment, and Processes at University of Liège). Compared to Figure 2.34, it can be seen that ellipsoids belonging to common cells have been merged into single ellipsoids.

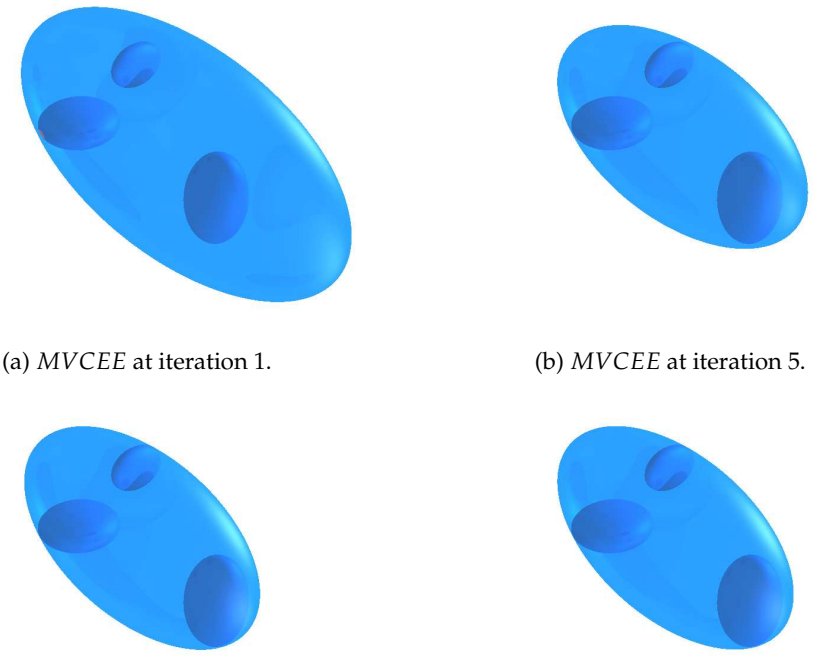
From here, it is assumed that clustered ellipsoids are provided in a form of a search tree, such an R*-tree. Sets of ellipsoids belonging to a same cluster are then merged together into a single *Minimum Volume Covering Ellipsoid of Ellipsoids* (MVCEE), thank to an algorithm developed by E.A. Yildirim [185].

Given a set of m full-dimensional (i.e. non–degenerated) ellipsoids $\mathcal{E}_1, \ldots \mathcal{E}_m \in \mathcal{R}^n$, let's S denotes their convex hull. The merging of the m ellipsoids $\mathcal{E}_1, \ldots \mathcal{E}_m$ will be performed by searching (an approximation to) the *Minimum Volume Covering Ellipsoid* (MVCE) of S, denoted by $MVCE(S)$. If S has a non-empty interior, the MVCE is guaranteed to exists and is unique [50].

Given a finite set of ellipsoids $S = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\}$, the problem is here to find the minimum volume covering ellipsoid of these ellipsoids (MVCEE) as stated by problem 2.75.

$$\begin{cases} min_{M,\mathbf{c}} Vol(\mathcal{E}) \\ \text{s.t. } \mathcal{E} \supseteq \{\mathcal{E}_1, \ldots, \mathcal{E}_m\} \end{cases} \tag{2.75}$$

Where $\mathcal{E}$, with representation given by definition 2.5, is the MVCEE of ellipsoids $\mathcal{E}_i, < i = 1, \ldots, m$.



(a) *MVCEE* at iteration 1.

(b) *MVCEE* at iteration 5.

(c) *MVCEE* at iteration 10.

(d) *MVCEE* at iteration 20.

FIGURE 2.41: *MVCEE* algorithm 29 used on a three-dimensional set of three ellipsoids (solids) at different iterations. The *MVCEEs* are showed in semi-transparent blue. (Images generated using *POV-Ray* [29]).

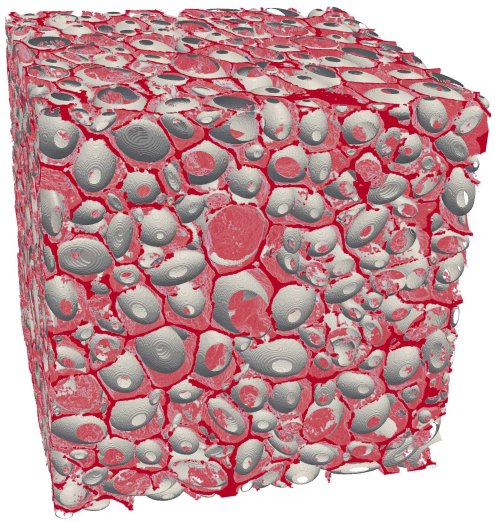


FIGURE 2.42: Red: walls of a $670 \times 670 \times 670$ 3D CT-scan image of a polypropylene foam provided by E. Plougonven, Department of Chemical Engineering at University of Liège. gray: clustered and merged ellipsoids by MVCEE. It can be noticed, by comparing to Figure 2.34, that some cells that used to contain several ellipsoids, now contain only one ellipsoid.

Link between $Vol(\mathcal{E})$ and its shape matrix M. Problem 2.75 can be restated in terms of the matrix M as follows:

The volume $Vol(\mathcal{E})$ of an ellipsoid $\mathcal{E} \subseteq \mathcal{R}^3$ is given by:

$$Vol(\mathcal{E}) = \frac{4\pi}{3} r_1 r_2 r_3 \tag{2.76}$$

Where the $r_i,\ i = 1,2,3$ are the semi-axis lengths of the considered ellipsoid.

These semi-axis lengths $r_i,\ i = 1,2,3$ are related to the eigenvalues $\lambda_i,\ i = 1,2,3$ of the corresponding shape matrix M as follow:

$$r_i = \frac{1}{\sqrt{\lambda_i}},\ i = 1,2,3 \tag{2.77}$$

Leading, for a fully dimensional ellipsoid (i.e. positive definite shape matrix $M \succ 0$), to the result:

$$\begin{array}{rl} & det(M) = \lambda_1\lambda_2\lambda_3 = \frac{1}{r_1^2 r_2^2 r_3^2} \\ \Leftrightarrow & det(M^{-1}) = \frac{1}{\lambda_1\lambda_2\lambda_3} = r_1^2 r_2^2 r_3^2 \end{array} \tag{2.78}$$

The problem 2.75 can be restated as:

$$\begin{cases} min_{M,\mathbf{c}}\ ln\left(det\left(M^{-1}\right)\right) \\ \text{s.t.} \quad M \succ 0, \text{ symmetric} \\ \qquad\ \mathcal{E} \supseteq \{\mathcal{E}_1, \ldots \mathcal{E}_m\} \end{cases} \tag{2.79}$$

Covering criterion. In order to determine if an ellipsoid $\mathcal{E}$ is covering another ellipsoid $\mathcal{E}_i,\ i \in \{1, \ldots, m\}$, E.A. Yildirim suggests to use the following two propositions:

Proposition 2.3. (Proposition 2.7 in E.A. Yildirim [185])
Let's $\mathcal{E} \subset \mathcal{R}^n$ and $\mathcal{E}_i \subset \mathcal{R}^n$ denote two full-dimensional ellipsoids, then $\mathcal{E}_i \subseteq \mathcal{E}$ if and only if the exists $\tau > 0$ such that:

$$\tau \begin{pmatrix} M_i & -M_i\mathbf{c}_i \\ -\mathbf{c}_i^t M_i & \mathbf{c}_i^t M_i \mathbf{c}_i - 1 \end{pmatrix} \succeq \begin{pmatrix} M & -M\mathbf{c} \\ -\mathbf{c}^t M & \mathbf{c}^t M \mathbf{c} - 1 \end{pmatrix} \tag{2.80}$$

Where the symbol "$\succeq$" in an expression of type "$A \succeq B$" stands for $A - B \succeq 0$, meaning that the matrix $A - B$ is positive semi–definite.

Proposition 2.4. (Lemma 2.8 in E.A. Yildirim [185])
Condition 2.80 is equivalent to:

$$\tau \begin{pmatrix} M_i & -M_i\mathbf{c}_i & 0 \\ -\mathbf{c}_i^t M_i & \mathbf{c}_i^t M_i \mathbf{c}_i - 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \succeq \begin{pmatrix} M & -M\mathbf{c} & 0 \\ -\mathbf{c}^t M & -1 & \mathbf{c}^t M \\ 0 & M\mathbf{c} & -M \end{pmatrix} \tag{2.81}$$

Using condition 2.81 for all ellipsoids $\mathcal{E}_i$, $i = 1, \ldots, m$, the problem 2.79 can be restated as:

$$\left\{ \begin{array}{l} min_{M,\mathbf{c}}\ ln\left(det\left(M^{-1}\right)\right) \\ \quad M \succ 0, \text{ symmetric} \\ \quad \exists\, \tau_1 > 0, \ldots, \tau_m > 0 \\ \text{s.t.} \quad \begin{pmatrix} M & -M\mathbf{c} & 0 \\ -\mathbf{c}^t M & -1 & \mathbf{c}^t M \\ 0 & M\mathbf{c} & -M \end{pmatrix} - \tau_i \begin{pmatrix} M_i & -M_i\mathbf{c}_i & 0 \\ -\mathbf{c}_i^t M_i & \mathbf{c}_i^t M_i \mathbf{c}_i - 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \preceq 0 \end{array} \right. \tag{2.82}$$

Finally, by letting $\mathbf{z} = M\mathbf{c}$ the problem 2.82 reads:

$$\left\{ \begin{array}{l} min_{M,\mathbf{z}}\ ln\left(det\left(M^{-1}\right)\right) \\ \quad M \succ 0, \text{ symmetric} \\ \quad \exists\, \tau_1 > 0, \ldots, \tau_m > 0 \\ \text{s.t.} \quad \begin{pmatrix} M & -\mathbf{z} & 0 \\ -\mathbf{z}^t & -1 & \mathbf{z}^t \\ 0 & \mathbf{z} & -M \end{pmatrix} - \tau_i \begin{pmatrix} M_i & -M_i\mathbf{c}_i & 0 \\ -\mathbf{c}_i^t M_i & \mathbf{c}_i^t M_i \mathbf{c}_i - 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \preceq 0 \end{array} \right. \tag{2.83}$$

Which is (almost) a semi-definite program.

As pointed-out by E.A. Yildirim, the problem 2.83 may be solved by interior-point algorithm. However, the computational cost of these algorithms becomes quickly prohibitive as the dimension of the problem grows. Instead, E.A. Yildirim suggests the algorithm 29 for solving approximatively the problem 2.83.

Algorithm 29 Algorithm for solving problem 2.83 (E.A. Yildirim [185])

Require: Set of ellipsoids $\mathcal{E}_1, \ldots, \mathcal{E}_m \subset \mathcal{R}^n$.
Require: $\epsilon > 0$.
procedure MVCEE($\mathcal{E}_1, \ldots, \mathcal{E}_m, \epsilon$)
  $\chi_0 = \{\mathbf{x}^1, \ldots \mathbf{x}^{2n}\} \leftarrow$ MVCEEINITIALISE($\mathcal{E}_1, \ldots, \mathcal{E}_m$) ▷ Algorithm 30
  $\mathbf{u}^0 \leftarrow (1/(2n))\mathbf{e} \in \mathcal{R}^{2n}$ ▷ $\mathbf{e}$: vector of ones.
  $\mathbf{w}^0 \leftarrow \sum_{j=1}^{2n} \mathbf{x}^j u_j^0$
  $(M^0)^{-1} \leftarrow n \sum_{j=1}^{2n} u_j^0 (\mathbf{x}^j - \mathbf{w}^0)(\mathbf{x}^j - \mathbf{w}^0)^t$
  $\mathcal{F}^0 \leftarrow \left\{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{w}^0)^t M^0 (\mathbf{x} - \mathbf{w}^0) \leq 1\right\}$
  $\mathbf{x}^{2n+1} \leftarrow argmax_{i=1,\ldots,m} \left\{(\mathbf{x} - \mathbf{w}^0)^t M^0 (\mathbf{x} - \mathbf{w}^0) \mid \mathbf{x} \in \mathcal{E}_i\right\}$
  $\epsilon_0 \leftarrow (\mathbf{x}^{2n+1} - \mathbf{w}^0)^t M^0 (\mathbf{x}^{2n+1} - \mathbf{w}^0) - 1$
  $k \leftarrow 0$
  while $\epsilon_k > (1+\epsilon)^{2/n} - 1$ **do**
    $\beta_k \leftarrow \frac{\epsilon_k}{(n+1)(1+\epsilon_k)}$
    $k \leftarrow k + 1$
    $\mathbf{u}^k \leftarrow \begin{pmatrix} (1-\beta_{k-1})\mathbf{u}^{k-1} \\ \beta_{k-1} \end{pmatrix}$
    $\mathbf{w}^k \leftarrow \sum_{j=1}^{2n+k} \mathbf{x}^k u_j^k$
    $(M^k)^{-1} \leftarrow n \sum_{j=1}^{2n+k} u_j^k (\mathbf{x}^j - \mathbf{w}^k)(\mathbf{x}^j - \mathbf{w}^k)^t$
    $\mathcal{F}_k \leftarrow \left\{\mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{w}^k)^t M^k (\mathbf{x} - \mathbf{w}^k) \leq 1\right\}$
    $\chi_k \leftarrow \chi_{k-1} \cup \{\mathbf{x}^{2n+k}\}$
    $\mathbf{x}^{2n+k+1} \leftarrow argmax_{i=1,\ldots,m} \left\{(\mathbf{x} - \mathbf{w}^k)^t M^k (\mathbf{x} - \mathbf{w}^k) \mid \mathbf{x} \in \mathcal{E}_i\right\}$
    $\epsilon_k \leftarrow (\mathbf{x}^{2n+k+1} - \mathbf{w}^k)^t M^k (\mathbf{x}^{2n+k+1} - \mathbf{w}^k) - 1$
  end while
  return $\sqrt{1+\epsilon_k}\mathcal{F}_k, \chi_k$
end procedure

Algorithm 30 Initialisation for algorithm 29 (E.A. Yildirim [185])

Require: Set of ellipsoids $\mathcal{E}_1, \ldots, \mathcal{E}_m \subset mathcal R^n$.
procedure MVCEEINITIALISE($\mathcal{E}_1, \ldots, \mathcal{E}_m$)
  $\psi \leftarrow \{0\}$, $\chi_0 \leftarrow \emptyset$, $k \leftarrow 0$
  while $\mathcal{R}^n \backslash \psi \neq \emptyset$ **do**
    $k \leftarrow k + 1$
    Pick an arbitrary unit vector $\mathbf{b}^k \in \mathcal{R}^n$ in the orthogonal complement of ψ
    $\mathbf{x}^{2k-1} \leftarrow argmax_{1=1,\ldots,m} \left\{(\mathbf{b}^k)^t \mathbf{x} \mid \mathbf{x} \in \mathcal{E}_i\right\}$, $\chi_0 \leftarrow \chi_0 \cup \{\mathbf{x}^{2k-1}\}$
    $\mathbf{x}^{2k} \leftarrow argmin_{i=1,\ldots,m} \left\{(\mathbf{b}^k)^t \mathbf{x} \mid \mathbf{x} \in \mathcal{E}_i\right\}$, $\chi_0 \leftarrow \chi_0 \cup \{\mathbf{x}^{2k}\}$
    $\psi \leftarrow span\left(\psi, \{\mathbf{x}^{2k-1} - \mathbf{x}^{2k}\}\right)$
  end while
  return χ_0
end procedure

Remarks about algorithm 29 $MVCEE$.

1. From line 2 to line 6, the algorithm computes from algorithm 30 a first estimate $\mathcal{F}^0$ of the $MVCEE(\mathcal{E}_1,\ldots,\mathcal{E}_m)$. This first ellipsoid $\mathcal{F}^0$ is simply centred on the centre of mass of points provided by χ_0 (line 4) while its shape matrix M^0 is simply (up to a scaling factor) the inverse covariance matrix of those points (line 5).

2. On line 3, $\mathbf{u}^0$ simply contains the weights for computing the (weighted) centre of mass and will be iteratively updated for computing the final $MVCEE$ centre.

3. Once the first ellipsoid $\mathcal{F}^0$ constructed, the furthest point $\mathbf{x}^{2n+1}$ from $\mathcal{F}^0$ belonging to $\{\mathcal{E}_1,\ldots\mathcal{E}_m\}$ is computed on line 7. This quadratic constrained quadratic problem is solved using the *Minpack-2* package [9].

4. On line 8, ϵ_0 "measure" how far from ellipsoid $\mathcal{F}^0$ is the furthest point $\mathbf{x}^{2n+1}$ computed on line 7.

5. Next, the loop on line 10 will repeat the above steps for iteratively updating the ellipsoid $\mathcal{F}^0$. The centre of ellipsoid $\mathcal{F}^0$ is iteratively updated by computing new (weighted) centres (line 14) based of the updated set of points χ_k (line 17); where χ_k is iteratively enriched with the furthest point $\mathbf{x}^{2n+k+1}$ from the current ellipsoid $\mathcal{F}^0$ belonging to the set $\{\mathcal{E}_1,\ldots,\mathcal{E}_m\}$. The shape matrix M^0 of ellipsoid $\mathcal{F}^0$ is also iteratively updated by computing the (scaled) inverse covariance matrix of the points in χ_k (line 15).

6. Lines 11 to 13 update the weights used in the computations of the current centre $\mathbf{w}^k$ and current shape matrix M^k of the current ellipsoid $\mathcal{F}^k$. These updates are based on the "measures" of distances from the current farthest point $\mathbf{x}^{2n+k}$ of the current ellipsoid $\mathcal{F}^k$.

7. Line 18 requires to solve a quadratic problem subjects to quadratic constraints. As this problem is identical to the problem on line 7, it is also solved using the *Minpack-2* package [9].

8. Finally, algorithm 29 returns a dilated version of the found ellipsoid around its centre (line 21). For an ellipsoid $\mathcal{E}\subset\mathcal{R}^3$ of shape matrix M, its semi-axis lengths $r_i,\ 1=1,2,3$ can be uniformely scaled by a factor $|a|$ if the shape matrix M is multiplied by a factor $\alpha=1/a^2$. Indeed, as the shape matrix M is real symmetric definite positive, its eigen decomposition is given by:

$$M = U\Sigma^2U^t$$

Where U is unitary and Σ is a diagonal matrix holding the eigenvalues $\lambda_i,\ i=1,2,3$.

Multiplying M by $\alpha=1/a^2$ leads to:

$$\alpha M = \alpha U\Sigma^2U^t = U(\sqrt{\alpha}\Sigma)^2U^t = U\tilde{\Sigma}^2U^t$$

Where $\tilde{\Sigma}$ now contains the eigenvalues $\tilde{\lambda}_i=\sqrt{\alpha}\lambda_i=\lambda_i/|a|,\ i=1,2,3$.

As the following relation between the semi–axes lengths $r_i,\ 1 = 1,2,3$ and the corresponding eigenvalues $\lambda_i,\ 1 = 1,2,3$ holds:

$$r_i = \frac{1}{\lambda_i},\ i = 1,2,3$$

One obtain:

$$\tilde{r}_i = \frac{1}{\tilde{\lambda}_i} = \frac{|a|}{\lambda_i} = |a| r_i,\ i = 1,2,3$$

i.e., the ellipsoid $\mathcal{E}$ is indeed dilated uniformly by a factor $|a|$ around its centre.

Remarks about algorithm 30 *MVCEEInitialise*.

1. The idea of algorithm 30 is to pick arbitrary linearly independent lines and projects the ellipsoids $\mathcal{E}_i,\ i = 1,\ldots,m$ on them (see Figure 2.43). For a given line, these projections form a set of segments. In this set of segments, lines 6 and 7 aim to find the two furthest points from each other. These points are then added to the set χ_0 such that the $MVCE(\chi_0)$ will be a first rough approximation of of $MVCEE(\mathcal{E}_1,\ldots,\mathcal{E}_m)$.

2. Each segment coming from the projection of ellipsoid $\mathcal{E}_i$ on line of direction $\mathbf{b}^k$ has extremities given by

$$\tilde{\mathbf{x}}^{i,k}_{max,min} = \mathbf{c}^i \pm \left(1/\left\|(U^i)^{-t}\mathbf{b}^k\right\|\right)(U^i)^{-1}(U^i)^{-t}\mathbf{b}^k$$

with corresponding values [185]:

$$(\mathbf{b}^k)^t\mathbf{c}^i \pm \left(1/\left\|(U^i)^{-t}\mathbf{b}^k\right\|\right)(\mathbf{b}^k)^t(U^i)^{-1}(U^i)^{-t}\mathbf{b}^k$$

Where $M^i = (U^i)^t U^i,\ i = 1,\ldots,m$ denotes the upper Cholesky factorisation of $M^i,\ i = 1,\ldots,m$. An exhaustive search among a list of m values allows to solve optimisation problems of lines 6 and 7.

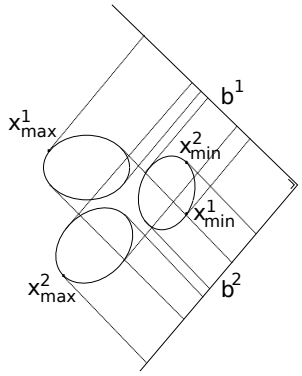


FIGURE 2.43: Sketch of the *MVCEEInitialise* algorithm 30 in two dimensions. Ellipsoids are projected along two arbitrary orthogonal directions $\mathbf{b}^1$ and $\mathbf{b}^2$. Minimum and maximum ellipsoid projections $\mathbf{x^i_{min}}$ and $\mathbf{x}^i_{max},\ i = 1,2$ along these two directions are computed.

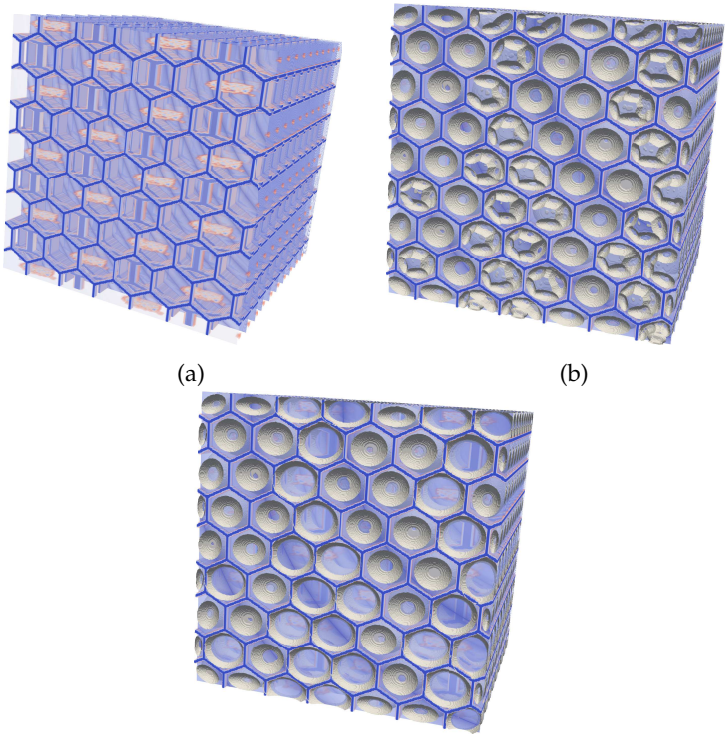


FIGURE 2.44: (a): Artificially generated honeycomb foam with groups of spurious voxels. (b): Obtained ellipsoids without merging. (c): Obtained ellipsoids with merging. It can be seen that merging of ellipsoids helps to obtain a univocal correspondence between cells and ellipsoids.

Usefulness of clustering and merging ellipsoids. One may ask if clustering and merging the ellipsoids is really useful, especially when using auxiliary ellipsoids (see Section 2.3.11). Indeed, auxiliary ellipsoids will grow in all cells and reproduce the same microstructure regardless if they are several parent ellipsoids or a single parent ellipsoid in a given cell. The reason for merging parent ellipsoids is twofold:

1. One may want to know some statistics about the cell in the considered foam. Among them, some important statistic are the number of cells, their mean number of neighbours and the presence of a preferred cell orientation, which can be estimated from the ellipsoids only after merging.

2. At this stage of the processing chain, spurious voxels may still be present, despite the use "cleaning" filters such as the *Box filter* (see Section 2.3.5), or any other image processing filter available in the ITK library (such as the *BinaryMorphologicalOpeningImageFilter* filter) that a user may take advantage of. This is particularly true if spurious voxels form groups that are hard to erase without discarding useful voxels. Clustering and merging of ellipsoids may help to dispose of such groups of spurious voxels by merging parent ellipsoids neighbouring these groups. Figure 2.44 illustrates this on the case of an artificially generated honeycomb foam.

Discussion: GDBSCAN and merging quality assessment

The GDBSCAN algorithm depends on the four following parameters:

- *MinCard* the cluster volume threshold which has been set to the volume of one voxel. This means that any cluster whose volume is smaller than one voxel will be discarded.
- $\tau \in [0,1]$ the overlapping volume ratio given in condition 2.64. When the relative intersection volume of two intersecting ellipsoids is above τ, these two ellipsoids are considered as belonging to the same cluster.
- M_{max} the maximum number of integration point used by the Monte–Carlo method for computing the intersection volume between two ellipsoids (see algorithm 28). This parameters has been set to $M_{max} = 2^{15} = 32768$ integration points.
- *tol* the convergence tolerance used by the Monte–Carlo method for computing the intersection volume between two ellipsoids (see algorithm 28). This parameter has been set to $tol = 0.1$.

Among the above four parameters, the only non-trivial one is the overlapping volume ratio τ. As τ is the main parameter for controlling the merging of ellipsoids, it is important to assess which values are suitable for it. In order to determine a suitable range of values for τ, the following criterion has been used.

Criterion for overlapping volume ratio τ. In the GDBSCAN algorithm take τ such that the *Minimum Volume Covering Ellipsoid of Ellipsoids* (MVCEE) belonging to a same cluster satisfies criterion 2.84 when τ is slightly varied.

$$MatchPixels(\tau)/MismatchPixels(\tau) \approx constant \tag{2.84}$$

Where $MatchPixels(\tau)$ is the number of pixels belonging to the MVCEEs of the computed clusters and identified as *non-feature* pixels (i.e. not belonging to cell-boundaries), and $MismatchPixels(\tau)$ are pixels also belonging to the MVCEEs of the computed clusters but identified as *feature* pixels (i.e. belonging to cell boundaries). Note that, in general tomographic images, some non–feature pixels may be identified as feature ones and vice-versa. Here, these pixels are referenced as *noisy* pixels. However, it has been observed that the GDBSCAN algorithm 43 is rather tolerant to it.

It is believed that criterion 2.84 provides suitable values of τ for obtaining satisfactory clusterings. Indeed, for $\tau = 1$ the GDBSCAN algorithm only cluster ellipsoids containing other ellipsoids (the volume overlapping ratio has to be 100%), and for $\tau = 0$ this algorithm clusters any intersecting ellipsoids (the volume overlapping ratio is 0%). More generally, a value of τ close to one will prevents the clustering of two ellipsoids associated to two different cells, while it will also prevents the clustering of intersecting ellipsoids belonging to the same cell. Conversely, a value of τ close to zero will allow the clustering of intersecting ellipsoids belonging to the same cell, but will also allow the clustering of intersecting ellipsoids belonging to different cells. This may happen if the wall separating the two considered cells contains openings[33].

[33] At least on the provided tomographic images. This may occurs for closed but very thin walls.

Starting from $\tau = 0$ (see Figure 2.45, box region 1) and increasing it, clustered ellipsoids belonging to well separated cells will be isolated (because their intersection volume ratios should be inexistent), while ellipsoids belonging to the same cell will remain clustered (since their intersection volume ratios should be positive, see Figure 2.46c). With τ increasing from 0, this will cause the $MatchPixels(\tau)$ to decrease slower than the $MismatchPixels(\tau)$ function because MVCEEs will contain less pixels wrongly identified as non–feature. As outcome, the ratio given in criterion 2.84 will slightly raise.

At intermediate values of τ (figure 2.45, box region 2) the ratio given in criterion 2.84 will stay approximatively constant (or will slightly increase) as clusters of ellipsoids will remain approximatively stable (or will slightly raise).

An intermediate value of τ should thus trigger the algorithm GDBSCAN to cluster only intersecting ellipsoids belonging to the same cell in order to obtain the final purpose of this section: achieve a one-to-one correspondence between ellipsoids and cells (see Figure 2.46d).

At $\tau \approx 1$ (Figure 2.45, box region 3), intersecting ellipsoids belonging to a same cell will not be clustered. The resulting MVCEEs will contain some noisy feature pixels (pixels wrongly identified as belonging to cell boundaries) that were separating the initial ellipsoids (see Figure 2.46e). This will cause the $MatchPixels(\tau)$ function to stay approximatively constant, while the $MismatchPixels(\tau)$ function will quickly decrease. As a consequence, the ratio in criterion 2.84 will rapidly raise. In general, the curve giving the ratio $MatchPixel(\tau)/MismatchPixel(\tau)$ versus τ will have the general shape given in schematic 2.45.

It should be noted that, at this stage, intersecting ellipsoids belonging to different cells should, in principle, not be clustered. Indeed, if a cell wall (even with openings) separates the two considered ellipsoids, their intersection volume ratios should be small and not satisfying criterion 2.84. However, cell walls with big openings may accidentally satisfy this criterion and trigger the clustering of the two considered ellipsoids; though it may then be questionable to see these considered cells as two separated cells (at least for closed foams).

In order to illustrate the criterion 2.84, a dataset D of ellipsoids has been generated from the set of 3D–image provided by E. Plougonven (department of Applied Chemistry, University of Liège). Several values of the intersecting volume ratios τ between zero and one with a step of 0.05 have been tested and plotted against the ratio $MatchPixels(\tau)/MismatchPixels(\tau)$ in Figure 2.47. It can be seen that the graph of this figure is very similar to the one given in schematic 2.45. From figure 2.47 it can be inferred that a value of $\tau \in [0.1, 0.4]$ is suitable for giving a satisfactory clustering. For $\tau = 0.1$, the ratio $MatchPixels(0.1)/\text{total number of pixels} = 0.77$ and $MismathcPixels(0.1)/\text{total number of pixels} = 0.37$, with 1664 clusters found.

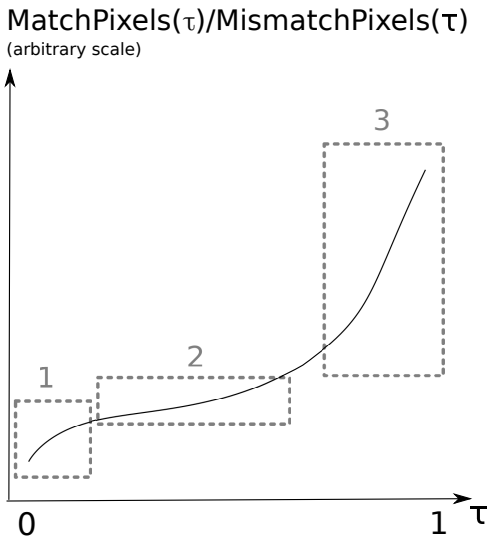
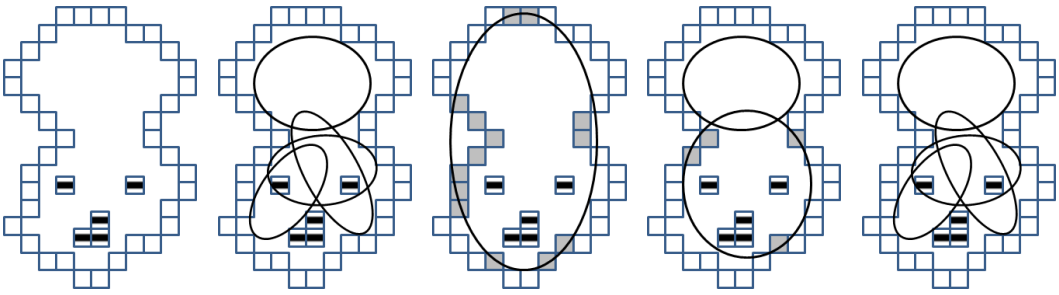


FIGURE 2.45: Schematic of the general behaviour of the ratio $MatchPixels(\tau)/MismatchPixels(\tau)$ with respect to the intersection volume ratio τ (see criterion 2.64). $MatchPixels(\tau)$ is the number of non-feature pixels contained in the MVCEEs (see section 2.3.10) of the clustering computed with parameter τ, while $MismatchPixels(\tau)$ is the number of feature pixels contained in the same MVCEEs. See text for comments about the different regions shown.



(a) Two cells with incomplete separating wall.

(b) Grown ellipsoids inside cells before merging.

(c) Merged ellipsoids with $\tau = 0$. $\frac{MatchPixels}{MismatchPixels} = \frac{5}{14} \approx 0.35$.

(d) Merged ellipsoids with $\tau = 0.5$. $\frac{MatchPixels}{MismatchPixels} = \frac{5}{4} \approx 1.25$.

(e) Merged ellipsoids with $\tau = 1.0$. $\frac{MatchPixels}{MismatchPixels} = \frac{2}{0} = \infty$.

FIGURE 2.46: Schematics illustrating the general behaviour of the ratio $MatchPixels(\tau)/MismatchPixels(\tau)$ (see text). Empty pixels represent feature pixels (walls), greyed pixels represent mismatched pixels, pixels with a line represent non–feature/noisy pixels. Black lines represent ellipsoids.

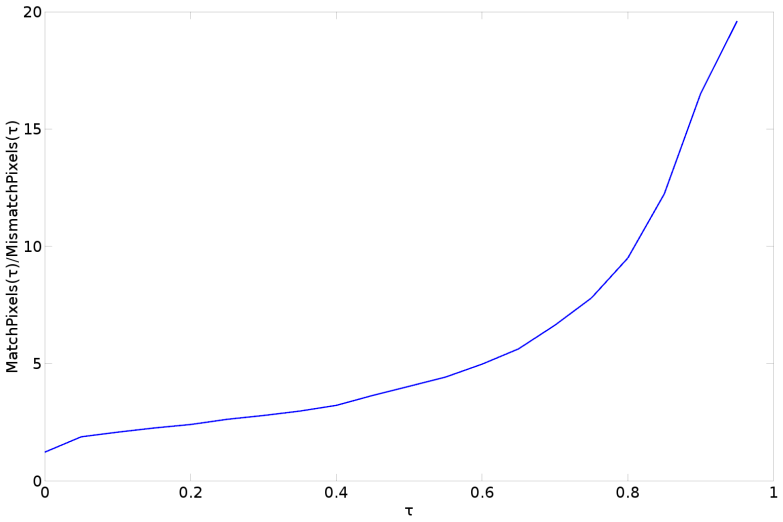


FIGURE 2.47: Behaviour of the ratio $MatchPixels(\tau)/MismatchPixels(\tau)$ with respect to the intersection volume ratio τ (see criterion 2.64) for the 3D–image provided by E. Plougonven (department of Applied Chemistry, University of Liège). $MatchPixels(\tau)$ is the number of non-feature pixels contained in the MVCEEs of the clustering computed with parameter τ, while $MismatchPixels(\tau)$ is the number of feature pixels contained in the same MVCEEs.

2.3.11 Step 6: Auxiliary ellipsoids

Auxiliary ellipsoids are grown using the previously expanded ellipsoids for generating new seeds. These auxiliary ellipsoids are constructed as follows (see Figure 2.48): each previously grew ellipsoid is uniformly discretised into a set of points and serves as parent to its associated auxiliary ellipsoids. This discretisation is parametrised by an azimuthal angle increment $\Delta\theta$, which drives, in turn, the polar angle increment

$$\Delta\phi_i = \frac{2\pi}{\lfloor 2\pi sin(\theta_i)/\theta_i \rfloor + 1} \tag{2.85}$$

Where $\theta_i = \theta_{i-1} + \Delta\theta$ is the ith increment in angle θ, and $\lfloor . \rfloor$ returns the closest lower integer of its argument.

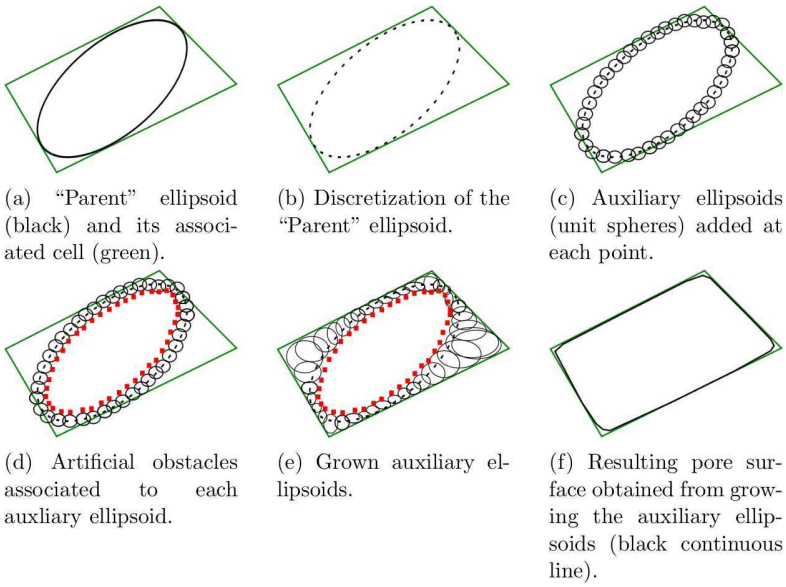
In practice, the discretisation is computed as follows: by using (Equation 2.86), the discretisation is computed as given by algorithm 31. In practice the factor α is chosen to a value slightly lower than 1 (e.g. $\alpha = 0.99$), in order to avoid having the seldom case of a point located on a feature pixel (in which case, the associated auxiliary ellipsoid could never be grown).

$$\mathcal{E}_\alpha = \left\{ \mathbf{x} \in \mathcal{R}^3 \,|\, \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq \alpha \right\} \tag{2.86}$$

Algorithm 31 Given a parent ellipsoid $\mathcal{E}$, an azimuthal angle increment $\Delta\theta$, and a factor α, uniformly discretises the iso-surface of value α of $\mathcal{E}$ in a set of points.

Require: Ellipsoid $\mathcal{E} = \left\{ \mathbf{x} \in \mathcal{R}^3 \,|\, \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1 \right\}$, angle increment $\Delta\theta$, factor α.

1: **procedure** DISCRETIZEELLIPSOID($\mathcal{E}$, $\Delta\theta$, α)
2: Set of points $P \leftarrow \emptyset$.
3: **for** $\theta_i = 0; \theta_i \leq \pi; \theta_i \leftarrow \theta_i + \Delta\theta$ **do**
4: Compute $\Delta\phi_i$ as given by eq. 2.85.
5: **for** $\phi_{ij} = 0; \phi_{ij} \leq 2\pi; \phi_{ij} = \phi_{ij} + \Delta\phi_i$ **do**
6: Compute $\mathbf{y} = (y_0, y_1, y_2)$ as:
7: $$\begin{cases} y_0 &= \alpha \sin\theta_i \cos\phi_{ij} \\ y_1 &= \alpha \sin\theta_i \sin\phi_{ij} \\ y_2 &= \alpha \cos\theta_i \end{cases}.$$
8: $P \leftarrow P \cup \left\{ \mathbf{c} + G^{-t}\mathbf{y} \right\}$.
9: **end for**
10: **end for**
11: **return** P.
12: **end procedure**



(a) "Parent" ellipsoid (black) and its associated cell (green).

(b) Discretization of the "Parent" ellipsoid.

(c) Auxiliary ellipsoids (unit spheres) added at each point.

(d) Artificial obstacles associated to each auxliary ellipsoid.

(e) Grown auxiliary ellipsoids.

(f) Resulting pore surface obtained from growing the auxiliary ellipsoids (black continuous line).

FIGURE 2.48: Schematic sequence illustrating the growth of auxiliary ellipsoids into a cell. The last step f. is obtained by taking the zero iso-surface (iso-line) of the distance field to the ellipsoids.

Once the discretisation computed, each of the associated point is attached to an auxiliary ellipsoid. Initially, these auxiliary ellipsoids are simply spheres the radii of which are one voxel wide. Then, between each sphere and the centre of the parent ellipsoids an "artificial" obstacle is added. These obstacles are simply the closest points of each sphere to the centre of the parent ellipsoid. The aim of these "artificial" obstacles is to avoid the auxiliary ellipsoids to all decay into their parent ellipsoids. Once initialised by unit spheres and their associated artificial obstacles set, the auxiliary ellipsoids are then grown using the same algorithm as their parent ellipsoids [39], taking into account for each auxiliary ellipsoid the same obstacles as for its associated parent ellipsoid and the artificial ones generated from it and the spheres. The result shown in Figure 2.48f is obtained by extracting the zero iso-surface of the distance field to the ellipsoids. Algorithm 32 details the whole process. The result of this process is illustrated in Figure 2.49 where it can be seen that auxiliary ellipsoids match the cell walls much closer than the associated parent ellipsoids.

Algorithm 32 Given a parent ellipsoid, compute the associated auxiliary ellipsoids and their associated obstacles.

Require: Image I of domain D_I.
Require: Pixel value $p \in Im(I)$ figuring the value associated to the cell walls.
Require: Parent ellipsoid $\mathcal{E}$, angle increment $\Delta\theta$, factor α.

procedure AUXILIARYELLIPSOIDS(I, p, $\mathcal{E}$, $\Delta\theta$, α)
  $E_{aux} \leftarrow$ INITIALAUXILIARYELLIPSOIDS($\mathcal{E}$, $\Delta\theta$, α) ▷ Algo. 33.
  for $\mathcal{E}_{aux,i} \in E_{aux}$ **do**
    $\mathcal{O}_{artif} \leftarrow$ COMPUTEARTIFICIALOBSTACLE($\mathcal{E}$, $\mathcal{E}_{aux,i}$) ▷ Algo. 34.
    $\mathcal{E}_{aux,i} \leftarrow$ ELLIPSOIDEXPAND(I, p, $\mathcal{E}_{aux,i}$, $\mathcal{O}_{artif}$) ▷ Algo. 13.
  end for
  return E_{aux}.
end procedure

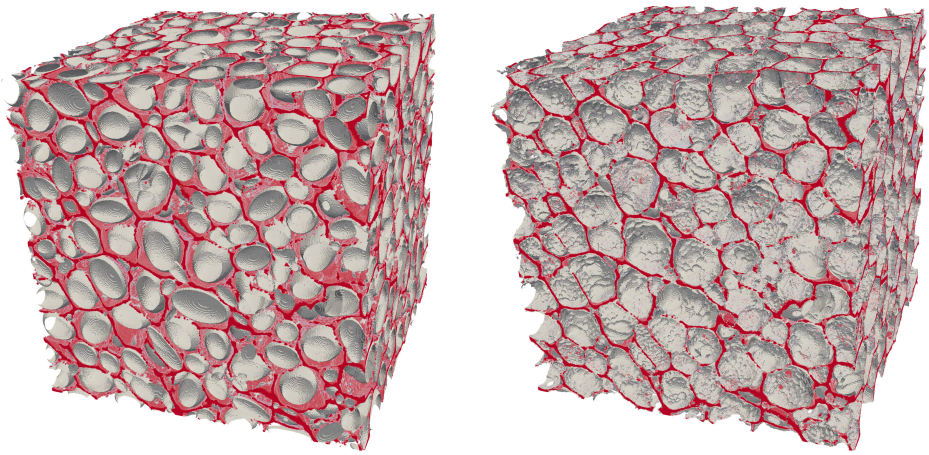
Algorithm 33 Constructs initial auxiliary ellipsoids from a parent ellipsoid $\mathcal{E}$.

Require: Parent ellipsoid $\mathcal{E}$, angle increment $\Delta\theta$, factor α.
1: **procedure** INITIALAUXILIARYELLIPSOIDS($\mathcal{E}$, $\Delta\theta$, α)
2: Set au auxiliary ellipsoids $E_{aux} \leftarrow \emptyset$
3: $P \leftarrow$ DISCRETZIZEELLIPSOID($\mathcal{E}$, $\Delta\theta$, α) ▷ Algo. 31.
4: **for** $P_i \in P$ **do**
5: $\mathcal{E}_{aux,i} = \left\{\mathbf{x} \in \mathcal{R}^3 \,|\, \mathbf{x} = P_i + Id\,\mathbf{y},\ \|\mathbf{y}\| \leq 1\right\}$ ▷ $Id \in \mathcal{R}^{3\times3}$, identity matrix.
6: $E_{aux} \leftarrow E_{aux} \cup \{\mathcal{E}_{aux,i}\}$.
7: **end for**
8: **return** E_{aux}.
9: **end procedure**

Algorithm 34 Given a parent ellipsoid and an auxiliary ellipsoid, compute an artificial obstacle.

Require: Parent ellipsoid $\mathcal{E} = \left\{\mathbf{x} \in \mathcal{R}^3 \,|\, \mathbf{x} = \mathbf{c} + G^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1\right\}$.
Require: Auxiliary ellipsoid $\mathcal{E}_{aux} = \left\{\mathbf{x} \in \mathcal{R}^3 \,|\, \mathbf{x} = \mathbf{c}_{aux} + G_{aux}^{-t}\mathbf{y},\ \|\mathbf{y}\| \leq 1\right\}$.
1: **procedure** COMPUTEARTIFICIALOBSTACLE($\mathcal{E}$, $\mathcal{E}_{aux}$)
2: $\mathbf{v} = (\mathbf{c} - \mathbf{c}_{aux}) \,/\, \|\mathbf{c} - \mathbf{c}_{aux}\|$.
3: $\mathcal{O} \leftarrow \left\{\mathbf{c}_{aux} + G_{aux}^{-t}\mathbf{v}\right\}$.
4: **return** $\mathcal{O}$.
5: **end procedure**

From all the above described steps, an approximation of a CT-scan image of foam in terms of a set of ellipsoids (and associated polyhedra) can be obtained. This set can then be used to reconstruct the geometry of the foam; geometry that can be subsequently used into a FEM simulation as it will be demonstrated later on in this thesis. These steps avoid the memory expensive watershed and H-maxima transforms, replacing them by the clustering and merging of overlapping ellipsoids. Depending on the number of auxiliary ellipsoids associated to each parent ellipsoid, the obtained set of ellipsoids allows a rather precise fitting of the struts structure present in the CT-scan images, including local defects such as partially missing or deformed struts, as it will be shown further in this thesis.

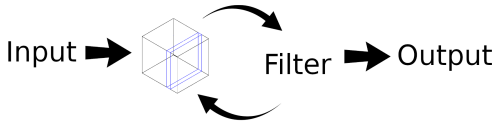


(a) Parent ellipsoids. (b) Auxiliary ellipsoids.

FIGURE 2.49: Parent ellipsoids and surface obtained by the auxiliary ellipsoids using an angle increment $\Delta\theta = 10^{\circ}$. 3D-image (in red) provided by E. Plougonven (department of Applied Chemistry, University of Liège).

Chapter 3

Streaming



- **Aim**: shows how to adapt the filters presented in the preceding chapters for processing data slice by slice.
- **Input**: depends on algorithm that has been streamed.
- **Output**: depends on algorithm that has been streamed.

Streaming is the ability to process an image by slices. It offers the advantage to be able to process large amounts of data that are not manageable otherwise (e.g. that do not fit the RAM of a computer, and/or parallelise the processing). Indeed, 3–dimensional μ-CT imaging systems may easily generate datasets of the order of 10 giga–bytes of more (typically $1024 \times 1024 \times 1024$ images with 64–bits voxels). The Insight Toolkit framework [69] offers such streaming capabilities [194]. All the steps described in Section 2.3 have been implemented as ITK filters which are streamable along slices (see Figure 3.2).

3.1 Summary

The following chapter is structured as follow:

- Section 3.2 gives the framework used for implementing streaming.
- Sections from 3.3 to 3.10 describe how the different steps of the image analysis procedure given in chapter 2 have been adapted to support streaming.
- Section 3.10.1 quantitatively show that ellipsoids found using the "normal" processing steps and the streamed ones are identical to numerical precision.
- Section 3.12 finally shows that streaming indeed allows the processing of 3D CT-scans that do not fit into the available RAM.

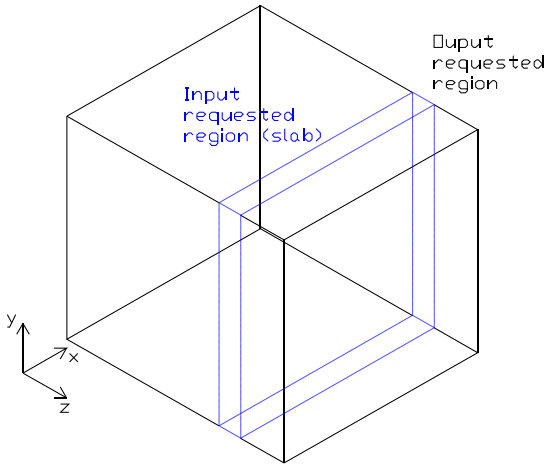


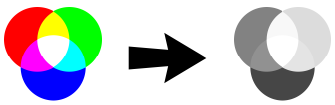
FIGURE 3.2: Extraction of a slice from a 3D image.

3.2 The *itk::StreamingFilter*

The *itk::StreamingFilter* filter is a customised version of the *itk::StreamingImageFilter* in the sense that is requests specials regions to the upstream filters in order to accommodate the requirements of the *itk::MaurerDistanceMapFilter*

These special regions take the form of slabs (see Figure 3.2) that traverses the whole output region requested by the downstream filter in one given direction. When the streaming gets the output requested region, it partitions the given output region into these slabs. Then, the upstream filters are called on each subregions through an overridden version of the *UpdateOutputdata()* method. More details on how the ITK library implements streaming can be found in Appendix A

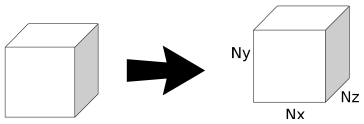
3.3 Streaming of the RGB to luminance filter (step 0)



- **Aim**: convert a RGB (Red, Green, Blue) image to a gray–level image.
- **Input**: RGB image.
- **Output**: gray–level image.
- **Original filter**: see Section 2.3.2.

The original RGB to luminance filter converts RGB images to gray–level images. As this filter operates on images voxel by voxel, it is trivially streamable without any modification.

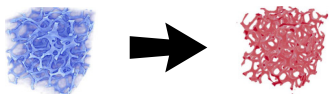
3.4 Streaming of the spacing filter (step 0bis)



- **Aim**: add spacing information at the voxels of an image.
- **Input**: RGB or gray–level image.
- **Output**: RGB or gray–level image with spacing information.
- **Original filter**: see Section 2.3.3.

The spacing filter sets the lengths of the sides of each voxel in an image. It allows for taking into account acquired images with anisotropic resolutions[1] along the x, y and z direction. As this filter only update meta information about an image, it is trivially streamable without any modification.

3.5 Streaming of the threshold filter (step 1)



- **Aim**: convert a gray–level image into a binary (black and white) image.
- **Input**: gray–level image.
- **Output**: binary image.
- **Original filter**: see Section 2.3.4.

Usually, threshold filters process image voxel by voxel, and are therefore trivially streamable. However, the algorithm of Ridler and Calvard algorithm [139] used in this thesis needs the computation of an histogram over the whole image, which may not fit the RAM (see procedure *ComputeHistogram* at line 24 of algorithm 3). This problem is solved by setting up two pipelines, and performing streaming on both of them. The first pipeline (hereafter called "temporary" pipeline) is used to accumulate the histogram. This is done by calling several times the inner loop of procedure *ComputeHistogram* (Algorithm 3) during the streaming. Once the temporary pipeline has been updated and the histogram computed, the second pipeline (hereafter called "main" pipeline) is updated and the thresholding process can be applied slice by slice (streamed) on the image, knowing the previously computed exact histogram on the whole image. Figure 3.6 sketches the whole process.

[1]The resolution in a given direction is defined as the number of voxels in this direction per unit of physical length.

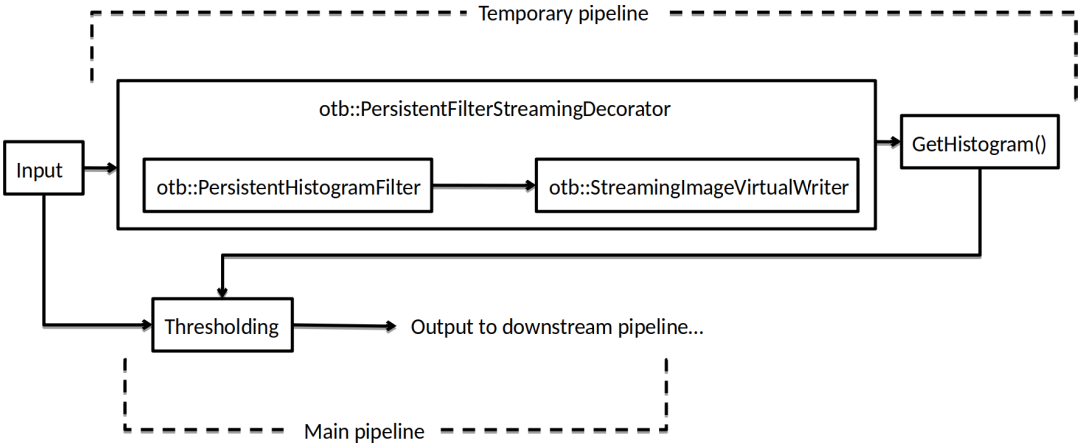


FIGURE 3.6: Temporary and main pipeline for computing a histogram-driven threshold of an image while streaming.

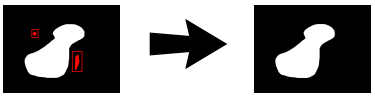
The above procedure is implemented using the *Orfeo Toolbox* (OTB [66]). The Orfeo Toolbox extends the capabilities of ITK, particularly with regards to the processing of huge data set (of the order of the terabyte). The above procedure can be easily set up thanks to the *OTB persistent filters*[2].

More precisely, the *otb::PersistentImageFilter* allows to compute some global feature of an image (as its mean, histogram...) and uses that information for a downstream filter while streaming. The *otb::PersistentImageFilter* behaves as a classical ITK filter. It contains however two additional pure virtual methods *Reset()* and *Synthetize()*. These two methods have to be implemented by the user via a class inheriting from *otb::PersistentImageFilter*. The *Reset()* method sets the intermediate result members of the inheriting class in order to start a fresh processing. The *Synthetize()* method, in turn, processes the final result once the intermediate results have been computed.

Any class derived from the *otb::PersistentImageFilter* is intended to work with the *otb::PersistentFilterStreamingDecorator* and a class named *otb::StreamingImageVirtualWriter*. The *otb::PersistentFilterStreamingDecorator* creates a temporary pipeline implying the derived class of *otb::PersistentImageFilter* and the *otb::StreamingImageVirtualWriter*. The later is then updated in order to simulate a writer filter but discards immediately the passed data without writing anything on the disk. Once the temporary pipeline has been updated, the intermediate results accumulated are gathered thanks to a call of the *Synthetize()* method. Finally, the main pipeline can be updated, using the final result obtained during the updating of the temporary pipeline.

[2]See `https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech29.html` for a more comprehensive description of this feature.

3.6 Streaming of the box filter (step 1bis)



- **Aim**: remove small groups of isolated voxels from a binary image.
- **Input**: binary image.
- **Output**: processed binary image with small groups of voxel removed.
- **Original filter**: see Section 2.3.5.

The *Box filter* removes groups of connected feature pixels that are connected and smaller than a given box. However, taking only a slice of the data may cut such groups into two subsets and made them appear smaller than they really are. As a consequence, some such subgroups may be removed by this filter when they should not. The following section describes how such event can be prevented when using streaming.

During the streaming the *Box filter* takes advantage of the ITK negotiation process between filters. Indeed, in order to accommodate the streaming process, the input requested image region received by the downstream filter is padded by the user's parallelepipedic box size. The *Box filter* then requests, via the *GenerateInputRequestedRegion()* method, the padded image region[3] as input region and returns the unpadded corresponding image region after processing (see Figure 3.8).

This negotiation process guarantees that, during the streaming process, the *Box filter* will output the same results as the normal (i.e. non-streaming) process. Actually, the *Box filter* algorithm will cluster the foreground pixels in a different ways following if the streaming is used or not. For instance, if a set of connected of foreground pixels crosses two streamed parts of the whole image, the set will be divided into two clusters instead of one for the non-streaming process. From here, two cases may arise:

1. The connected set of foreground pixels S fits into the user's parallelepipedic box and is discarded.
2. The connected set of foreground pixels S does not fit into this box and should be kept.

[3] Possibly cropped by the largest possible region.

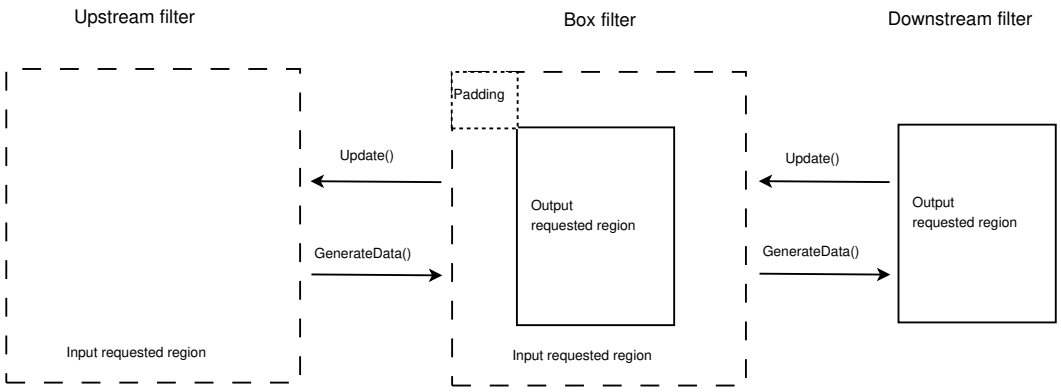
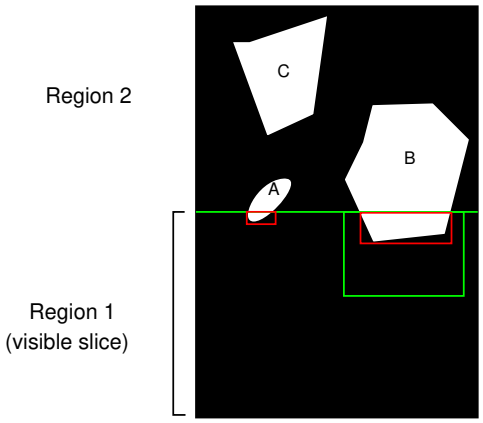


FIGURE 3.8: Image region negotiation process for the box filter.

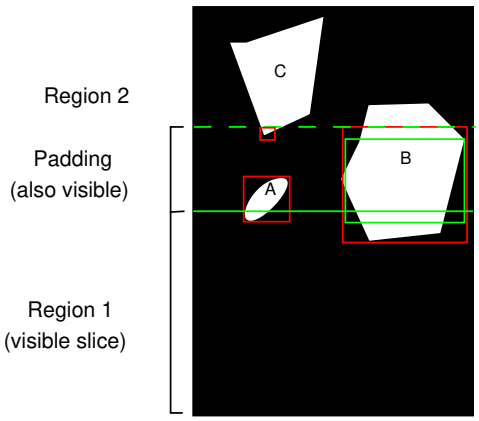
The first case is trivial: during the streaming process, this set S will be divided into two smaller sets S_1 and S_2, with $S_1 \cup S_2 = S$. As the initial bigger set S fits into the user's box, so will do the two smaller sets. The two smaller sets S_1 and S_2 will then be discarded. This is thus equivalent to discard the initial set S.

The second case is more ambiguous. If the initial bigger set S does not fit into the user's box and should be kept, there is no guarantee that the sets S_1 and S_2 will still be bigger than the user's box. As a result, S_1 or S_2, or both may be discarded instead of being kept. It is here that the negotiation process between filters intervenes. As the output requested region by the downstream filter is padded by the user's box size, the *Box filter* can determine if the set S is bigger or not than the user's box. Indeed, if S is entirely contained in the padded image region, this means that $S_1 = S$ and $S_2 = \emptyset$, and there is no more ambiguity. If S is possibly not entirely contained in the padded image region (foreground pixels of S are located on the boundary of the padded region), this means that S extends inside the output requested region and, at least, through all its padding. As the padding is of the size of the user's box, this means that the visible part of S in the padded region is bigger than the user's box. As a result, S is correctly kept. Figure 3.9 illustrates how this ambiguity is resolved.

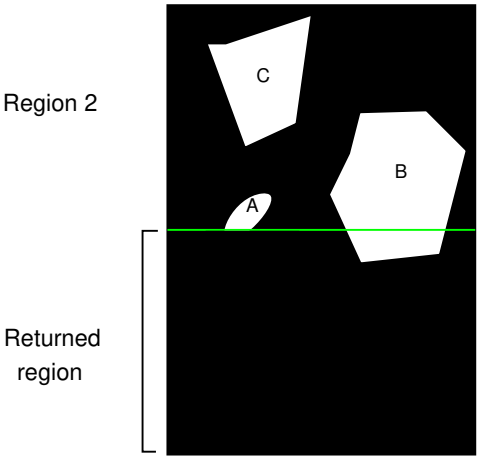
Note that in the padded image region, new sets of foreground connected pixels may be visible for the box filter. Again, theses sets may or may not extend beyond the padded image region and, consequently may or may not fit into the user's box. Therefore, there is a risk that they may wrongly discarded by the *Box filter*. However, this does not impact the output result of the *Box filter*. As a matter of fact, only the output requested region is returned to the downstream filter. Wrongly discarded sets of foreground pixels belonging to the padded region and not to the output requested region are not returned to the downstream filter. The involved pixels will (respectively have been) simply and correctly be processed into the next (respectively previous) region of the image during the streaming process. Figure 3.10 compares the results of the *Box filter* with the streaming and non-streaming case on the same input image.



(a) Without image region padding, the visible part of set B (inside the red box on the right) will be discarded because it fits the parallelippipedic user's box.

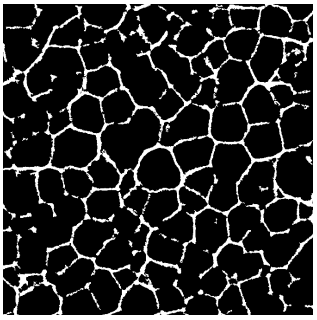


(b) With image region padding, a bigger part of set B is visible. As this set crosses the padding it will not fit inside the user's box and will be kept, as it should. The set A is now entirely visible and will be discarded.

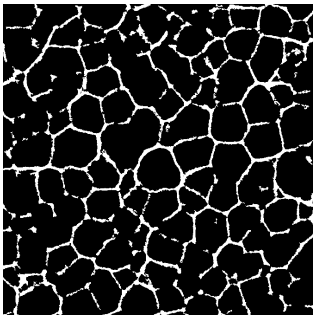


(c) Result: part of set A in returned region is discarded, while set B is kept. Set C is simply ignored and will be processed later on within region 2.

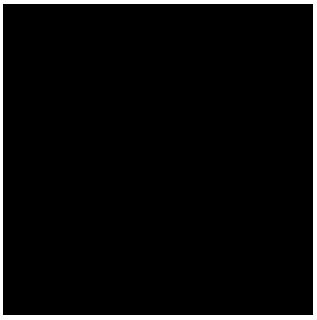
FIGURE 3.9: Scheme showing how the *Box filter* discards or keep sets of connected foreground pixels during the streaming process.



(a) *Box filter* of a whole image.



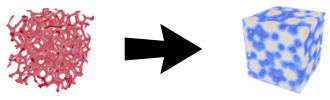
(b) *Box filter* of image by streaming (10 slices).



(c) Difference between the two previous images.

FIGURE 3.10: Results of the box filter for the streaming (10 slices) and non–streaming case for the same input image.

3.7 Streaming of the distance transform (step 2)



- **Aim**: compute the Euclidean distance transform of a binary image, where white (feature) voxels represent cell walls/struts and are assigned distance zero.
- **Input**: binary image with black and white (feature) voxels.
- **Output**: gray–level image[4] with each voxel value equal to the Euclidean distance to the closest feature voxel.
- **Original filter**: see Section 2.3.6

For computing the Euclidean distance transform, Section 2.3.6 uses the algorithm of Maurer et al. [111]. The aim of the distance transform is to find voxel candidates that can be located at the centres of the cells. These candidates are found as local maxima of the distance transform. The algorithm of Maurer et al. relies on a dimensional reduction paradigm and the construction of partial Voronoï diagrams (see Appendix B for details). Here, this dimensional reduction paradigm is exploited in order to obtain a streamed version of the algorithm of Maurer et al.

Thanks to its dimensional reduction paradigm, Maurer et al. algorithm [111] is quite well-suited for streaming. Indeed, as this algorithm performs scans along lines in each direction, the use the slice-shaped subregions during streaming appears as a natural choice. For 3-dimensional images, the streaming filter uses slices perpendicular to the z-direction (see Figure 3.2). This choice is further motivated by the fact that the parallel algorithm implemented in ITK already distributes among the threads contiguous slice-shaped subregions perpendicular to the z-direction [158]. Thus, this streaming choice is the most likely to minimise cache misses during computations.

However, during the streaming process, distance transforms are computed independently on each slice leading to inaccurate results, especially on boundary faces between two slices. Indeed, the Closest Feature Pixel (CFP) of a given pixel in a slice may be located outside that slice. As the distance transform does not have data from outside the current slice, it will take as CFP a Feature Pixel (FP) located inside the slice. This will lead the distance transform to underestimate certain distances. This remark is important as it ensures that the below extended region strategy will work.

[4]Note: here, for visualisation purposes, this gray–level image is coded using a blue-to-red convention instead of a black-to-white one.

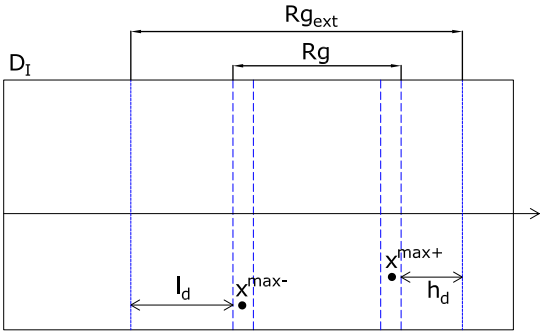


FIGURE 3.12: Computation of the correct distance transform on slice Rg, by computing an incorrect distance transform on slice Rg_{ext}. See text for details.

3.7.1 Extended region strategy

The aim of computing the distance transform on an extended region is to be able to retrieve the exact distance transform of an image (as there was no streaming at all) while performing streaming and, thus, never have to load all the image data in memory at once. This can be achieved by computing inexact distance transforms on extended image regions. As a matter of fact, the distance transforms will be mostly inexact on the faces of the considered slices. On the contrary, they should be almost (if not entirely) exact for pixels located in the "bulk" (inside) of the slice. Naturally, this assumption depends on the distribution of feature pixels inside the considered image. If feature pixels are rare and/or unevenly distributed inside the image, this assumption is more likely to be wrong.

Though, in the context of image analysis of cellular materials, feature pixels can generally be considered as homogeneously distributed at the scale of several typical cell size. This is a common assumption done in the context of computational homogenisation [84]. For slices with thickness of the order of several (three or four) typical cell size, "bulk" pixels should have their corresponding CFP located in the same slice. The idea of the extended region strategy is thus the following: given a input requested region, compute an inexact distance transform on a extended region containing the input requested region. The extended region is computed such that the distance transform over the contained input requested region will be exact.

More precisely, the extended region strategy consists in computing an incorrect distance transform on a thicker slice Rg_{ext} so as to ensure a correct distance transform inside the initial slice Rg. The thicker slice Rg_{ext} is computed as follows (see Figure 3.12): first, an incorrect distance transform is computed on the initial slice Rg. Second, voxels x^{max-} and x^{max+} on the boundaries of Rg with maximum associated (incorrect) distances l_d and h_d are searched for. Third, the slice Rg is extended to Rg_{ext} along both directions by, respectively, l_d and h_d voxels. Finally, an incorrect distance transform is computed on Rg_{ext}, with the guarantee to be correct on slice Rg. The correct distance transform is then returned on slice Rg.

3.7.2 Discussion

1. The extended region strategy is similar to the one applied for computing the histogram needed by the algorithm of Ridler & Calvard and takes some ideas of the extended-region strategy used for the *Box-filter*.

2. If a cell traverses the whole image or if the feature pixels are rare and/or unevenly distributed inside the image, the returned extended region can be the largest possible region (i.e., in general the whole image) or, at least, a large part of it. For that case, the above strategy is pointless as (almost) the whole image has to be loaded into memory. Nevertheless, it is believed that such a case is extreme and seldom arise in the context of image analysis of cellular materials (and especially foams). Actually, having a foam sample with one cell traversing most of the sample is an almost certain sign that the sample is not well suited for any study of any kind[5].

3. The extended region Rg_{ext} is bigger than necessary, as upper bounds are used for computing the extension in each direction. In general, the distance transform is therefore exact over a bigger region than Rg. These bounds may be tighten if the full positions of the CFPs of $\mathbf{x}^{max\pm}$ are taken into account. For the moment, only their distances to their corresponding points are considered here.

3.7.3 Distance transform of a subregion of an image

This section demonstrates that the above heuristic extended region strategy is actually mathematically correct and accurate.

In order to ensure an exact distance transform on the input requested region, a first inexact distance transform is computed on this region. Then, the below propositions and definitions are used.

Proposition 3.1. Given a binary image I of domain D_I and two image regions Rg_1 and Rg_2 such that $Rg_1 \subseteq Rg_2 \subseteq D_I$, and a distance transform $\mathcal{D}$, then:

$$\forall \mathbf{x} \in Rg_1,\ \left[\mathcal{D}_{|Rg_1}(I)\right](\mathbf{x}) \geq \left[\mathcal{D}_{|Rg_2}(I)\right](\mathbf{x})$$

Where $\mathcal{D}_{|Rg_i}$, $i = 1, 2$ denotes the distance transform restricted to regions Rg_i, $i = 1, 2$.

i.e., the pixel values of the distance transform over region Rg_1 are bigger or equal to pixel values of the distance transform over region Rg_2.

Proof. If the CFP $\mathbf{u}$ of a point $\mathbf{x} \in Rg_1$ is located outside Rg_1 (i.e. $\mathbf{u} \in Rg_2 \setminus Rg_1$), the distance transform will look for the closest FP $\mathbf{v} \in Rg_1$ of $\mathbf{x}$. Therefore $d(\mathbf{x}, \mathbf{v}) \geq d(\mathbf{x}, \mathbf{u})$, where d is the distance associated to the distance transform. If it is not the case, this would mean that $\mathbf{u}$ is not the CFP of $\mathbf{x}$, which is a contradiction.

If $\mathbf{u} \in Rg_1$, then $\mathbf{u} = \mathbf{v}$ and $\left[\mathcal{D}_{|Rg_1}(I)\right](\mathbf{x}) = \left[\mathcal{D}_{|Rg_2}(I)\right](\mathbf{x})$. □

[5] Excepted if the aim is precisely to study such samples...

Corollary 3.1. From proposition 3.1 and its hypothesises it follows:

$\forall \mathbf{x} \in Rg_1$, if $\mathbf{u} \in Rg_2$ is the CFP of $\mathbf{x}$, then $d(\mathbf{x}, \mathbf{u}) \leq \left[\mathcal{D}_{|Rg_1}(I)\right](\mathbf{x})$

i.e. $\mathbf{u}$ is located somewhere in a ball of radius $\left[\mathcal{D}_{|Rg_1}(I)\right](\mathbf{x})$ centred on $\mathbf{x}$.

Proof. Let's suppose the above corollary is false:

$\exists \mathbf{x} \in Rg_1$ such that $d(\mathbf{x}, \mathbf{u}) > \left[\mathcal{D}_{|Rg_1}(I)\right](\mathbf{x})$.

Then, by proposition 3.1, $d(\mathbf{x}, \mathbf{u}) > \left[\mathcal{D}_{|Rg_2}(I)\right](\mathbf{x})$.

This means that $\mathbf{u}$ is not the CFP of $\mathbf{x}$ inside the region Rg_2, which is a contradiction. □

Interior boundaries of image

Definition 3.1. Given an image I of domain D_I, and a region $Rg \subseteq D_I$, $\mathbf{x} \in Rg$ is said to belong to *an interior boundary* of Rg if $\exists \mathbf{y} \in N_I(\mathbf{x}) \mid \mathbf{y} \notin Rg$.

Where $N_I(\mathbf{x})$ designates the neighbouring pixels of $\mathbf{x}$. See Figure 3.13b.

Definition 3.2. Given an image I of domain D_I, $\mathbf{x}, \mathbf{y} \in D_I$ are said to be *linked* if $\mathbf{x} \in N_I(\mathbf{y})$ or $\mathbf{y} \in N_I(\mathbf{x})$. See Figure 3.13c.

Definition 3.3. Given an image I of domain D_I, a set of pixels $\{\mathbf{z}_i\}_{1 \leq i \leq N} \subseteq D_I$, is said to be *chained* if $\mathbf{z_i}$ is linked to $\mathbf{z_{i+1}}$, $i = 1, \ldots, N-1$.

Such a set forms a *chain* of linked pixels. See Figure 3.13d.

Definition 3.4. Given an image I of domain D_I and a region $Rg \subseteq D_I$, $\mathbf{x}, \mathbf{y} \in Rg$ are said to be *chained inside* Rg if it exists a chain $C = \{\mathbf{z}_i\}_{1 \leq i \leq N} \subseteq Rg$ such that $\mathbf{x}, \mathbf{y} \in C$. See Figure 3.13d.

Definition 3.5. Given an image I of domain D_I, a region $Rg \subseteq D_I$ and a pixel $\mathbf{x}$ located on an interior boundary of Rg, the *interior boundary* $Fr_{Rg}(\mathbf{x})$ of Rg *relatively* to $\mathbf{x}$ is:

$Fr_{Rg}(\mathbf{x}) = \{\mathbf{y} \in$ interior boundary of $Rg \mid \mathbf{y}$ and $\mathbf{x}$ are chained inside $Rg\}$. See Figure 3.13e.

Definition 3.6. Given an image I of domain D_I, the interior boundaries of a region $Rg \subseteq D_I$ are:

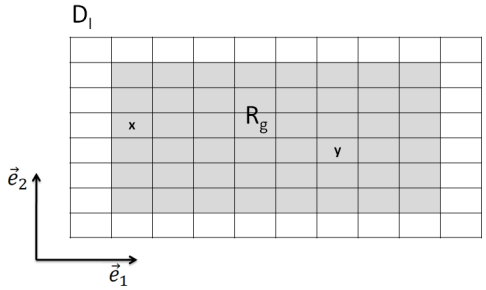
$Fr_{Rg} = \{\mathbf{y} \in Fr_{Rg}(\mathbf{x}) \mid \mathbf{x} \in$ interior boundary of $Rg\}$. See Figure 3.13e.

Definition 3.7. Given an image I of domain D_I, a region $Rg \subseteq D_I$ and $\mathbf{x}$ belonging to an interior boundary of Rg, the interior boundary $Fr_{Rg}(\mathbf{x})$ is said to be *flat* for the direction R_d if:

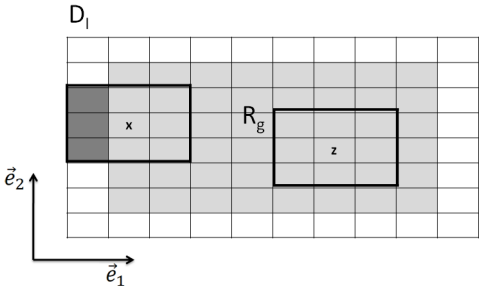
$\forall \mathbf{y}, \mathbf{z} \in Fr_{Rg}(\mathbf{x})$, $y_d = z_d$.

Where the subscript d indicates the d–th component of a vector. See Figures 3.13f, 3.13g and 3.13h.

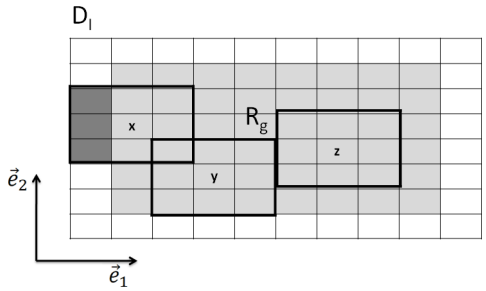
Definition 3.8. Given an image domain D_I of dimension n, a region $Rg \subseteq D_I$ is said to be *parallelepipedic* if it is of the form $Rg = [b_1, t_1] \times \ldots \times [b_n, t_n]$; $b_i, t_i \in \mathcal{R}$, $i = 1, \ldots, n$. See Figure 3.14.



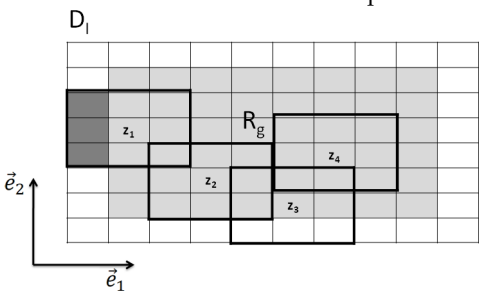
(a) Two pixels **x** and **y** inside a region R_g of an image of domain D_I.



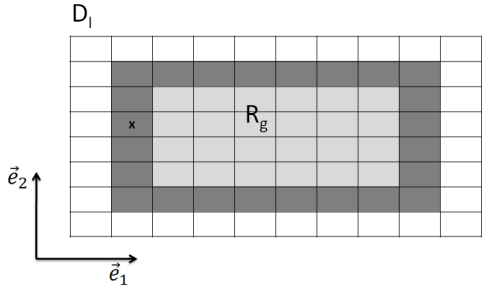
(b) Pixel $\mathbf{x} \in R_g$ belongs to the interior boundary of R_g because at least one of its neighbouring pixels (in dark–gray colour) do not belong to R_g, while it is not the case for pixel **z**.



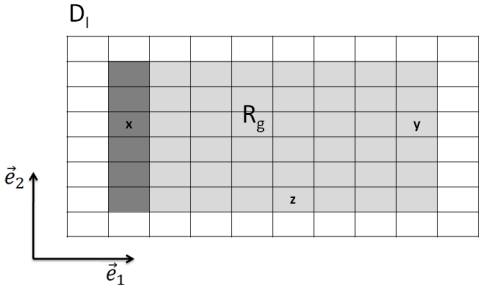
(c) Pixels **x** and **y** are *linked* because their respective neighbouring regions are overlapping, while it is not the case for pixel **z**.



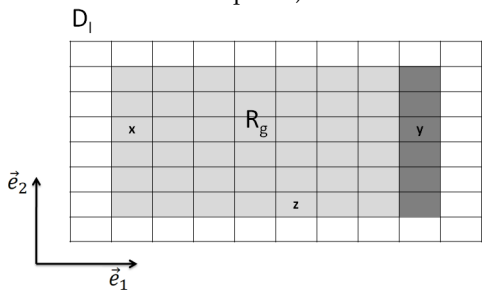
(d) Pixels $\mathbf{z}_1 = \mathbf{x}$, $\mathbf{z}_2$, $\mathbf{z}_3$ and $\mathbf{z}_4 = \mathbf{y}$ are chained inside R_g.



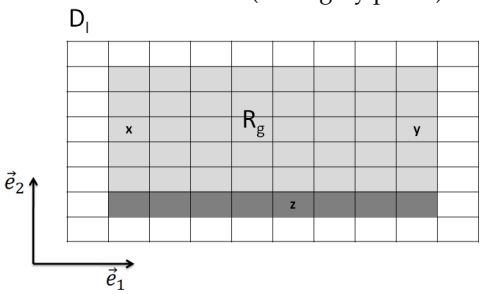
(e) Interior boundary of R_g w.r.t pixel **x** (dark–gray pixels).



(f) Flat interior boundary of R_g w.r.t pixel **x** along direction $d = 1$ (dark–gray pixels).



(g) Flat interior boundary of R_g w.r.t pixel **y** along direction $d = 1$ (dark–gray pixels).



(h) Flat interior boundary of R_g w.r.t pixel **z** along direction $d = 2$ (dark–gray pixels).

FIGURE 3.13: Schematics illustrating definitions 3.1 to 3.8 (see text). D_I: whole image domain (white and light–gray pixels), R_g: considered region inside D_I (light–gray pixels). Vectors $\vec{e}_1$ and $\vec{e}_2$ indicate directions $d = 1$ and $d = 2$ respectively.

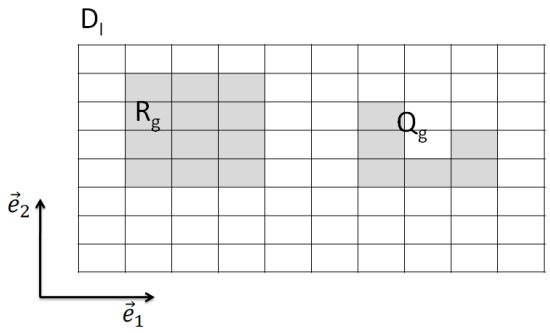


FIGURE 3.14: Schematic illustrating definition 3.8 (see text). D_I: whole image domain (white and light–gray pixels), R_g an Q_g: considered regions inside D_I (light–gray pixels), which are, respectively, parallelipipedic and non–parallelipipedic.

Useful lemmas

Lemma 3.1. (Inspired from property 4 of Maurer et al. [111]). Given an image I of dimension n and domain D_I, a region $Rg \subseteq D_I$, $\mathbf{y} \in Rg$ and $\mathbf{x}$ belonging to a flat interior boundary of Rg for the direction R_d. Moreover, $\mathbf{u}$ is the CFP of $\mathbf{x}$ and $\mathbf{v}$ is the CFP of $\mathbf{y}$ such that:

1. $x_i = y_i,\ \forall i \neq d;\ i = 1, \ldots, n.$
2. $x_d < y_d$ (resp. $x_d > y_d$).
3. $u_i = v_i,\ \forall i \neq d;\ i = 1, \ldots, n.$
4. $u_d, v_d \leq y_d$ (resp. $u_d, v_d \geq y_d$).

Then $u_d \leq v_d$ (resp. $u_d \geq v_d$) (see Figure 3.15).

Proof. The proof is given for the '$<$' case. It is similar for the converse case. If the lemma is false, then $v_d < u_d \leq x_d < y_d$ and if d is the distance associated to the distance transform: $d(\mathbf{y}, \mathbf{u}) < d(\mathbf{y}, \mathbf{v})$.

However, $\mathbf{v}$ is the CFP of $\mathbf{y}$. This implies: $d(\mathbf{y}, \mathbf{v}) \leq d(\mathbf{y}, \mathbf{u})$, which is a contradiction. □

Lemma 3.2. (Inspired from property 4 of Maurer et al. [111]). Given an image I of dimension n and domain D_I, a region $Rg \subseteq D_I$, $\mathbf{y} \in Rg$ and $\mathbf{x}$ belonging to a flat interior boundary of Rg for the direction R_d. Moreover, for the Euclidean distance transform $\mathbf{u}$ is the CFP of $\mathbf{x}$ and $\mathbf{v}$ is the CFP of $\mathbf{y}$ such that:

1. $x_i = y_i,\ \forall i \neq d;\ i = 1, \ldots, d.$
2. $x_d < y_d$ (resp. $x_d > y_d$).
3. $u_d, v_d \leq y_d$ (resp. $u_d, v_d \geq y_d$).

Then $u_d \leq v_d$ (resp. $u_d \geq v_d$) (see Figure 3.16).

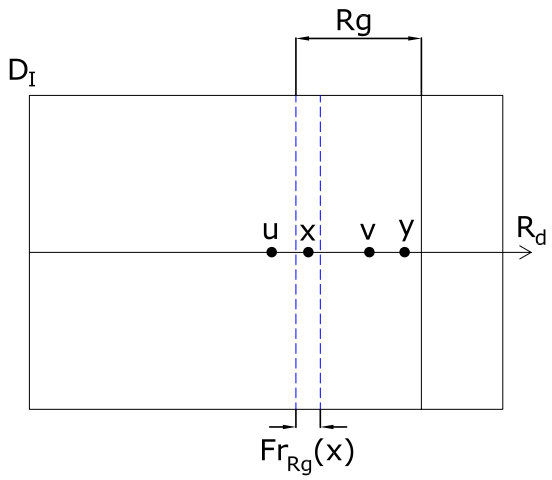


FIGURE 3.15: For an image of domain D_I, a region $Rg \subseteq D_I$ and pixels $\mathbf{x} \in$ interior boundary of Rg ($Fr_{Rg}(\mathbf{x})$) and $\mathbf{y} \in Rg$ located along the direction R_d such that $x_d < y_d$: if their respective CFP (closest feature pixels) $\mathbf{u}$ and $\mathbf{v}$ are also aligned along R_d, then $u_d \leq v_d$.

Proof. The proof is given for the '$<$' case. It is similar for the converse case. In that case, let's denote $\hat{\mathbf{e}}_d = (0,\ldots,0,1,0,\ldots,0)$ the d-th canonical vector of $\mathcal{R}^n$ which consists on an all zeros tuple except on the d-th position which is a one.

Let's $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ be the following projections of respectively $\mathbf{u}$ and $\mathbf{v}$ on the line R_d:

$\tilde{\mathbf{u}} = <\hat{\mathbf{e}}_d|\mathbf{u}>\hat{\mathbf{e}}_d + \mathbf{t} = u_d\hat{\mathbf{e}}_d + \mathbf{t}$

$\tilde{\mathbf{v}} = <\hat{\mathbf{e}}_d|\mathbf{v}>\hat{\mathbf{e}}_d + \mathbf{t} = v_d\hat{\mathbf{e}}_d + \mathbf{t}$

Where: $\mathbf{t} = (t_1,\ldots,t_n)$ with $t_i = \begin{cases} x_i & i \neq d \\ 0 & i = d \end{cases}$

Then, $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ satisfy hypothesis 3 of lemma 3.1 and therefore $u_d \leq v_d$. □

Lemma 3.3. Given an image I of domain D_I, a region $Rg \subseteq D_I$, a pixel $\mathbf{a}$ belonging to a flat interior boundary of Rg for the direction R_d and the distance transform $\mathcal{D}$.

$\forall \mathbf{x} \in Fr_{Rg}(\mathbf{a})$ such that $\mathbf{u}$ is CFP of $\mathbf{x}$ and $u_d \leq x_d$ (resp. $u_d \geq x_d$):

$$u_d \geq x_d - [\mathcal{D}(I)](\mathbf{x}^{max}) \text{ (resp. } u_d \leq x_d + [\mathcal{D}(I)](\mathbf{x}^{max})).$$

Where $\mathbf{x}^{max} = arg\,max_{(\mathbf{z}\in Fr_{Rg}(\mathbf{a}))}\{[D(I)](\mathbf{z})\}$ and $[\mathcal{D}(I)](\mathbf{z}) = min_{\mathbf{y}\in D_I}\{d(\mathbf{z},\mathbf{y}) \mid I(\mathbf{y}) = 0\}$ is the distance transform of pixel $\mathbf{z}$ (see Appendix B).

Proof. Given an $\mathbf{x} \in Fr_{Rg}(\mathbf{a})$ of CFP $\mathbf{u}$ such that $u_d < x_d$ or $u_d > x_d$. Then, for the distance d associated to the distance transform $\mathcal{D}$ and the dth canonical vector $\hat{\mathbf{e}}_d$ of $\mathcal{R}^n$:

$$\begin{aligned} |u_d - x_d| &= d(u_d\hat{\mathbf{e}}_d, x_d\hat{\mathbf{e}}_d) \\ &\leq d(u_d\hat{\mathbf{e}}_d + \tilde{\mathbf{u}}, x_d\hat{\mathbf{e}}_d + \tilde{\mathbf{x}}) \text{ (Triangular inequality and symmetry)} \\ &= d(\mathbf{u},\mathbf{x}) \\ &= [\mathcal{D}(I)](\mathbf{x}) \\ &\leq [\mathcal{D}(I)](\mathbf{x}^{max}) \end{aligned}$$

Where: $\tilde{\mathbf{u}} = \mathbf{u} - u_d\hat{\mathbf{e}}_d$ and $\tilde{\mathbf{x}} = \mathbf{x} - x_d\hat{\mathbf{e}}_d$.

If $u_d \leq x_d$: $-u_d + x_d \leq [\mathcal{D}(I)](\mathbf{x}^{max}) \rightarrow u_d \geq x_d - [\mathcal{D}(I)](\mathbf{x}^{max})$.

If $u_d \geq x_d$: $u_d - x_d \leq [\mathcal{D}(I)](\mathbf{x}^{max}) \rightarrow u_d \leq x_d + [\mathcal{D}(I)](\mathbf{x}^{max})$. □

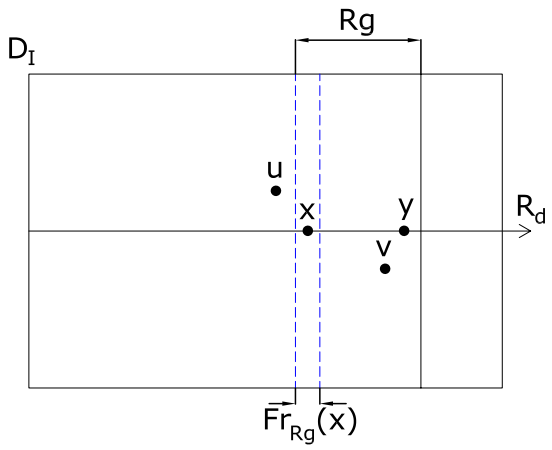


FIGURE 3.16: For an image of domain D_I, a region $Rg \subseteq D_I$ and pixels $\mathbf{x} \in$ interior boundary of Rg ($Fr_{Rg}(\mathbf{x})$) and $\mathbf{y} \in Rg$ located along the direction R_d such that $x_d < y_d$: if their respective CFP (closest feature pixels) are $\mathbf{u}$ and $\mathbf{v}$, then $u_d \leq v_d$.

Lemma 3.4. Given:

1. An image I of dimension n and domain D_I.
2. A parallelepipedic region $Rg \subseteq D_I$.
3. The Euclidean distance transform $\mathcal{ED}$.
4. A pixel $\mathbf{a}$ belonging to a flat interior region of Rg for the direction Rd.
5. A pixel $\mathbf{y} \in Rg$ such that $a_d \leq y_d$ (resp. $a_d \geq y_d$).
6. $\mathbf{v}$ the CFP of $\mathbf{y}$ such that $v_d \leq y_d$ (resp. $v_d \geq y_d$).

Then: $v_d \geq x_d^{max} - [\mathcal{ED}(I)](\mathbf{x}^{max})$ (resp. $v_d \leq x_d^{max} + [\mathcal{ED}(I)](\mathbf{x}^{max})$).

Where: $\mathbf{x}^{max} = arg\ max_{(\mathbf{z} \in Fr_{Rg}(\mathbf{a}))} \{[\mathcal{ED}(I)](\mathbf{z})\}$.

Proof. The proof is given for the '$\leq$' case. It is similar for the converse case.

Let's $\mathbf{x} \in Fr_{Rg}(\mathbf{a})$ be such that $x_i = y_i$, $\forall i \neq d$; $i = 1, \ldots, n$. Such a $\mathbf{x}$ exists because $Fr_{Rg}(\mathbf{a})$ is flat and Rg is parallelepipedic.

Let's $\mathbf{u}$ be the CFP of $\mathbf{x}$ such that $u_d \leq x_d$.
By lemma 3.3 $u_d \geq x_d - [\mathcal{ED}(I)](\mathbf{x}^{max})$.

As $Fr_{Rg}(\mathbf{a})$ is flat for the direction R_d and, $\mathbf{x}, \mathbf{x}^{max} \in Fr_{Rg}(\mathbf{a})$, it follows that $x_d = x_d^{max}$ and $u_d \geq x_d^{max} - [\mathcal{ED}(I)](\mathbf{x}^{max})$

If $y_d = x_d$, then $v_d = u_d$ and the proof is finished.
If $x_d < y_d$, then $u_d \leq x_d < y_d$ and the conditions of lemma 3.2 are satisfied. Hence: $v_d \geq u_d \geq x_d^{max} - [\mathcal{ED}(I)](\mathbf{x}^{max})$. □

Propositions

Proposition 3.2. Given:

1. An image I of dimension n and domain D_I.
2. A parallelepipedic region $Rg \subseteq D_I$.
3. The Euclidean distance transform $\mathcal{ED}$.
4. A pixel $\mathbf{a}$ belonging to a flat interior region of Rg for the direction Rd.
5. A pixel $\mathbf{y} \in Rg$ such that $a_d \leq y_d$ (resp. $a_d \geq y_d$).
6. $\mathbf{v}$ the CFP of $\mathbf{y}$ such that $v_d \leq y_d$ (resp. $v_d \geq y_d$).

Then: $v_d \geq a_d - \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max}_{|Rg})$ (resp. $v_d \leq a_d + \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max}_{|Rg})$).

Where: $\mathbf{x}^{max}_{|Rg} = arg\ max_{(\mathbf{z} \in Fr_{Rg}(\mathbf{a}))} \left\{ \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{z}) \right\}$.

This proposition means that any pixel in a parallelepipedic region Rg has its CFP located below a given distance in a given direction from the boundaries of Rg. Moreover, this distance can be computed only by knowing the distance transform on Rg. This proposition is at the core of the extended region strategy.

Proof. On the one hand, lemma 3.4 ensures that exists $\mathbf{x}^{max}$ such that $v_d \geq x^{max}_d - [\mathcal{ED}(I)](\mathbf{x}^{max})$ (resp. $v_d \leq x^{max}_d + [\mathcal{ED}(I)](\mathbf{x}^{max})$).

On the other hand, with $Rg_1 = Rg$ and $Rg_2 = D_I$, proposition 3.1 states that: $\forall \mathbf{x} \in Rg$, $\left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}) \geq [\mathcal{ED}(I)](\mathbf{x})$.

Moreover, as $Fr_{Rg}(\mathbf{a})$ is flat for direction R_d: $a_d = x^{max}_d$.

Hence: $v_d \geq a_d - \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max}_{|Rg})$ (resp. $v_d \leq a_d + \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max}_{|Rg})$). □

Definition 3.9. Given an image I of dimension n and domain D_I, the Euclidean distance transform $\mathcal{ED}$, and a parallelepipedic region $Rg = [b_1, t_1] \times \ldots \times [b_n, t_n] \subseteq D_I$, the extended region Rg^d_{ext} in a given direction d, $(1 \leq d \leq n)$ is:

$Rg^d_{ext} = [b_1, t_1] \times \ldots \times [b_{d-1}, t_{d-1}] \times [b_d - l_d, t_d + h_d] \times [b_{d+1}, t_{d+1}] \times \ldots \times [b_n, t_n]$.

Where: $l_d = \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max-})$ and $h_d = \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{x}^{max+})$.

With: $\mathbf{x}^{max\pm} = arg\ max_{(\mathbf{z} \in Fr_{Rg}(\mathbf{a}^{\pm}))} \left\{ \left[\mathcal{ED}_{|Rg}(I)\right](\mathbf{z}) \right\}$ and $\mathbf{a}^+, \mathbf{a}^-$ two tuples of $\mathcal{R}^n$ such that, respectively, $a^-_d = b_d$ and $a^+_d = t_d$.

It should be noted that it is not necessary to compute $\mathbf{x}^{max\pm}$ itself, only the Euclidean distance transform is needed.

Definition 3.10. Given an image I of dimension n and domain D_I, the Euclidean distance transform $\mathcal{ED}$, and a parallelepipedic region $Rg = [b_1, t_1] \times \ldots \times [b_n, t_n] \subseteq D_I$, the extended region Rg_{ext} is the union of the extended regions in all directions cropped by D_I: $Rg_{ext} = \left(\cup^n_{d=1} Rg^d_{ext}\right) \cap D_I$.

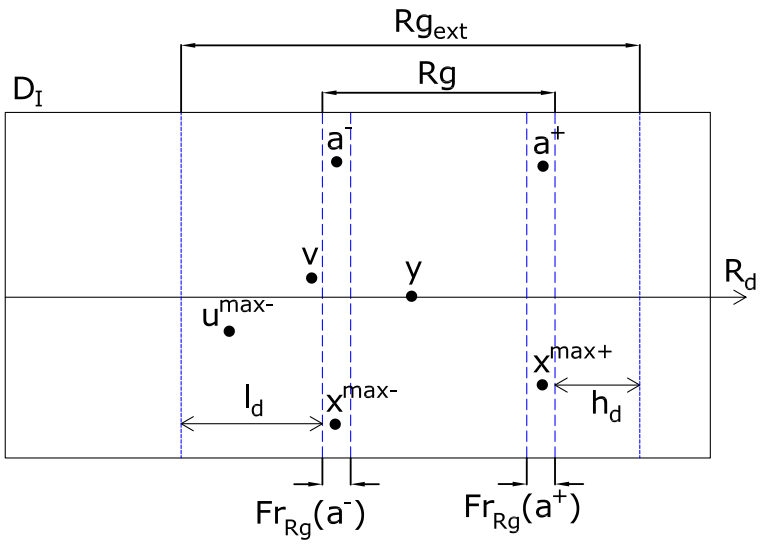


FIGURE 3.17: For an image of domain D_I, a region $Rg \subseteq D_I$ an extended region Rg_{ext} is computed such that $\forall \mathbf{y} \in Rg$ with CFP (closest feature point) $\mathbf{v}$, then $\mathbf{v} \in Rg_{ext}$. The extended region Rg_{ext} is determined by computing the maxima l_d and h_d of the inexact Euclidean distance transform over Rg on the boundaries $Fr_{Rg}(\mathbf{a}^{\pm})$. Computing the inexact Euclidean distance transform over Rg_{ext} then ensures that this transform is exact on Rg. $\mathbf{y} \in Rg$ is an arbitrary pixel and $\mathbf{v}$ is its feature pixel. $\mathbf{x}^{max\pm}$ are the boundary pixels where the maxima are located, and $\mathbf{u}^{max-}$ is the CFP of $\mathbf{x}^{max-}$. R_d is the direction of the extension (only one direction for the sketched case).

Corollary 3.2. Given an image I of domain D_I and a parallelepipedic region $Rg \subseteq D_I$: $\forall \mathbf{y} \in Rg$, if $\mathbf{v}$ is the CFP of $\mathbf{y}$ for the Euclidean distance transform, then $\mathbf{v} \in Rg_{ext}$.

This corollary means that the inexact Euclidean distance transform computed over the extended region Rg_{ext} is guaranteed to be exact over the initial region Rg (see Figure 3.17).

Proof. This follows immediately from proposition 3.2. □

The extended region strategy is thus the following: given a parallelepipedic subregion Rg of an image I of domain D_I, the maxima of the Euclidean distance transform are computed over each interior boundary of Rg using only the pixels of Rg (the rest of the image is not loaded into the memory at this stage). Then, a new extended region Rg_{ext} is computed accordingly to definition 3.10 and a new Euclidean distance transform is computed over Rg_{ext}. Corollary 3.2 ensures that the Euclidean distance transform is then exact over Rg.

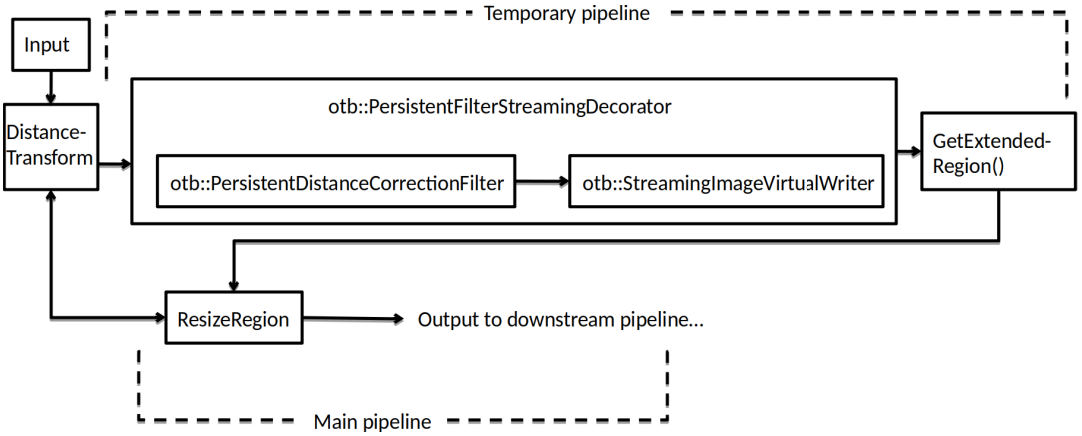
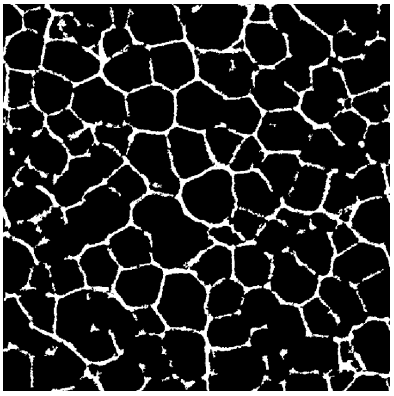


FIGURE 3.18: Temporary and main pipeline for computing the exact distance transform of an image while streaming.

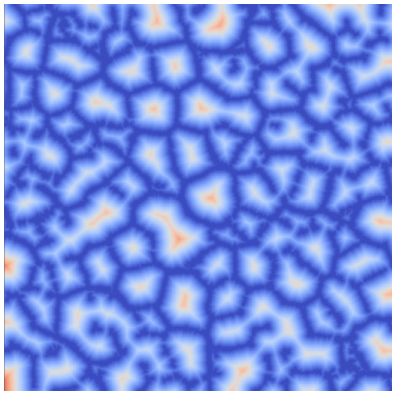
3.7.4 Implementation of the strategy

In order to compute the needed extended regions, a temporary pipeline is set thanks to the *otb::PersistentFilterStreamingDecorator*[6] and the class *otb::PersistentDistanceCorrectionFilter* deriving from the *otb::PersistentImageFilter*. When the temporary pipeline is updated, this derived class updates the *distance filter* on each slice and computes the needed extended regions. It then passes the extended regions to the *ResizeRegion filter*. The *ResizeRegion filter* updates the main pipeline by calling-back the *distance filter* (see Figure 3.18). The *ResizeRegion filter* does not process the data. All it does, is to query upstream the extended regions computed by the temporary pipeline during the negotiation process, and to return downstream the required output regions. By this mean, the *ResizeRegion filter* receive from the *distance filter* an inaccurate distance transform on an extended region, and returns downstream the exact distance transform on the initial required region. Figure 3.19 compares what happens with the euclidean distance transform of a 2-dimensional image while streaming with ten slices without and with the extended region strategy. Table 3.1 shows by how much each input requested region Rg has been extended on the left and right-hand sides and the sizes of the extended regions Rg_{ext}. It can be seen that each extended region Rg_{ext} is on average ≈ 4.72 times thinner than the original (670×670) image.

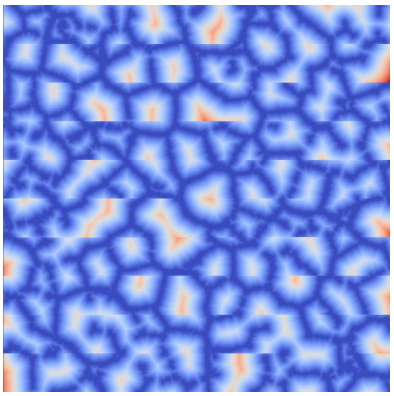
[6]See the website https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech29.html for a more comprehensive description of this feature.



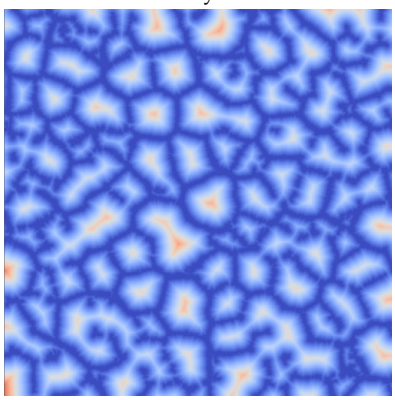
(a) Original binary image before the euclidean distance transform.



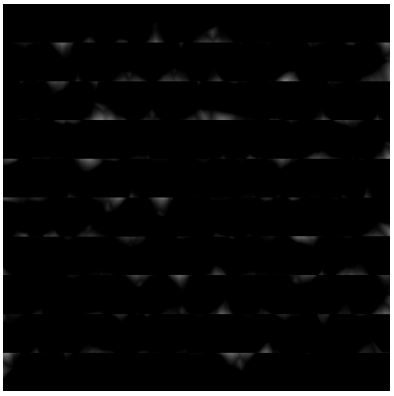
(b) Direct euclidean distance transform of the whole original image. Arbitrary scale.



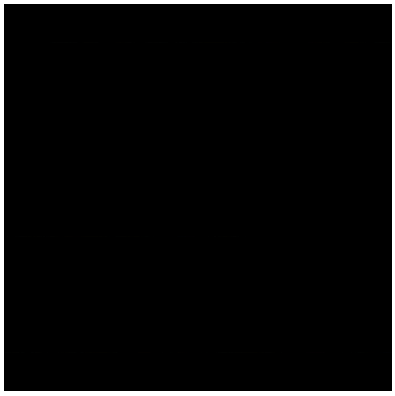
(c) Euclidean distance transform of the original image by naïve streaming (10 slices).



(d) Euclidean distance transform of the original image by streaming (10 slices) using the extended region strategy.



(e) Difference between the original Euclidean distance transform 3.19b and its naïve streamed version 3.19c. Differences appear especially at the boundaries of each slice.



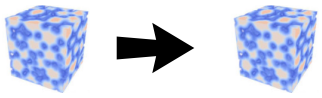
(f) Difference between the original euclidean distance transform 3.19b and streamed version with extended region strategy 3.19d. No difference appears.

FIGURE 3.19: Example of Euclidean distance transforms on a (670×670) 2–dimensional image without and with the extended region strategy.

TABLE 3.1: Left and right extensions and final sizes of extended regions Rg_{ext} needed for computing the exact Euclidean distance transform over ten (670×67) slices of the (670×670) 2-dimensional image shown in figure 3.19a.

Slice	Left extension	Right extension	Rg_{ext}
1	0	42	670×109
2	42	40	670×149
3	40	39	670×146
4	47	45	670×159
5	52	41	670×160
6	39	38	670×144
7	38	30	670×135
8	49	32	670×148
9	59	36	670×162
10	39	0	670×106
Average	**40.5**	**34.3**	**670×141.8**

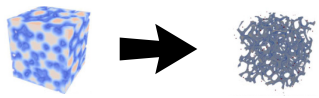
3.8 Streaming of distance post-processing (step 3)



- **Aim**: smooth a distance transform or perform other post-processing tasks.
- **Input**: gray–level image representing a distance transform.
- **Output**: gray–level image representing the corresponding smoothed distance transform.
- **Original filter**: see Section 2.3.7

If some additional filters are added in **step 3**, such as a smoothing step for the distance transform, the user must ensure that they are streamable in order to benefit from this feature.

3.9 Streaming of local maxima (step 4)



- **Aim**: given a gray–level image representing the distance transform of a binary foam image, identify local maxima that will be used as seeds for growing ellipsoids.
- **Input**: gray–level image.
- **Output**: gray–level image marked with local maxima.
- **Original filter**: see Section 2.3.8.

The procedure for finding local maxima of the distance transform described in Section 2.3.8 uses the algorithm of Pham [130]. This algorithm first find maxima candidates by only considering slope changes along scanned lines (see Figure 2.22). It then select the genuine local maxima among the candidates by looking at the neighbours around each candidate and computing local Hessians on each one.

The computation of local maxima from the distance transform in section 2.3.8 requires modifications to the original algorithm of Pham [130]: it needs to "scan" along voxel lines for local maxima candidates. The algorithm has been implemented such that these voxel lines are always located inside a given slice.

More precisely, given a requested region $R_g \subset D_I$ of a grayscale image I of domain D_I, the modified algorithm of T.Q. Pham (7) requires a neighbourhood around the region R_g. This request is handled using the image region negotiation process between filters, in a similar fashion that has been done for the *Box filter*. Moreover, in order to find all local peaks along a given 1D-restriction line of the considered image in the modified algorithm of T.Q. Pham, the 1D-restriction lines $\tilde{I}_d$ are oriented along the x-axis, while the streaming is performed along slice-shaped regions oriented perpendicularly to the z-axis (see Figure 3.2).

Computation of local maxima candidates by this algorithm is thus exact. Local maxima are then selected from candidates by analysing the values associated to their neighbouring voxels in a region of radius R. In order to ensure a correct computation of the local maxima inside the current slice Rg, the same procedure as described earlier for step 2 is used: the current slice Rg is extended to a thicker slice Rg_{ext} where the (incorrect) maxima are looked for, and the (guaranteed correct) maxima inside slice Rg are returned.

The computation of the thicker slice Rg_{ext} is, however, simpler than in step 2: extension distances l_d and h_d are simply equal R; where R is computed as:

$$R = max\left(R_{user}, R_{Hessian}\right) \tag{3.1}$$

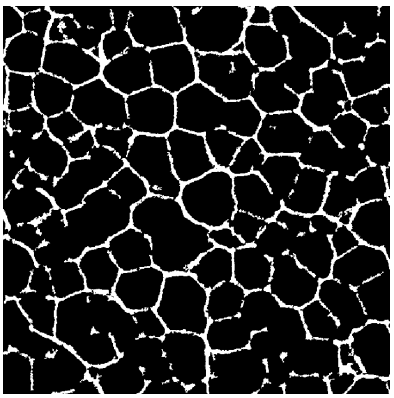
With:

- R_{user} is the radius of the neighbourhood requested by the user for testing each maximum candidate against its neighbouring pixels.
- $R_{Hessian}$ is the radius of the neighbourhood requested for computing an accurate Hessian around each maximum candidate (see Section 2.3.8 on page 52 and following).

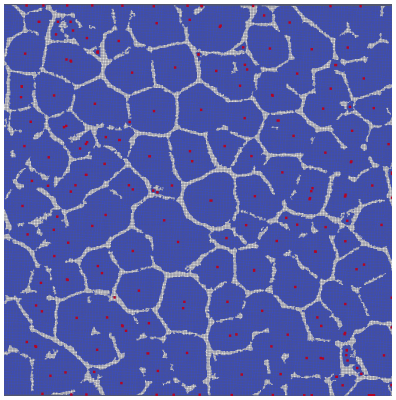
3.9.1 Note on boundaries

In order to detect local maxima at the boundaries ∂D_I of domain D_I, pixels $\mathbf{y}$ belonging to the neighbourhood of $N_I(\mathbf{x})$ and located outside the domain D_I ($\mathbf{y} \notin D_I$) are set to their minimal possible value (i.e. for a grayscale image I this minimal possible value is 0).

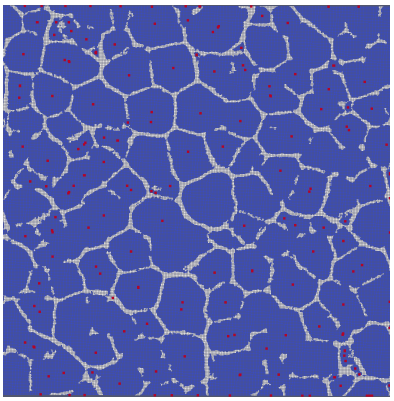
Figure 3.22 shows a comparison example of maxima found on a 670×670 2-dimensional image when streaming it into ten slices using the extended region strategy and without the extended region strategy.



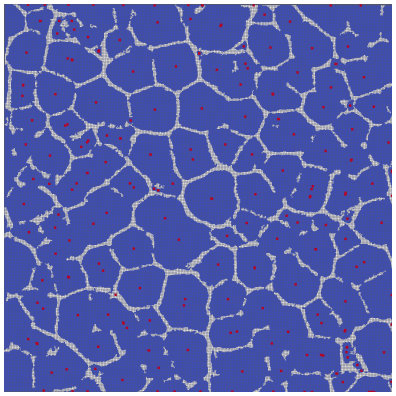
(a) Original binary image before the non-maxima suppression.



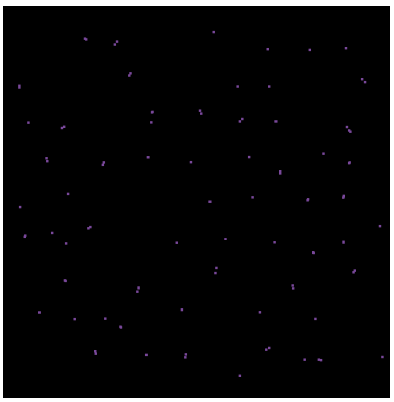
(b) Non-maxima suppression computed from the distance transform 3.19b of image 3.22a.



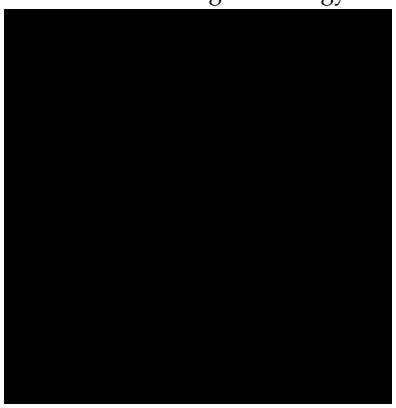
(c) Non-maxima suppression of the distance transform of the original image by naïve streaming (10 slices).



(d) Non-maxima suppression of the distance transform of the original image by streaming (10 slices) using the extended region strategy.



(e) Difference between the original non-maxima suppression 3.22b and its naïve streamed version 3.22c. Differences appear.



(f) Difference between the original non-maxima suppression 3.22b and its streamed version 3.22d with extended region strategy. No difference appears.

FIGURE 3.22: Example of non–maxima suppression on a (670×670) 2–dimensional image with $\sigma = 1.5$, $eigenTol = 10^{-5}$, and $tol = 10^{-5}$ and $R_{user} = 5$ (see algorithm 7 and/or table 2.5). Maxima are artificially highlighted as red squares for visualization purposes. Cell walls appear in light gray.

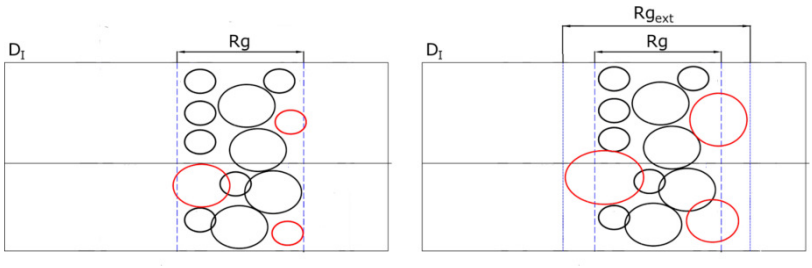
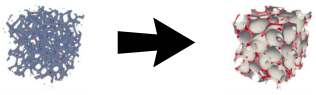


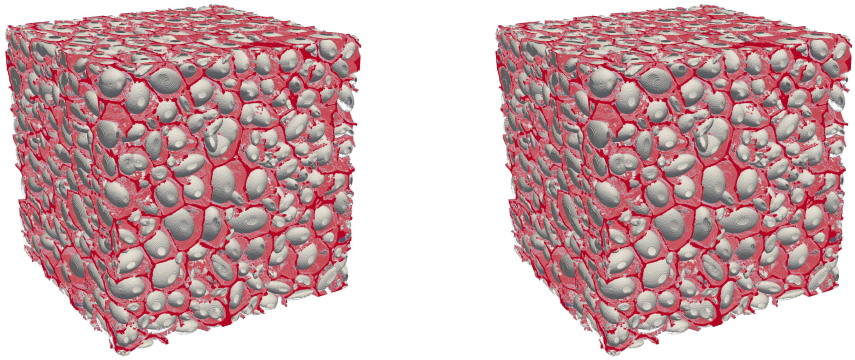
FIGURE 3.24: Growth of ellipsoids inside a slice R_g. If some ellipsoids grew outside Rg (red ellipsoids), their growths are recomputed in a thicker slice Rg_{ext} until no more ellipsoids grow outside the last considered slice.

3.10 Streaming of ellipsoid growth and merge (steps 5 and 6)



- **Aim**: from found local maxima, fit ellipsoids to cells. Cluster and merge overlapping ellipsoids so to obtain one fitted ellipsoid per cell.
- **Input**: local maxima and threshold image.
- **Output**: parent ellipsoids, each fitted to one particular cell.
- **Original filter**: see Section 2.3.9.

Regarding the growth of parent and auxiliary ellipsoids (Sections 2.3.9 and 2.3.11), a similar strategy as in Sections 3.7 and 3.9 is used. Ellipsoids with centres located inside the current slice Rg are grown. If some of the ellipsoids grew outside of the current slice Rg, their growths are recomputed within a thicker slice Rg_{ext} (see Figure 3.24). If the slice Rg_{ext} is not thick enough (meaning that there are still some ellipsoids which grew outside this slice) then, the slice Rg_{ext} is extended further until no more ellipsoid grows outside it or if a boundary of the CT-scan images is reached (meaning that there are no more voxel data available beyond the extended region Rg_{ext}). Figure 3.25 compare qualitatively parent ellipsoids obtained on a 3D-image taken in a whole, and parent ellipsoids obtained on the same 3D-image streamed using 10 slices. Algorithm 35 describes how to expand correctly ellipsoids while using streaming.



(a) (b)

FIGURE 3.25: Ellipsoids generated from the 3D-image in (a) "normal" and (b) streaming (10 slices) modes. No differences can be seen. 3D foam image provided by E. Plougonven (department of Applied Chemistry, University of Liège). Note: no ellipsoid merging were performed.

Algorithm 35 Given a set of ellipsoids $E = \{\mathcal{E}_i\}_{1 \leq i \leq M}$ and a box B_{slice} defining the current considered image slice, expands the ellipsoids as much as possible.

Require: Set of ellipsoids $E = \{\mathcal{E}_1, \ldots, \mathcal{E}_M\}$, box slice B_{slice} of image I, image domain D_I.
Require: Pixel value $p \in Im(B_{slice})$ figuring the value associated to the cell walls (feature pixel).

procedure GROWTHELLIPSOIDSINSIDESLICE(E, B_{slice}, D_I)
  repeat
    $need\_extension \leftarrow false$.
    for $1 \leq i \leq M$ **do**
      $\mathcal{E}_i \leftarrow$ ELLIPSOIDEXPAND(B_{slice}, p, $\mathcal{E}_i$) ▷ Algo. 13
      $B_i \leftarrow$ AXISALIGNEDBOX($\mathcal{E}_i$) ▷ algorithm 27.
      if left side of $B_i <$ left side of B_{slice} **then**
        $need\_extension \leftarrow true$.
        left side of B_{slice} = left side of B_i.
      end if
      if right side of $B_i >$ right side of B_{slice} **then**
        $need\_extension \leftarrow true$.
        right side of B_{slice} = right side of B_i.
      end if
    end for
    if *need_extension* **then**
      $B_{slice} \leftarrow B_{slice} \cap D_I$.
      Load into memory image data inside slice B_{slice}. ▷ This is easily done with ITK by setting the *RequestedImageRegion* to B_{slice}.
    end if
  until !*need_extension*
end procedure

3.10.1 Quantitative analysis

In order to check if the above strategy sketched in figure 3.24 and given in algorithm 35 is effective, a comparison test has been conducted on a $670 \times 670 \times 670$ 3D-image, involving 4250 (non-merged) ellipsoids. Ellipsoids obtained by the "normal" process (i.e. by loading the whole 3D-image in memory) and by streaming (with 10 slices) were compared in Figure 3.25. Furthermore, a quantitative analysis comparing two-by-two the ellipsoids obtained by the "normal" and streaming modes has also been conducted. Definition 3.11 provides the quantitative criterion for measuring dissimilarities between ellipsoids.

Definition 3.11. Let's $\mathcal{E}_1\left(\mathbf{c}_1, M_1\right)$ and $\mathcal{E}_2\left(\mathbf{c}_2, M_2\right)$ be two ellipsoids in an Euclidean affine space Y_n, then their *difference* is given by the application 3.2.

$$\Delta_{\mathrm{Y}_n} : \mathrm{Y}_n \times \mathrm{Y}_n \to \mathcal{R}^+, (\mathcal{E}_1, \mathcal{E}_2) \mapsto \Delta_{\mathrm{Y}_n}\left(\mathcal{E}_1, \mathcal{E}_2\right) = \|\mathbf{c}_1 - \mathbf{c}_2\|_2 + \|M_1 - M_2\|_2 \tag{3.2}$$

The first term in the expression of the *difference* Δ_{Y_n}, which is the simple Euclidean norm of a vector, tells if there is a translation between the two considered ellipsoids, while the second term, which is the 2 matrix-norm[7], measures the shape dissimilarity between them.

Computation of the difference

The *total difference* $diff_{total}$, *mean difference* $diff_{mean}$, and *maximum difference* $diff_{max}$ between the two sets of ellipsoids – obtained from the 3D-image using the "normal" and streamed modes – have been computed as follows:

$$diff_{total} = \sum_{i=1}^{m} \Delta_{\mathrm{Y}_n}\left(\mathcal{E}_{normal,i}, \mathcal{E}_{streaming,i}\right) \tag{3.3}$$

$$diff_{mean} = \left(\frac{1}{m}\sum_{i=1}^{m} \Delta^2_{\mathrm{Y}_n}\left(\mathcal{E}_{normal,i}, \mathcal{E}_{streaming,i}\right)\right)^{1/2} \tag{3.4}$$

$$diff_{max} = max_{i=1,\ldots,m}\, \Delta_{\mathrm{Y}_n}\left(\mathcal{E}_{normal,i}, \mathcal{E}_{streaming,i}\right) \tag{3.5}$$

Where m is the number of ellipsoids, $\mathcal{E}_{normal,i}$ is the ith ellipsoid computed in "normal" mode, and $\mathcal{E}_{streaming,i}$ is the corresponding ellipsoid computed in streaming mode.

The three above differences have been numerically computed on the data shown in Figure 3.25 for $2, 10, 20, \ldots, 80$ slices. For all cases, no difference (up to numerical precision) has been observed with respect to the ellipsoids obtained using the "normal" mode.

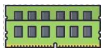
It should be emphasised that the computation of the ellipsoids, both in "normal" and streaming modes, depends on the output of the filters described in the previous chapters. Particularly, the above observation indirectly shows that the different strategies used for handling the streaming for each former filter is indeed effective.

[7] The 2 matrix-norm of a matrix A can be computed as $\sqrt{\lambda_1}$, where λ_1 is the highest eigenvalue of $A^t A$.

3.11 Discussion

With the adaptations described above, it is possible to use the proposed image analysis procedure within a streaming framework, provided that each optional added filter (step 3) is also streamable. As shown in Section 3.12 this allows processing CT-scan data of foams that would normally not fit in the available RAM. Nonetheless, there is one downside and one limitation when using streaming. The downside is that the computational time for processing the whole considered image will increase compared to a normal image analysis procedure (where the whole CT-scan data is fitted into the RAM). This is caused by the fact that some algorithms used for the image analysis procedure have to be called twice. That fact can be seen as "communication costs" between slices. The scalability will therefore not be what one would expect from a parallelised setting. Regarding the limitation, there is no strict guarantee that the extended regions Rg_{ext} will not cover the entire CT-scan data set (and thus obliterating the whole point of doing streaming). As a matter of fact, the biggest extended region Rg_{ext} will approximatively be of the size of the biggest cell present in the foam. In general, cells seldom span an entire foam sample, especially for large samples. Thus, the memory benefits obtained by streaming can be significant.

3.12 Memory usage



Memory usage can be considered as an issue regarding the techniques of geometric reconstructions of foams. Indeed, storing in the RAM the CT-scan data alone may already be an impossible task for standard computers (such as consumer market computers), even less regarding the processing of those data. The above proposed image analysis procedure aims to overcome this limitation by trying to use less RAM than a standard image analysis procedure (and, especially, the watershed). Moreover, its streaming ability offers the possibility for standard computers to process CT-scan data that would not even fit their RAM.

In order to determine the ability of the proposed image analysis procedure, an investigation has been conducted on three types of data sets. Namely:

- Six sets of artificially generated honeycomb foams of different sizes, ranging from a cube of $150 \times 150 \times 150$ voxels to a cube of $1800 \times 1800 \times 1800$ voxels (see left of Figure 3.27).
- Two sets of artificially generated random foams using the DN-CT-SCAN model [79]. One set being a cube of $200 \times 200 \times 200$ voxels and the other a cube of $400 \times 400 \times 400$ voxels (see right of Figure 3.27).
- One set of a real–world foam presented in Section 3.13.1, and of size $600 \times 600 \times 600$ voxels.

Table 3.2 reports the sizes of the different generated artificial honeycomb foams along with their corresponding number of cells. The cell sizes are kept constant for all honeycomb foams. Table 3.2 also reports the theoretical minimum amount of RAM needed for storing the corresponding data, assuming 8 bytes per voxel. The last three columns display the measured peak memory usages[8] for the watershed algorithm (as implemented in the ITK library [69]) and the proposed image analysis procedure without and with using the streaming ability (using 10 slices). The "> 64.0" entry in the table corresponds to a peak memory usage exceeding the maximum of 32 GB RAM available (extended to 64 GB RAM thanks to the RAM compression utility *ZRAM*[9]) of the computer on which this investigation was conducted. It was, therefore, not possible to measure the peak memory usage for this entry of the table.

From Table 3.2 it can be seen that the proposed image analysis procedure is indeed less memory demanding than the watershed algorithm (as implemented in the ITK library [69]) for the considered artificial honeycomb data sets. Especially, the usage of streaming allowed reducing the peak memory usage by a large amount. Thanks to streaming, it was possible to process an artificially generated foam of $1800 \times 1800 \times 1800 \approx 5.8\,10^9$ voxels; which, due to RAM limitations, was possible neither using the watershed nor using the proposed image analysis procedure without using the RAM compression utility *ZRAM*.

[8] Peak memory usages were measured using the software *valgrind* [118].
[9] https://kernelnewbies.org/Linux_3.14

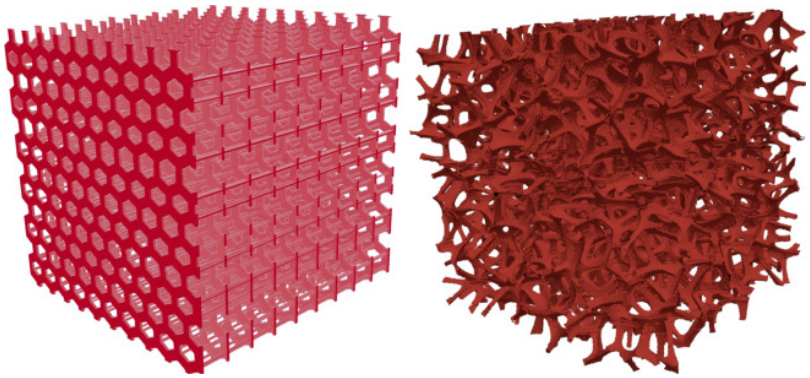


FIGURE 3.27: Left: artificially generated honeycomb foam of size $600 \times 600 \times 600$ voxels and containing 1430 cells. Right: artificial foam generated using the DN-CT-SCAN model [79] with 616 random inclusions.

It should be noted that, technically, only one bit per voxel is required for storing a thresholded image such as the ones shown in figure 3.27. However, some filters, such as the (inverse) euclidean distance transform, require at least 4 bytes per voxel in order to perform their calculations. Distance calculations indeed require a floating type, or at least a sufficiently "big" integer type (such as "int" encoded on 32 bits) if one whishes to compute squared distances. Thus, the real storage cost of voxel data is not 1 bit, but at least 4 bytes per voxel. Here, Table 3.2 considers a worst case scenario were a filter might require 8 bytes per voxel.

Figure 3.28 displays the peak memory usages for the watershed and the proposed image analysis procedure, with and without streaming (points). It can be noted that, for the generated artificial honeycomb foam, the proposed image analysis has always a lower peak memory usage than the watershed, and that the streaming increases this trend. Roughly, by using ten slices, the streaming was able to use ten times less memory than the watershed. This ability can be an advantage when it comes to process large data sets with limited capabilities, or can authorise the processing of bigger data sets which were not processable earlier.

Diamonds in Figure 3.28 show the peak memory usages measured for the real–world foam data set that will be presented in Section 3.13.1. Circles show, for their part, the peak memory usages on artificial foams generated using the DN-CT-SCAN model [79], described in Section 4.1, with random inclusions following a statistical distribution similar to that of the cells of the real-world foam (right of Figure 3.27). It can be seen that the trends are the same as for the artificially generated foam, indicating that the streaming strategy is likely also effective on real-world foam data.

TABLE 3.2: Sizes in pixels, number of cells and minimum RAM (assuming 8 bytes per voxel) needed for storing data for artificially generated honeycomb 3D foams images. Note: Last three columns: measured peak memory usages for watershed and the proposed image analysis procedure with and without streaming. Bold values identify minimum peak memory usages.

Cube side (voxels.)	**Nb. cells**	**Min. RAM (GB)**	**Watershed (GB)**	**Proposed (GB)**	**Streaming (GB)**
150	120	0.03	0.1	**0.05**	0.07
300	390	0.2	0.7	0.27	**0.17**
600	1430	1.6	5.7	2.2	**0.8**
800	3094	3.8	13.4	5.3	**2.0**
1200	10,500	12.9	45.1	16.4	**4.3**
1800	34,410	43.5	> 64.0	59.9	**14.9**

It should be noted, though, that the earlier mentioned limitation of the streaming capability of the proposed image analysis with respect to the maximum cell size drives the peak memory usage. For instance, if a cell extends across the entire foam sample, the proposed streaming capability will be of no use for reducing the peak memory usage. However, for the purpose of reconstructing foam geometries, the cases of cells extending across the whole foam sample can be considered quite exceptional. Especially, this case should arise less frequently with larger foam samples, for which the usage of streaming can be useful.

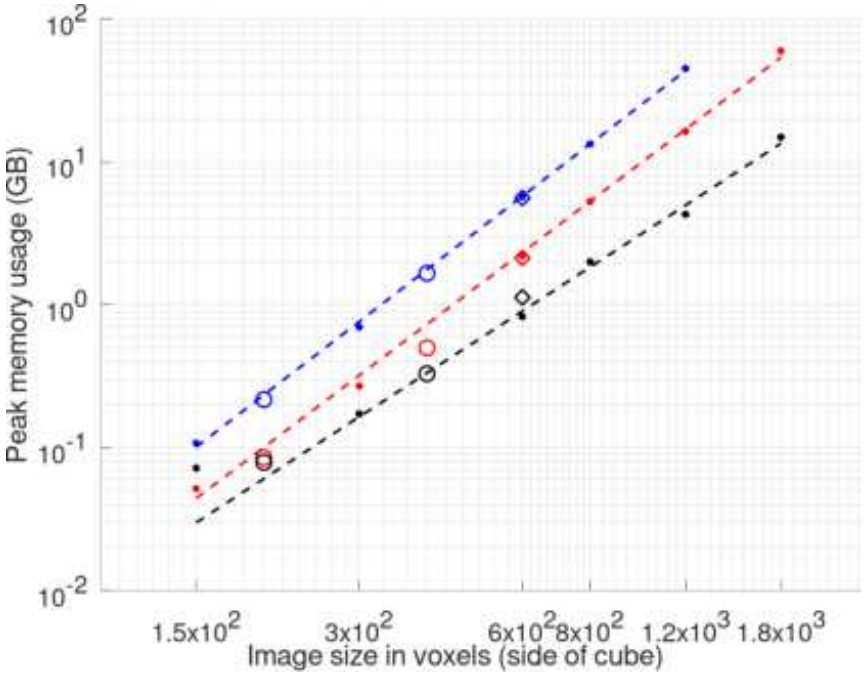


FIGURE 3.28: Peak memory usages in GB on a set of artificially generated honeycomb foams with constant cell size. In blue: watershed (as implemented in the ITK library [69]). In red and black: proposed image analysis with and without streaming, respectively. Discontinuous lines show the global trends for each method. Diamonds show the peak memory usages for each method on the real-world data set presented in Section 3.13.1. Circles show the peak memory usages on artificial foam data sets generated by random inclusions using the DN-CT-SCAN model [79] described in section 4.1.

3.13 Application to foam reconstruction

- **Aim**: qualitatively visualise the fits of ellipsoids using the proposed processing image analysis procedure applied to two real–world foams (one open, and one closed); starting from their respective two 3D CT-scan images.
- **Input**: 3D raw CT-scan image of a foam.
- **Output**: set of fitted ellipsoids to the foam's cells and their associated polyhedra.

3.13.1 Open foam

Data acquisition

A cubic aluminium foam ($AlSi_7Mg_{0.3}$) with an edge length of $15 \times 15 \times 15\,mm^3$ and a pore size of 20 pores per inch (ppi) was purchased from *Celltec Material GmbH*, Dresden, Germany. X–ray computed tomography – performed at the Fraunhofer Institute for Non-destructive Testing (IZFP), Saarbrücken, Germany – was used to determine the real microstructure of the foam. The scans were performed with a resolution of 1984×1984 pixels per layer and a voxel size of $12\,\mu m$. From the raw scanned images, a set of 673 2D-images with a resolution of 711×711 pixels each has been extracted (see Figure 3.30).

Image analysis step

The above set of described images contains void boundary regions with no struts at all. Therefore a subsample made of $613 \times 598 \times 621$ voxels was extracted in order to keep only the relevant data. From this subsample, ellipsoids were generated with the procedure presented in Section 2.3. Table 3.3 gives the parameters that were used. In this subsample each voxel is of length $\approx 24\,\mu$m in all the directions. Figure 3.31[10] shows the ellipsoids obtained in a sub-region cropped, for visualisation purposes, between the 127th and the 484th voxels in each direction. From the images, it can be noticed that one cell corresponds unequivocally to one ellipsoid. Quantitatively, if feature voxels represent the struts (in red in the images), 69% of the non-feature voxels (respectively only 0.3% of the feature voxels) are located inside the ellipsoids; indicating that the ellipsoids indeed fit the cells and do not cross the struts.

Growing auxiliary ellipsoids allows obtaining a much better fit of the cells as shown in Figure 3.32. Indeed, 96% of the non-feature voxels are now located inside at least one ellipsoid, while still only 0.3% of the feature voxels are located inside an ellipsoid. However, this better fit comes at the expense of optimising a much larger number of ellipsoids (the number of auxiliary ellipsoids grows linearly with the number of parent ellipsoids). Nevertheless, this is not an issue for the algorithm of R. Deits et al. [39] as only the number of associated optimisation problems will linearly grow and not their dimensions.

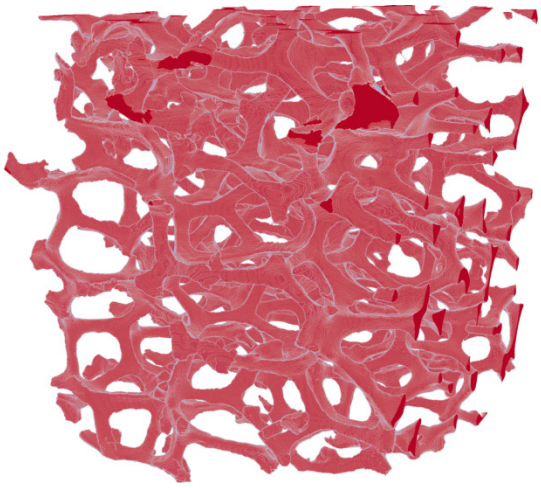


FIGURE 3.30: 3D–image extracted from raw CT-scans of a cubic aluminium foam.

TABLE 3.3: Parameters used in the image analysis of the CT–scan data presented in Section 3.13.1.

Parameter	**Value**	**Description**
Nr	10	Radius of the neighbouring region for testing local maxima candidates.
ϵ	10^{-5}	Hessian threshold for selecting local maxima.
σ	1.5	Gaussian standard deviation used for the convolution kernel.
τ	0.45	Relative intersection volume threshold.
$\Delta\theta$	10°	Azimuthal angle increment for discretising ellipsoids.

[10]Note: All figures in this section were generated using the *Paraview* software [2]. Surfaces of ellipsoids were obtained using the tool "contour" of *Paraview*.

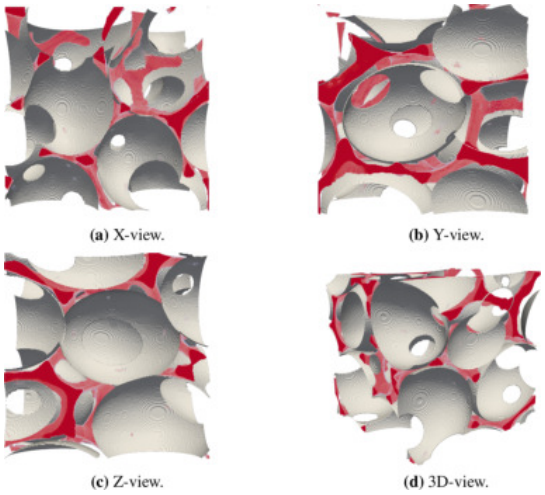


FIGURE 3.31: Surfaces of the parent ellipsoids (gray) obtained by the proposed image analysis steps and "struts" voxels extracted from the raw CT-scans showed for visual comparison (red). Cropped views of the $613 \times 598 \times 621$ set for visualisation purposes.

3.13.2 Closed foam

In order to demonstrate the versatility of the presented reconstruction process, it has been applied to the case of a closed foam. The image analysis steps and the reconstruction of the geometry are the same as described in the previous sections.

Data acquisition

A cubic polypropylene foam with 4% embedded carbon nanotubes with an edge length of $750 \times 750 \times 750\,\mu m^3$ was manufactured by F. Wan [177] at the University of Liège, Belgium. X-ray computed tomography scans where performed at the same location with a resolution of 670×670 pixels per layer and a voxel size of $1.12\mu m$, with a total of 670 layers. Figure 3.33 reports the obtained data set as well as the different image analysis steps applied to this foam.

Image analysis step

The proposed image processing steps described in Figure 2.3 have been applied to the above data set with the parameters given in Table 3.4. Figure 3.34 reports the generated ellipsoids, while Figure 3.35 gives some views of the surface generated by the auxiliary ellipsoids. In this later figure, it may be noticed some holes in the reconstructed surface, despite the foam being closed. This is due to the fact that some cell walls were locally to thin to be resolved by the CT-scan. As a result, some cells may appear as artificially partially open.

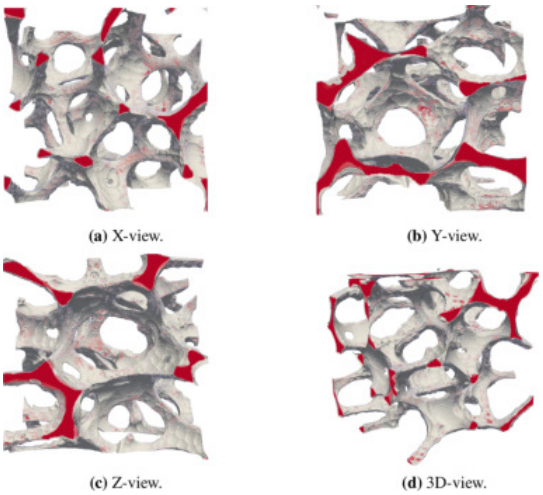
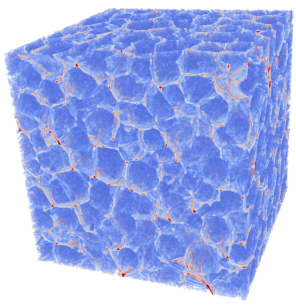


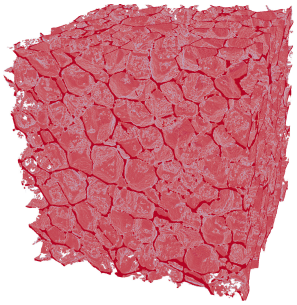
FIGURE 3.32: Surfaces of the auxiliary ellipsoids (gray) obtained by the proposed image analysis steps and "struts" voxels extracted from the raw CT-scans showed for visual comparison (red). Cropped views of the $613 \times 598 \times 621$ set for visualisation purposes.

TABLE 3.4: Parameters used in the image analysis of the CT–scan data presented in Section 3.13.2.

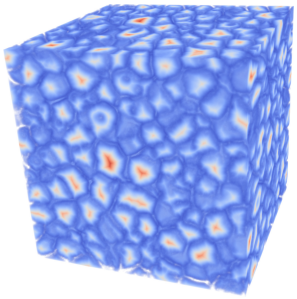
Parameter	Value	Description
N_r	5	Radius of the neighbouring region for testing local maxima candidates.
ϵ	10^{-1}	Hessian threshold for selecting local maxima.
σ	1.5	Gaussian standard deviation used for the convolution kernel.
τ	0.1	Relative intersection volume threshold.
$\Delta\theta$	10°	Azimuthal angle increment for discretising ellipsoids.



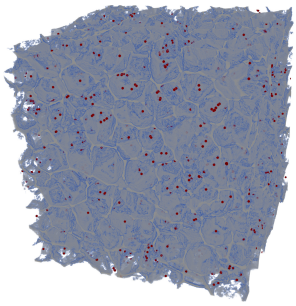
(a) Original CT-scan image.



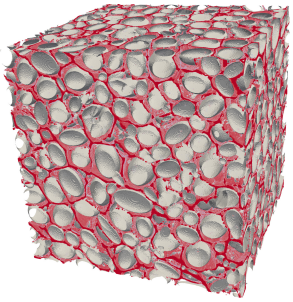
(b) Threshold image.



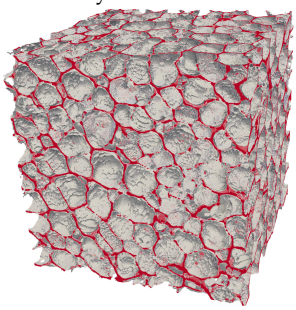
(c) Distance transform image. Low values represented in blue, high values represented in yellow and red.



(d) Local maxima (red balls), exaggerated for visualisation purposes.

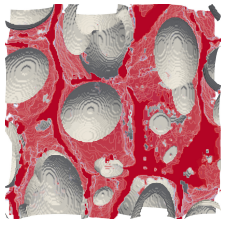


(e) Parent ellipsoids (gray surfaces) in their respective cell.

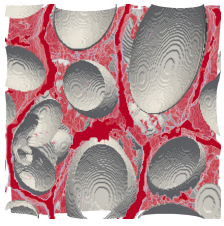


(f) Surface reconstruction obtained with the auxiliary ellipsoids.

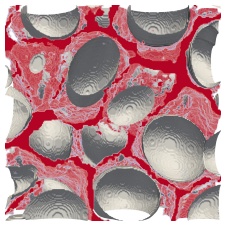
FIGURE 3.33: Example of the proposed processing steps on a 3D CT–scan image of a closed polypropylene foam with 4% embedded carbon nanotubes manufactured by F. Wan [177].



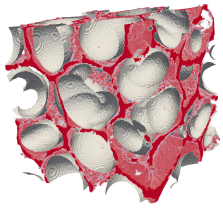
(a) X-view.



(b) Y-view.

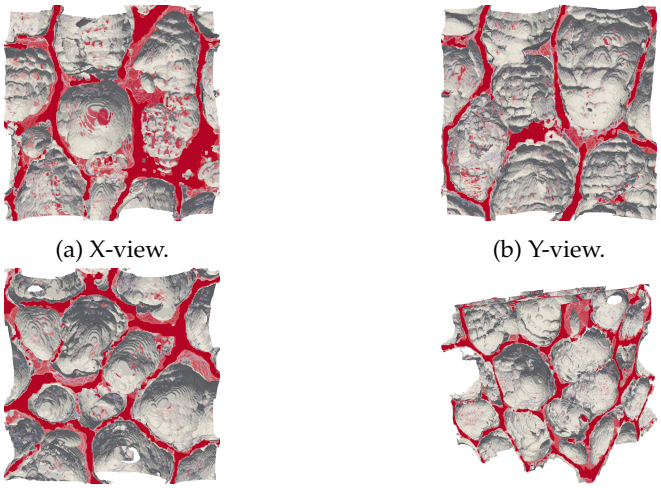


(c) Z-view.



(d) 3D-view.

FIGURE 3.34: Surfaces of the parent ellipsoids (gray) obtained by the proposed image analysis steps and "walls" voxels (red). Cropped views of the $670 \times 670 \times 670$ set for visualisation purposes.



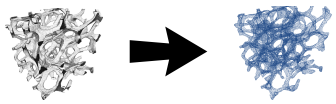
(a) X-view. (b) Y-view.

(c) Z-view. (d) 3D-view.

FIGURE 3.35: Surfaces of the auxiliary ellipsoids (gray) obtained by the proposed image analysis steps and "walls" voxels (red). Cropped views of the $670 \times 670 \times 670$ set for visualisation purposes.

Chapter 4

Reconstruction of the geometry



- **Aim**: obtaining a geometric representation of the microstructure of a foam that can be used for finite element simulations (see Section 5).
- **Input**: set of ellipsoids fitted to the cells of a foam.
- **Output**: geometric surface representation of the microstructure of the foam.

In what follows two different geometrical models (namely the *DN-CT-SCAN* model and the *Ellipsoidal Model*) are used for reconstructing the geometry of a foam and are compared against the experimental image data. Both of these models use, as starting point, the ellipsoids and/or their associated polyhedra obtained from the proposed image analysis procedure given in Section 2.3.

The *DN-CT-SCAN* model is initialised with the parent ellipsoids or associated parent polyhedra[1]. The philosophy of this model is then to reproduce an approximation of the geometry of a given foam using level sets. These level sets are described as the zeros of combination of distance functions. It is possible to adjust the obtained approximation of the geometry by considering different distance functions and the combination thereof. However, it is still not clear, even if there are some guidelines from Reference [79], how these distance functions should be chosen and how to combine them. Finally, a tetrahedral mesh is generated from the level sets.

The *Ellipsoidal Model* relies on the parent and auxiliary ellipsoids. These ellipsoids are discretised in a set of points and associated normals. As the parent and auxiliary ellipsoids may overlap, points and associated normals located strictly inside any given ellipsoid are discarded. The other points and associated normals are located on the so–called "exterior" surface of the set of ellipsoids. Then, a closed surface is reconstructed using a Poisson surface reconstruction. Eventually this surface is output to the "geo" file format used by the software *GMSH* [54] which ultimately generates a tetrahedral mesh of the geometry.

[1] A parent polyhedron is simply the polyhedron associated to its corresponding parent ellipsoid.

As it can be noticed, these two models are rather different. On one hand, the *DN-CT-SCAN* model relies on distance functions that can be tuned in order to control wall thicknesses or struts sections. It can be used to obtain a *idealised* Representative Volume Element (RVE), similarly as other earlier models using, e.g., Laguerre tessellations (see Section 1.2.2), but with refinements regarding the control of local geometric features (see Section 1.2.3). On the other hand, the *Ellipsoidal Model* is rather oriented *direct discretisation* (see Section 1.2.1) and tries to avoid, or at least alleviates, issues common to direct discretisation models such as memory limitations when handing large data sets. At the same time it tries the faithfully reproduce local geometric features, including defects such as incomplete struts or wholes in cell walls; defects that can not be reproduced (at least trivially) by idealised models such as the *DN-CT-SCAN* model. It is worth noting that the shapes of the walls and struts are automatically reproduced based on the CT-scan image used for computing the ellipsoids. Thus, contrary to the *DN-CT-SCAN* model, the *Ellipsoidal Model* does not require any parameter tuning in order to insure a faithful reproduction of the microstructure of a foam.

4.1 Reconstruction using the DN-CT-SCAN model

In Reference [79], the authors have developed a method to extract open foam morphologies from inclusions packings using distance functions as described in Reference [156]. In this first model, the reconstruction of the geometry uses the set of ellipsoids that have been generated previously. The ellipsoids can then be used as the initial packing to extract distance functions that can then be treated to obtain open foam RVEs. This method presents the advantage of using experimental information to improve the fidelity.

As described in Reference [156], nearest neighbour distance functions, $DN_k(\mathbf{x})$ in a point $\mathbf{x}$, are defined as the distance from the considered point to the k-th nearest inclusion. Figure 4.2 illustrates some nearest neighbour distance functions on a spherical packing. This step ensures that the distance functions are not affected by the boundary inclusions. The distance functions are then used to extract the modified “Plateau" function,

$$O_P(\mathbf{x}) = \frac{(DN_3(\mathbf{x}) + DN_2(\mathbf{x}))}{2} - DN_1(\mathbf{x}) \tag{4.1}$$

as described in Reference [79], where $DN_1(\mathbf{x})$, $DN_2(\mathbf{x})$ and $DN_3(\mathbf{x})$ are the 1st, 2nd and 3rd neighbour distance functions respectively. The “Plateau" function enables the extraction of the 3-sided struts.

The evaluation of the nearest distance field did not bring particular difficulty when starting from a sphere packing as in Reference [79]. However, in the present case, it is possible that the ellipsoids used as reference and their associated polyhedra intersect (see Figure 4.3a). In that case, this distance field DN_k is not monotonically evolving in the intersection region. This problem can be avoided by introducing an offset to the ellipsoid surface. In Reference [183], the authors have built *ad-hoc* level set functions using previously computed distance fields. Similar *ad-hoc* level set functions can be built around the ellipsoids that can then be manipulated to remove residual interpenetrations by introducing a gap between two selected ellipsoid based inclusions (see Figure 4.4).

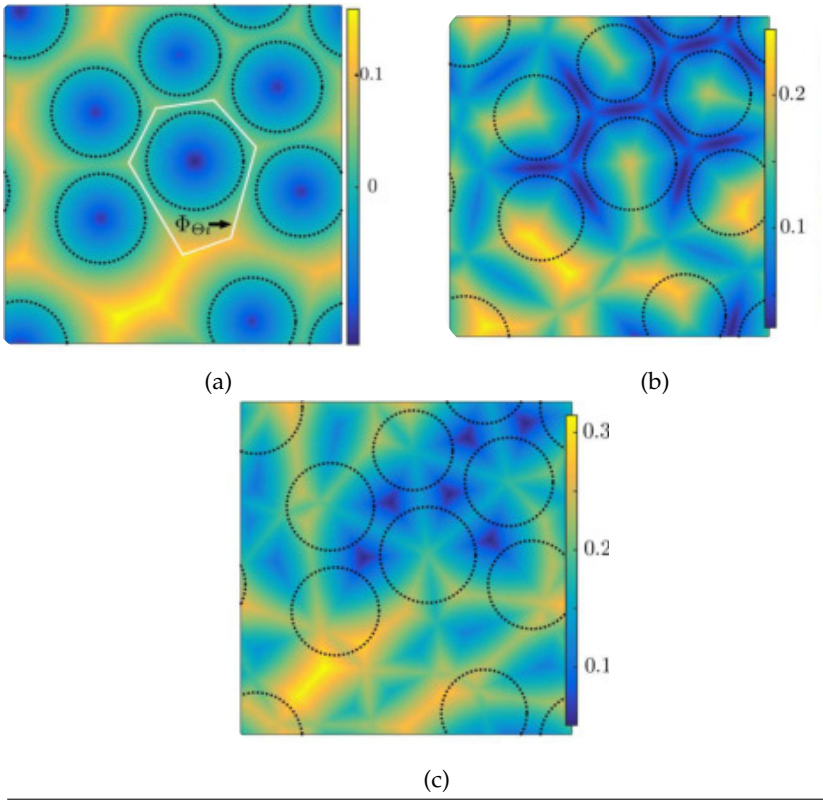


FIGURE 4.2: Functions (a) DN_1, (b) DN_2, and (c) DN_3. Image from Figure 3.5 in Reference [116].

The distance field associated to an inclusion i, denoted as DS_i, is defined as the signed distance field which is negative inside the inclusion and positive outside. The distance fields, DS_i and DS_j, resulting from two inclusions i and j, can be used to determine their mutual intersection volume as

$$\max(DS_i(\mathbf{x}), DS_j(\mathbf{x})) < 0, \quad \forall i \neq j. \tag{4.2}$$

The minimum distance to every inclusion other than inclusion i, $DO_i(\mathbf{x})$, is given by:

$$DO_i(\mathbf{x}) = \min(DS_j(\mathbf{x}) \ : \ \forall i \neq j), \tag{4.3}$$

And an *ad-hoc* level set function , $O_i(\mathbf{x})$, can be then extracted as follows:

$$O_i(\mathbf{x}) = \max(DS_i(\mathbf{x}), (DS_i(\mathbf{x}) - DO_i(\mathbf{x}))). \tag{4.4}$$

The zero level associated to the above function gives the ellipsoids in contact as shown in Figure 4.3b.

The necessary gap can be introduced as an offset c such that:

$$O_i^g(\mathbf{x}) = \max(DS_i(\mathbf{x}), (DS_i(\mathbf{x}) - DO_i(\mathbf{x})) + c), \tag{4.5}$$

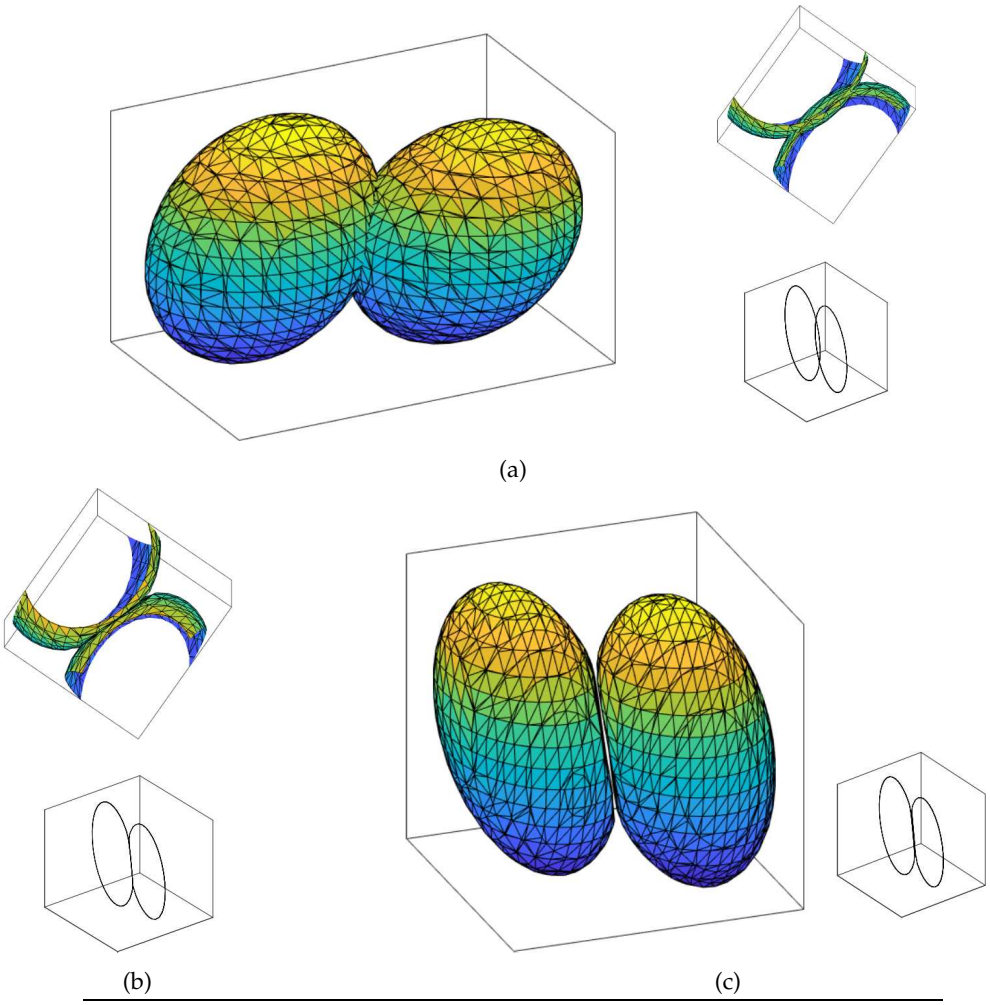


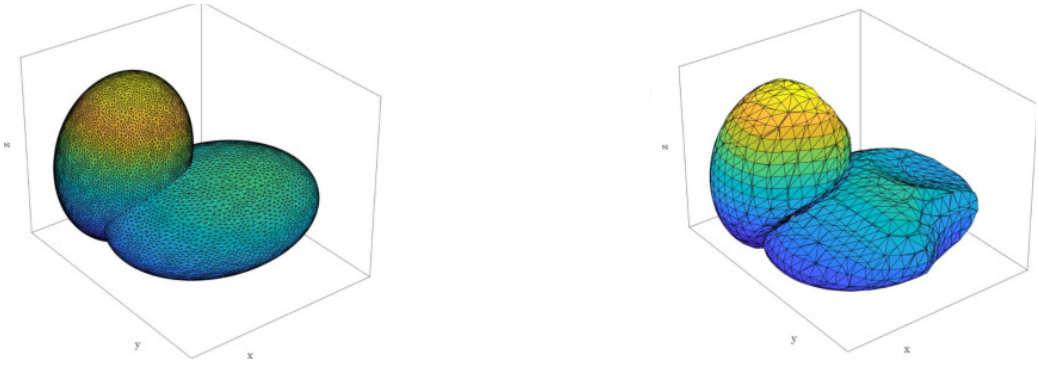
FIGURE 4.3: Distance fields and treatment of inclusions for the DN-CT-SCAN model. (a) Intersecting ellipsoids. The interpenetration can be seen by the sectional plot. (b) Post-processing of ellipsoids to avoid residual inter- penetrations, the ellipsoids now share a common face. (c) Ellipsoids after post-processing and introduction of a gap, the ellipsoids do no longer share a common face. Image from Figure 3.13 in Reference [116].

With the zero level of $O_i^g(\mathbf{x})$ resulting in the necessary inclusion surface of the inclusion i, as depicted in Figure 4.3c.

The function $O_i^g(\mathbf{x})$ is then used instead of the $DS_i(\mathbf{x})$ for refreshing the previously computed functions DN_k:

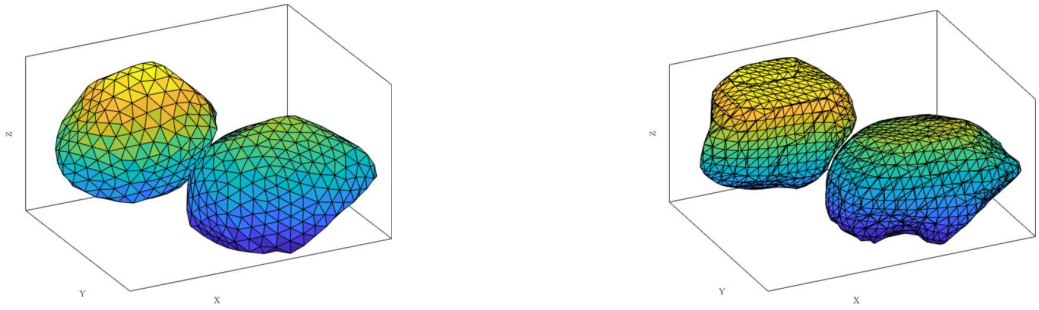
$$DN_k(\mathbf{x}) = \min_j(O_j^g(\mathbf{x})),\ j \in J_k(\mathbf{x}) \tag{4.6}$$

Where $J_k(\mathbf{x})$ denotes the set of k–th neighbours from position $\mathbf{x}$.



(a) (b)

FIGURE 4.4: (a) Ellipsoids obtained by processing the CT–scan; (b) Post–processing of ellipsoids to avoid residual interpenetration using the method described in Reference [183].



(a) (b)

FIGURE 4.5: (a) Polyhedra obtained by processing the CT-scan; (b) Post–processing of polyhedra to avoid residual interpenetration using the method described in Reference [183].

The ellipsoids can also be replaced by the polyhedra as the basis inclusion considering that the polyhedra are closer to the foam geometry (see Figure 4.5). Indeed, as described in Reference [39], polyhedra are defined by the interior region delimited by oriented planes. These planes are determined via the surfaces of the struts and tangents to the surfaces of the associated ellipsoids. Given that the polyhedra always contain their associated ellipsoid but not any strut, they are closer to the foam geometry than ellipsoids. The distance fields for the polyhedra can be similarly generated as for the ellipsoids.

The ellipsoids and polyhedra obtained from the CT-scan in Section 2.3 have the problem of boundary effect due to insufficient information at the boundaries to extract statistically representative inclusions. In order to analyse RVEs that are similar in size to those that were tested experimentally, it is possible to continue the process of adding inclusions using random sequential addition (RSA), as described in Reference [43]. The distance fields obtained previously ensure that the new spherical inclusions are statistically valid to extract open foam morphologies [79]. Once the RSA process is complete, an open foam morphology can be extracted using the computed distance fields.

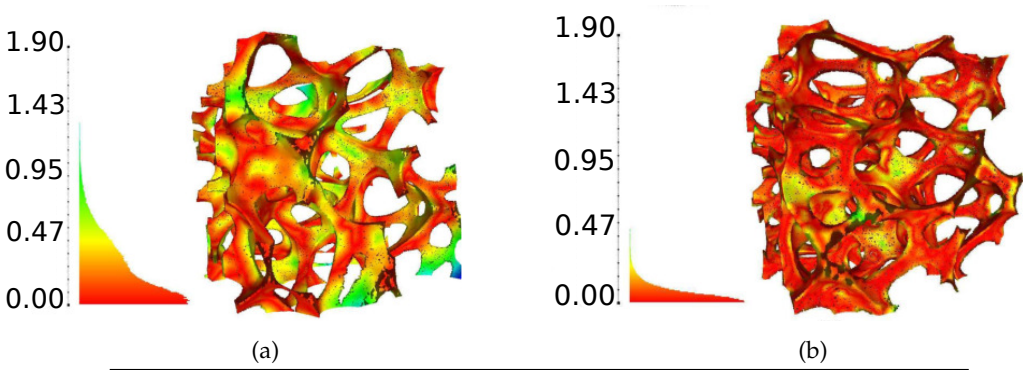


FIGURE 4.6: Visual representation of the Hausdorff distance in mm between the CT-scan image and the meshes extracted using DN-CT-SCAN. (a) Use of ellipsoidal packing. (b) Use of polyhedral packing.

Finally, a finite element mesh is generated using the tool developed in Reference [43]; where dynamic node repositioning based on level set functions is used to build high quality conforming meshes using the strategy developed in Reference [128].

For both packings (ellipsoids and polyhedra), the obtained porosity is around 93.5% which is very similar to that of the original foam of 93%. Thus, the relative densities of the two samples are considered close enough. The obtained RVE mesh was refined as explained in Reference [43] and a tetrahedral mesh was extracted using the Tetgen software [153]. Figure 4.6 shows the Hausdorff distances, computed using the software Meshlab [34] with 500.000 sample points, between the extracted RVEs and the thresholded CT-scan image for both ellipsoid and polyhedral packing. It can be seen, that the polyhedral packing leads to a better representation for the DN-CT-SCAN model. The maximum Hausdorff distance between the reconstructed geometry and the struts is 1.48 mm (corresponding to a relative maximum error of 9.9 %) when using the ellipsoid packing, while it reduces to 0.89 mm (corresponding to a relative maximum error of 5.9 %) with the polyhedral packing. For the polyhedral packing, it can be seen that for a statistically validated representation of the actual foam, the extracted RVE is very close in the reproduction of the various morphological features.

Discussion It should be noted that the DN-CT-SCAN model has also been used without using the provided ellipsoids and polyhedra by the proposed image analysis procedure in Section 2.3. For instance, in Reference [79], the methodology for obtaining an RVE from a given foam sample is based on the extraction of random tessellations from inclusion packings, following predetermined statistical packing distribution criteria. Then, in order to ensure obtaining a representative model, various distance fields need to be combined; which is not a trivial task. The above results show that the proposed image analysis procedure can efficiently "feed" the DN-CT-SCAN model. This allows the DN-CT-SCAN model to avoid the use of a random packing model and offers two main advantages: First there is no need to perform the challenging task of determining a statistical packing distribution criteria [1]. Second, the provided distribution of ellipsoids/polyhedra is guaranteed to be the correct one for obtaining a satisfactory matching model to the data, as shown in Figure 4.6.

4.2 Reconstruction using the Ellipsoidal Model

In this second model, the reconstruction of the geometry uses the set of parent and auxiliary ellipsoids that have been generated previously in Section 2.3. The procedure for reconstructing the geometry from the set of ellipsoids is conceptually simple. First, all ellipsoids (parent and auxiliary) are discretised to a set of points and associated normals. Then this set of points and normals is used for generating a closed triangulated surface using a Poisson surface reconstruction. Finally, this triangulated surface is meshed using the GMSH software [54]. What follows describes the process in details.

4.2.1 Discretisation of ellipsoids

First the iso-surfaces of the parent and auxiliary ellipsoids are discretised into a set of points and associated normals using the same discretisation technique as in step 6 of Section 2.3. The angle increment $\Delta\theta$ is the azimuthal angle increment for discretising ellipsoids and depicted in Figures 4.7a and 4.7b. Using this technique ensures that the set of points is approximatively homogeneous. Moreover, the ellipsoid's iso–surfaces ISO_{param} on which the points lie can be chosen so that the porosity of the final reconstructed geometry matches as much as possible the measured porosity of the foam. It may be noticed that the auxiliary ellipsoids already closely fit the real foam geometry. As a consequence, this parameter usually does not need to be fine tuned and can generally be kept at its default value $ISO_{param} = 1.0$; which corresponds to the ellipsoid's surfaces. Algorithm 36 describes how to implement this in practice.

Algorithm 36 Given an ellipsoid $\mathcal{E}$, an azimuthal angle increment $\Delta\theta$, and a factor α, uniformly discretise the iso–surface of value α of $\mathcal{E}$ in a set of points with associated normals.

Require: Ellipsoid $\mathcal{E} = \left\{\mathbf{x} \in \mathcal{R}^3 \mid (\mathbf{x}-\mathbf{c})^t GG^t (\mathbf{x}-\mathbf{c}) \leq 1\right\}$, angle increment $\Delta\theta$, factor $\alpha = ISO_{param}$.

procedure DISCRETIZEELLIPSOIDWITHNORMALS($\mathcal{E}$, $\Delta\theta$, α)
  Set of points $P \leftarrow \emptyset$
  Set of normals $N \leftarrow \emptyset$
  for $\theta_i = 0; \theta_i \leq \pi; \theta_i \leftarrow \theta_i + \Delta\theta$ **do**
    Compute $\Delta\phi_i$ as given by eq. 2.85.
    for $\phi_{ij} = 0; \phi_{ij} \leq 2\pi; \phi_{ij} = \phi_{ij} + \Delta\phi_i$ **do**
      Compute $\mathbf{y} = (y_0, y_1, y_2)$ as:
      $$\begin{cases} y_0 &= \alpha \sin\theta_i \cos\phi_{ij} \\ y_1 &= \alpha \sin\theta_i \sin\phi_{ij} \\ y_2 &= \alpha \cos\theta_i \end{cases}$$
      $P \leftarrow P \cup \left\{\mathbf{c} + G^{-t}\mathbf{y}\right\}$.
      $N \leftarrow N \cup \left\{ \left(GG^t(\mathbf{y}-\mathbf{c})\right) \,/\, \left\|GG^t(\mathbf{y}-\mathbf{c})\right\| \right\}$
    end for
  end for
  return P, N
end procedure

Then, points associated to a given ellipsoid E_i, but located strictly inside a different ellipsoid E_j (w.r.t. the considered iso–surface) are discarded. This ensures only surface points and associated normals are kept, even if the ellipsoids are intersecting (see Figure 4.7c). These remaining points and associated normals are hereafter called *exterior* points. Algorithm 37 describes how these points are obtained.

Algorithm 37 Given a set of points $\mathcal{P}$, a factor α and a set of ellipsoids $\mathcal{E}$ arranged inside a R*-tree data structure, returns *exterior* points. Note: boundaries are NOT taken into account here. See algorithm 38 for that.

Require: $\mathcal{E} = \{\mathcal{E}\}_{1 \leq i \leq m}$ a set of ellipsoids arranged in a R*-tree data structure.
Require: $\mathcal{P} = \{P_j\}_{1 \leq j \leq n}$ a set of points.
Require: $\alpha = ISO_{param}$ an iso-surface value. ▷ Must have the same value os the one used in algo. 36.

1: **procedure** EXTRACTPOINTSOUTSIDEELLIPOIDS($\mathcal{E}$, $\mathcal{P}$, α)
2: $\quad\mathcal{P}_{out} = \emptyset$
3: $\quad$**for** $P \in \mathcal{P}$ **do**
4: $\quad\quad\mathcal{E}^* \leftarrow \mathcal{E}$. QUERYNEIGHBOURHOOD($P$) ▷ Queries ellipsoids in the neighbourhood of $\mathcal{P}$
5: $\quad\quad$**if** $\#\mathcal{E}^* \leq 1$ **then** ▷ Point inside at most one ellipsoid
6: $\quad\quad\quad$**if** $\#\mathcal{E}^* = 0$ **then**
7: $\quad\quad\quad\quad\mathcal{P}_{out} \leftarrow \mathcal{P}_{out} \cup P$
8: $\quad\quad\quad$**else** ▷ Check if P strictly inside ellipsoid
9: $\quad\quad\quad\quad\mathcal{E} = \mathcal{E}_1^*$
10: $\quad\quad\quad\quad\left(\mathbf{c}, M = GG^t\right) \leftarrow \mathcal{E}$ ▷ Get ellipsoid's centre and metric
11: $\quad\quad\quad\quad$**if** $(\mathbf{c} - P)^t M (\mathbf{c} - P) \geq \alpha$ **then** ▷ P not inside the α iso-surface of ellipsoid $\mathcal{E}$
12: $\quad\quad\quad\quad\quad\mathcal{P}_{out} \leftarrow \mathcal{P}_{out} \cup P$
13: $\quad\quad\quad\quad$**end if**
14: $\quad\quad\quad$**end if**
15: $\quad\quad$**end if**
16: $\quad$**end for**
17: $\quad$**return** $\mathcal{P}_{out}$
18: **end procedure**

4.2.2 Boundary management

However, at the boundaries there is insufficient information to extract statistically representative ellipsoids. In order to avoid spurious reconstruction effects a subset of suitable surface points is determined from the set of surface points as follows. First, clipping planes are applied in order to discard points too close to the boundaries (Figure 4.7d). Second, "holes" created by the clipping planes in the set of points are filled by adding new points with a similar density as the density of the whole set of surface points (Figures 4.7e and 4.7f). Associated normals to the new points are simply set as the corresponding normals of each clipping plane. Algorithm 38 is an improvement of algorithm 37 that takes into account the boundaries. Thus, in practice, algorithm 37 is not used and is only mentioned here for pedagogic purposes.

Algorithm 38 Given a set of points $\mathcal{P}$ and associated normals $\mathcal{N}$, a set of planes $\mathcal{Q}$, a factor α and a set of ellipsoids $\mathcal{E}$ arranged inside a R*-tree data structure, returns *exterior* points. Note: improved version of algorithm 37 that takes into account boundaries.

Require: $\mathcal{E} = \{\mathcal{E}\}_{1 \leq i \leq m}$ a set of ellipsoids arranged in a R*-tree data structure.
Require: $\mathcal{P} = \{P_j\}_{1 \leq j \leq n}$ a set of points, and $\mathcal{N} = \{N_j\}_{1 \leq j \leq n}$ a set of associated normals.
Require: $\mathcal{Q} = \{Q_k\}_{1 \leq k \leq o}$ a set of planes, and $\alpha = ISO_{param}$ an iso-surface value. ▷ Must have the same value as the one used in algo. 36.

procedure ExtractPointsOutsideEllipoidsWithPlanes($\mathcal{E}$, $\mathcal{P}$, $\mathcal{N}$, $\mathcal{Q}$, α)
  $\mathcal{P}_{out} = \emptyset$
  $\mathcal{N}_{out} = \emptyset$
  $avg \leftarrow$ average point distance in $\mathcal{P}$
  for $Q \in \mathcal{Q}$ **do**
    $\{\mathcal{P}_Q, \mathcal{N}_Q\} \leftarrow$ discretise Q as a grid of points of average dist. avg
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_Q$. $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_Q$. ▷ Add points and normals of Q.
  end for
  for $P \in \mathcal{P}$ and N associated to P **do**
    $pointInsidePlanes \leftarrow true$
    for $Q \in \mathcal{Q}$ **do** ▷ Loop on planes to see if the current point is on the "inside" side of all planes.
      $(\mathbf{q}, \mathbf{n}) \leftarrow$ point and normal of plane Q.
      $\mathbf{v} = P - \mathbf{q}$.
      if $\mathbf{n}.\mathbf{v} > 0$ **then** ▷ Point not in the "inside" side of the current plane. Stop here.
        $pointInsidePlanes \leftarrow false$. Break.
      end if
    end for
    if NOT $pointsInsidePlanes$ **then** ▷ Point not on the "inside" sides of all planes, test the next point.
      Continue
    end if
    $\mathcal{E}^* \leftarrow \mathcal{E}$. QueryNeighbourhood($P$) ▷ Queries ellipsoids in the neighbourhood of $\mathcal{P}$
    if $\#\mathcal{E}^* \leq 1$ **then** ▷ Point inside at most one ellipsoid
      if $\#\mathcal{E}^* = 0$ **then**
        $\mathcal{P}_{out} \leftarrow \mathcal{P}_{out} \cup P$. $\mathcal{N}_{out} \leftarrow \mathcal{N}_{out} \cup N$
      else ▷ Check if P strictly inside ellipsoid
        $(\mathbf{c}, M = GG^t) \leftarrow \mathcal{E}_1^*$ ▷ Get ellipsoid's centre and metric
        if $(\mathbf{c} - P)^t M (\mathbf{c} - P) \geq \alpha$ **then** ▷ P not inside the α iso-surface of ellipsoid $\mathcal{E}$
          $\mathcal{P}_{out} \leftarrow \mathcal{P}_{out} \cup P$. $\mathcal{N}_{out} \leftarrow \mathcal{N}_{out} \cup N$
        end if
      end if
    end if
  end for
  return $\{\mathcal{P}_{out}, \mathcal{N}_{out}\}$
end procedure

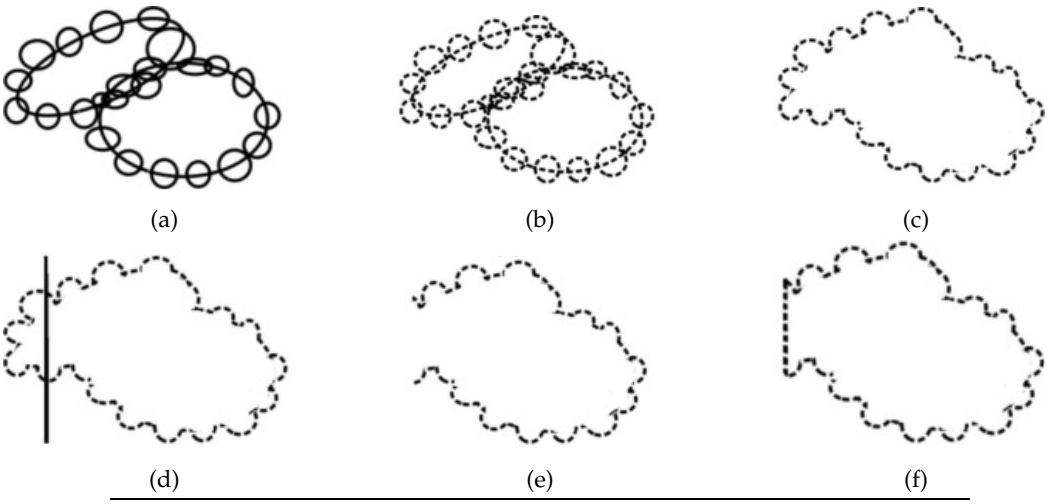


FIGURE 4.7: Schematics of the reconstruction of the geometry from the ellipsoids obtained in Section 2.3. (a) Obtained ellipsoids from Section 2.3. (b) Discretised ellipsoids. (c) Only points strictly not inside ellipsoids are kept. (d) Application of clipping planes for discarding points too close from boundaries. (e) Set of points with "holes". (f) "Holes" filled.

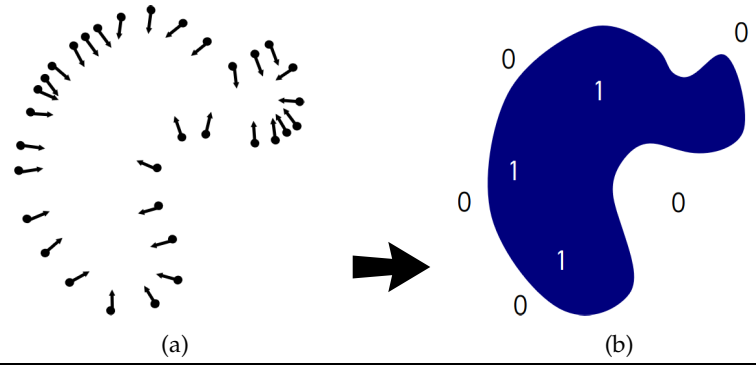


FIGURE 4.8: Poisson surface reconstruction: from a discrete set of oriented points belonging to a closed volume M (a), reconstruct its corresponding indicator function (b). Images from https://www.cse.iitb.ac.in/~cs749/spr2016/lecs/07_reconst.pdf.

4.2.3 Poisson surface reconstruction

Finally, in order to obtain the reconstructed geometry of the foam from the set of suitable points and normals $\vec{V}$, a Poisson surface reconstruction is performed [77] using the eponymous CGAL package[2]. The Poisson surface reconstruction is the process consisting in finding the indicator function χ_M best approximating a closed volume M for which only a discrete set of oriented points is available (see Figure 4.8). Once the indicator function found, it is then possible to extract an approximated surface corresponding to the set of oriented points.

[2] https://doc.cgal.org/latest/Poisson_surface_reconstruction_3

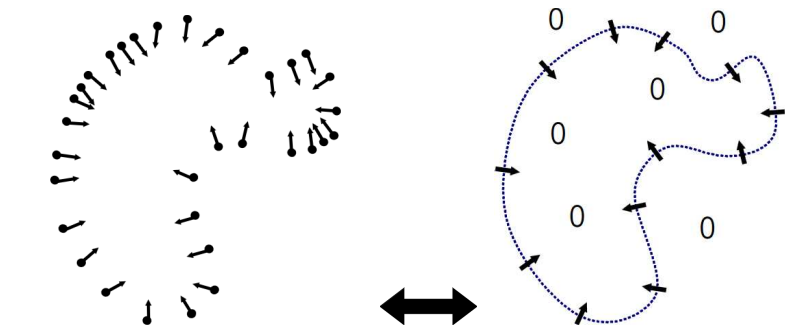


FIGURE 4.9: Relationship between a set of oriented points and the (smoothed) gradient of the indicator function. (a) Discrete set of oriented points. (b) Indicator gradient $\vec{\nabla}\chi_M$. Note: $\vec{\nabla}\chi_M$ is zero almost everywhere except on the boundaries. Image from https://www.cse.iitb.ac.in/~cs749/spr2016/lecs/07_reconst.pdf.

In order to obtain the indicator function χ_M from the set of oriented points $\vec{V}$, the Poisson surface reconstruction exploits a relationship between the set of points and the gradient of the (smoothed) indicator function $\vec{\nabla}\chi_M$[3] (see Figure 4.9). More precisely, this relationship is simply (see Reference [77] for a proof):

$$\vec{\nabla}\tilde{\chi}_M = \vec{V} \tag{4.7}$$

Where $\tilde{\chi}_M$ is χ_M smoothed.

However, in general, equation 4.7 does not admit an exact solution. Instead the best least–square approximate solution can be found by applying the divergence operator to form the Poisson equation (hence the name of the method):

$$\Delta\tilde{\chi}_M = \vec{\nabla} \,.\, \vec{V} \tag{4.8}$$

Equation 4.8 is then solved numerically using a method similar to finite elements on an octree mesh (see Reference [77] for details).

At each point the Poisson surface reconstruction method requires an associated normal. For a given point its normal is obtained as the local normal of its associated ellipsoid at that point position. Figure 4.10 shows the set of suitable points extracted from the data described in Section 3.13.1, using an azimuthal angle increment $\Delta\theta = 10^\circ$. The Poisson surface reconstruction algorithm provided in the CGAL package can be tuned via three parameters (see Table 4.1). These three parameters, denoted α, T_s and, S_a control the quality and refinement of the faceted surface which is obtained from the discretised ellipsoids. They are directly related to the parameters of the CGAL function *poisson_surface_reconstruction_delaunay*.

[3]Note that $\vec{\nabla}\chi_M$ is zero almost everywhere, except on the boundaries of the closed volume M.

The parameter α controls the bound for the minimum facet angle in degrees. It gives a guarantee that the to-be-computed finite–element mesh obtained from the reconstructed surface will not contains tetrahedra with triangular faces less than α degrees. T_s controls the maximum facet size with respect to the average point set spacing. It puts an upper bound to the sizes of the future tetrahedra in the to-be-computed finite–element mesh. Finally, S_a controls the surface approximation error with respect to the point set average spacing. Indeed, in general a Poisson reconstruction only approximates a given set of points and associated normals. So the obtained reconstructed surface may not interpolate all the points.

It should be noted that a Poisson surface reconstruction is not the only option for reconstructing a surface from an unorganised set of points. For instance, one may also consider the work of H. Hoppe et al (see References [64], [63], and [42])[4].

Ultimately, a B–REP of the surface geometry is then obtained as a "geo" file describing vertices positions, edges and faces relations that can be used by the GMSH software [54]. The advantage of having a meshable B-REP is that the 3D generated mesh can be, to some extent, "tuned" for satisfying some precise needs. For instance, the 3D generated mesh can be locally refined where needed. Moreover, the GSMH software easily allows associating several geometries together via boolean operations. For instance, one may consider to unite the surface geometry with a rectangular plate and conduct some simulations with a dedicated solver.

Open foam

Figure 4.11 shows an extracted mesh compared to the threshold CT-scan images for the open foam described in Section 3.13.1. It can be seen that the 3D mesh closely reproduces the different morphological characteristics of the foam, although the obtained porosity of 90.2% is slightly different from the experimental 93% porosity. For the Ellipsoidal Model this difference, as suggested by the Hausdorff distance in Figure 4.11, is due to the fact that the ellipsoidal model seems to accumulate more matter than needed at struts intersections. This problem could probably be alleviated by discretising the circumscribing auxiliary polyhedra instead of theirs associated auxiliary ellipsoids. Indeed, as each polyhedron contains its corresponding ellipsoid, the obtained porosity can only increase and sharp features should be better reconstructed. However, in any case, it should be noted that, despite the fact that the DN-CT-SCAN model using polyhedral packing presents a better porosity value than the Ellipsoidal model, both models present very similar structures and Hausdorff distances as it can observed in Figures 4.6b and 4.11. However, the maximum Hausdorff distance for the Ellipsoidal Modal is smaller with 0.58 mm (corresponding to a relative maximum error of 3.3 %).

It is worth noting that the *Ellipsoidal Model* is also able to reconstruct local features of the microstructure such as the typical tendency of struts to be thicker near their vertices than at their centre [138]. But this has also the capability to reproduce small defects as illustrated in Figure 4.12 without the need of some dedicated parametrisation.

[4] A C++ library is also available at `https://github.com/hhoppe/Mesh-processing-library`

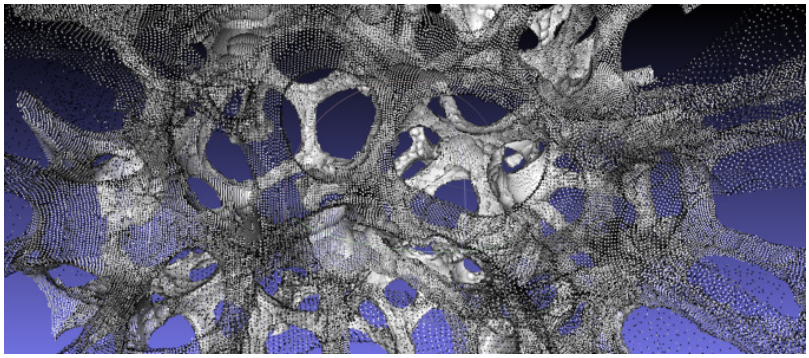


FIGURE 4.10: View inside the set of points extracted from the data described in Section 3.13.1.

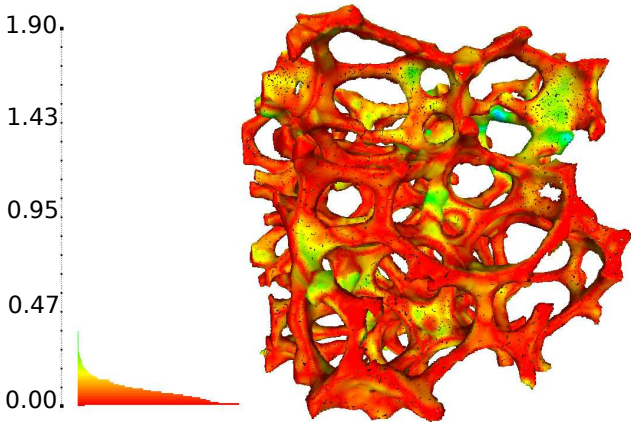


FIGURE 4.11: Visual representation of the Hausdorff distance in mm between the CT–scan image and the meshes extracted using Ellipsoidal Model for the open foam described in Section 3.13.1.

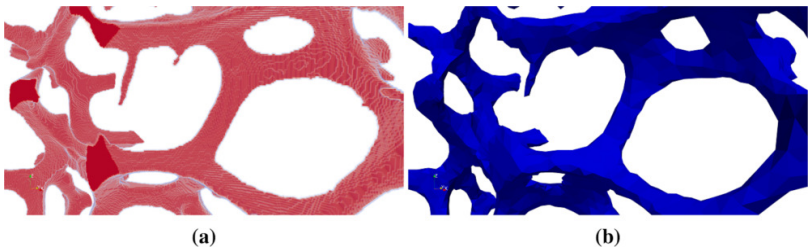


FIGURE 4.12: (a) Presence of a defect (spike) visible on the upper left-hand side of a 3D CT–scan image of a strut. (b) Reproduction of this defect by the *Ellipsoidal Model* for the open foam described in Section 3.13.1.

TABLE 4.1: Parameters used in the geometric reconstruction of the CT–scan data of the open foam presented in Section 3.13.1 for the Ellipsoidal Model.

Parameter	Value	Description
α	30°	Minimum facet angle.
T_s	600.0	Maximum facet size w.r.t. point set average spacing.
S_a	0.75	Surface approximation error w.r.t. point set average spacing.
ISO_{param}	1.0	Iso–surface parameter on which lie the extracted points.
$\Delta\theta$	10°	Azimuthal angle increment for discretising ellipsoids.

TABLE 4.2: Parameters used in the geometric reconstruction of the CT-scan data data of the closed foam presented in Section 3.13.2 for the Ellipsoidal Model.

Parameter	Value	Description
α	30°	Minimum triangle angle.
T_s	600.0	Maximum triangle size w.r.t. point set average spacing.
S_a	0.75	Surface approximation error w.r.t. point set average spacing.
ISO_{param}	1.0	Iso-surface parameter on which lie the extracted points.
$\Delta\theta$	2.5°	Azimuthal angle increment for discretising ellipsoids.

Closed foam

Figure 4.13 shows an extracted mesh compared to the threshold CT-scan images for the closed foam described in Section 3.13.2, using the parameters stated in Table 4.2. It can also be seen that the 3D mesh closely reproduces the different morphological characteristics of the foam. The maximum Hausdorff distance (computed using $5,000,000$ sample points) between the reconstructed geometry and the struts is $0.053\,mm$ (corresponding to a relative maximum error of 4.1 %).

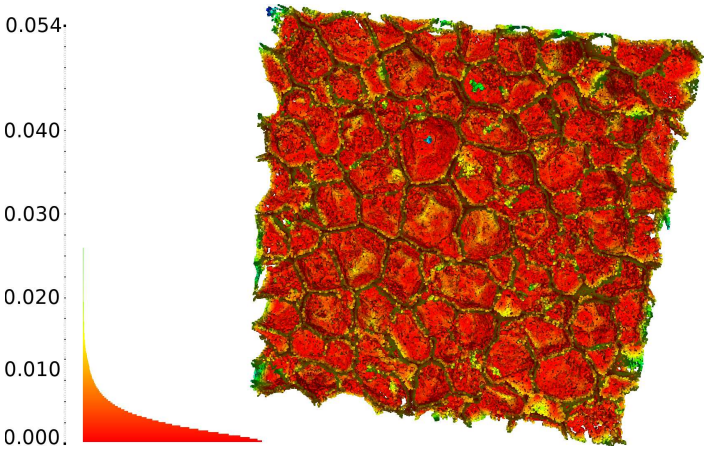
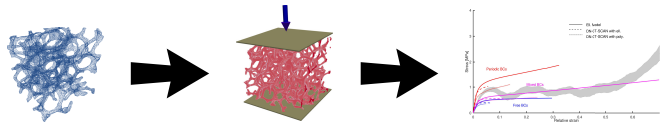


FIGURE 4.13: Visual representation of the Hausdorff distance in mm between the CT–scan image and the meshes extracted using Ellipsoidal Model for the closed foam described in Section 3.13.2.

Chapter 5

FEM simulations: uniaxial compression test on open foam sample



- **Aim**: given a tetrahedral mesh describing the microstructure of a real–world foam and suitable boundary and loading conditions, match by FEM simulation the strain–stress curves obtained experimentally.
- **Input**: mesh corresponding to a real–world foam and set of boundary and loading conditions.
- **Output**: stress–strain curves.

5.1 Summary

This chapter is structured as follow:

- Section 5.2 briefly describes some main computational approaches that exist for studying numerically the behaviours of cellular materials.
- Section 5.2.5 describes how the image analysis steps of chapter 2 and the geometric reconstructions (chapter 4) can be used to "feed" FEM simulations of foams.
- Sections 5.3, 5.4, and 5.5 explains how experimental data and material parameters were acquired for a real-world foam undergoing an uniaxial compression.
- Section 5.6 shows simulation results for the uniaxial compression of a real-world foam using different boundary conditions, using both the DN-CT-SCAN model and the Ellipsoidal Model described in chapter 4. These simulation results are compared against the acquired experimental data.
- Section 5.6.4 finally discusses the numerical convergence for both the DN-CT-SCAN and Ellipsoidal Models with respect to the number of elements used for the meshes.
- Section 5.6.5 finally briefly discuss a simulation result using the Gurson-Tvergaard-Needleman model.

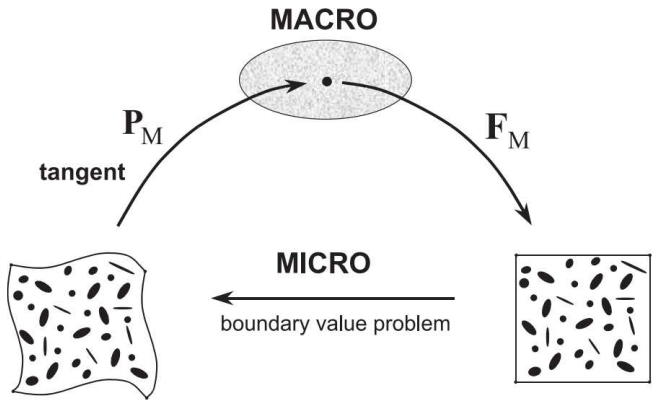


FIGURE 5.2: Computational homogenisation scheme. See text for details (Figure from Reference [84]).

5.2 Introduction

In experimental studies, cellular materials exhibit a wide variety of complex mechanical behaviours that are difficult to predict. This is due to several effects such as size effect [7], and localisation phenomena due to micro-buckling of thin components such as struts or walls [68, 14, 190, 127]. Three main computational approaches exist for studying numerically the behaviours of cellular materials such as foams.

1. The microscopic approach.
2. The macroscopic approach.
3. The multi-scale computational approach.

The first approach consists in fully discretising a full foam sample using standard finite element methods [56, 31, 104]. However, this approach rapidly requires to deal with an enormous amount of unknowns and is thus limited to relatively small samples. Dealing with larger foam samples with this approach is still a computational challenge for modern computers.

The second approach consists in replacing the cellular material by an equivalent continuous phenomenological material [60, 49]. Although more computationally efficient than the first approach, with this second approach, the material model and its parameters are difficult to identify. Moreover, this approach does not provide any insight about the evolution of the microstructure while a macroscopic loading is applied.

The last approach, also called the *multiscale* approach, is a combination of the first two above approaches. This technique lies in to the definition of two separate Boundary Value Problems (BVPs) at two separate scales. At the macroscopic level, the studied sample is considered as a continuum medium; but at each macroscopic material point a microscopic BVP is associated in order to take into account the effect of the microstructure. Each microscopic BVP is associated to a Representative Volume Element (RVE) which undergoes different microscopic boundary conditions associated to macroscopic quantities. This allows to incorporate both geometrical and

material non-linearites [84] into the numerical model. Tough this procedure does not provide a closed-form of the macroscopic material law, the stress-strain relation is always obtainable through the resolution of the BVPs. For a given macroscopic deformation gradient tensor $\mathbf{F}_M$, the stress $\mathbf{P}_M$, and the associated material tangent are estimated from the response of the microstructure (see Figure 5.2)

The multiscale approach (also called first–order approach) makes the assumption of separation of scales, which can be expressed as a set of inequalities:

$$l_{micro} << l_{RVE} << l_{macro} \tag{5.1}$$

Where l_{micro} represents the average pore size of the cellular material, l_{RVE} denotes the size of the domain on which local deformations occurs in the material, and l_{macro} represents the characteristics length over which the prescribed mechanical loading is varying.

The inequality:

$$l_{micro} << l_{RVE} \tag{5.2}$$

denotes the scale condition for an RVE to be valid, while the inequality:

$$l_{RVE} << l_{macro} \tag{5.3}$$

denotes the condition for the multiscale to be valid.

It may occur that the separation of scales assumptions 5.1 is no longer satisfied, e.g., when localisation and/or failure phenomena takes place. Enhanced schemes have been designed for tackling this issue. For instance, second-order FE^2 scheme were designed in References [47, 83], while continuous-discontinuous FE^2 schemes were developed in References [108, 123]. However, in most situations, a first order scheme proves to be sufficient and is a standard tool in computational homogenisation [109, 112, 161].

5.2.1 Macroscopic formulation

The macro-scale kinematics can be defined by:

$$\mathbf{F}_M = \mathbf{I} + \mathbf{u}_M \bigotimes \nabla_0 \tag{5.4}$$

Where:

- $\mathbf{F}_M$ is the macroscopic deformation gradient.
- $\mathbf{I}$ is the identity matrix.
- $\mathbf{u}_M$ is the macroscopic displacement field in the reference configuration.
- ∇_0 is the gradient operator with respect to the reference configuration.
- $\bigotimes$ denotes the outer product.

If one considers a body Ω, viewed as continuous, then continuum equilibrium equations read as:

$$\begin{array}{lll} \mathbf{P}_M(\mathbf{X}_M) \,.\, \nabla_0 & = 0, & \forall \mathbf{X}_M \in \Omega \\ \mathbf{P}_M(\mathbf{X}_M) \,.\, \mathbf{N}_M & = \mathbf{T}_M, & \forall \mathbf{X}_M \in \partial_N \Omega \end{array} \tag{5.5}$$

With $\mathbf{X}_M$ a material point in the body Ω, $\mathbf{P}_M$ the first Piola-Kirchhoff tensor, and $\mathbf{N}_M$ the local macroscopic unit normal to the surface $\partial_N \Omega$.

The associated boundary conditions are:

$$\mathbf{u}_M(\mathbf{X}_M) = \mathbf{u}_M^0, \forall \mathbf{X}_M \in \partial_D \Omega \tag{5.6}$$

$$\mathbf{T}_M(\mathbf{X}_M) = \mathbf{T}_M^0, \forall \mathbf{X}_M \in \partial_N \Omega \tag{5.7}$$

With the displacements $\mathbf{u}_M$ are constrained on the Dirichlet boundary $\partial_D \Omega$ and tractions $\mathbf{T}_M$ on reference unit surfaces are prescribed on Neumann boundary $\partial_N \Omega$.

Problem 5.5 along with the boundary conditions 5.6, 5.7 is completed by the stress-strain relationship at time t:

$$\mathbf{P}_M(t) = \mathcal{P}_M\{\mathbf{F}_M(t), \mathbf{Z}_M(t)\} \tag{5.8}$$

With $\mathbf{Z}_m(t)$ an internal state variable representing the state of the material following the evolution laws of the internal state. Relation 5.8 is computed by solving the microscopic BVP as explained in Section 5.2.2.

The weak form corresponding to the system defined by equations 5.5, 5.6, 5.7, and 5.8 can be expressed by defining an admissible kinematic vector field $\mathbf{U}(\Omega)$:

$$\mathbf{U}(\Omega) = \left\{\delta \mathbf{u}_M \in \mathcal{H}(\Omega) \mid \delta \mathbf{u}_{M|\partial_D \Omega} = 0\right\} \tag{5.9}$$

With $\mathbf{H}(\Omega)$ a Hilbert space on Ω, and $\delta \mathbf{u}_M$ a test function.

Then, the corresponding weak form can be written as:

$$\int_\Omega \mathbf{P}_M(\mathbf{u}_M) : \mathbf{P}_M(\delta \mathbf{u}_M)\, d\Omega = \int_{\partial_N \Omega} \mathbf{T}_M^0 \,.\, \delta \mathbf{u}_M d\partial\Omega, \forall \delta \mathbf{u}_M \in \mathbf{U}(\Omega) \tag{5.10}$$

5.2.2 Microscopic formulation

By the separation of scales hypothesis, the characteristic length of the microscopic BVP is assumed much smaller than the characteristic length of the macroscopic loading. The micro-scale kinematics in the reference configuration of an RVE ω is defined similarly to its macroscopic counter-part 5.4:

$$\mathbf{F} = \mathbf{I} + \mathbf{u} \bigotimes \nabla_0 \tag{5.11}$$

Similarly, its continuum equilibrium equations read:

$$\begin{array}{lll} \mathbf{P}(\mathbf{x}) \,.\, \nabla_0 & = 0, & \forall \mathbf{x} \in \omega \\ \mathbf{P}(\mathbf{x}) \,.\, \mathbf{N} & = \mathbf{T}, & \forall \mathbf{x} \in \partial\omega \end{array} \tag{5.12}$$

With $\mathbf{x}$ a material point of the RVE ω in the reference configuration, and N a microscopic unit normal to the surface $\partial\omega$.

The boundary conditions are computed from the macroscopic variables and the microscopic fluctuation field $\mathbf{w}$ defined as:

$$\mathbf{w} = \mathbf{u} - (\mathbf{F}_M - \mathbf{I}) \,.\, \mathbf{x} \tag{5.13}$$

Finally, the microscopic stress-strain relationship at time t is driven by a constitutive material law which can be written, similarly to its macroscopic counter-part 5.8, as:

$$\mathbf{P}(t) = \mathcal{P}\left\{\mathbf{F}(t), \mathbf{Z}(t)\right\} \tag{5.14}$$

With $\mathbf{Z}(t)$ a history–dependent vector.

The system of equations 5.12 can be expressed in the weak form as finding $\mathbf{w} \in \mathbf{U}(\omega)$ such that:

$$\int_\omega \mathbf{P} : \left(\delta\mathbf{w} \bigotimes \nabla_0\right) d\omega = \mathbf{0}, \; \forall \delta\mathbf{w} \in \mathbf{U}(\omega) \tag{5.15}$$

With $\delta\mathbf{w}$ a test function, and $\mathbf{U}(\omega) \subset \mathcal{H}(\omega)$ a kinematic vector field or whose vectors satisfies the kinematic constraints detailed in Section 5.2.3.

5.2.3 Scale transition

In order for the FE^2 scheme to be complete, a transition is needed between the microscopic variables $\mathbf{F}$, $\mathbf{P}$ and the macroscopic variables $\mathbf{F}_M$, $\mathbf{P}_M$.

This is performed through the averaging operator $< . >= \frac{1}{\#\omega} \int_\omega . \, d\omega$:

$$F_M =< F >= \frac{1}{\#\omega} \int_\omega F \, d\omega \tag{5.16}$$

$$P_M =< P >= \frac{1}{\#\omega} \int_\omega P \, d\omega \tag{5.17}$$

With $\#\omega = \int_\omega 1 \, d\omega$ the volume of the RVE ω.

In order to ensure that the modelling is energetically consistent, the Hill-Mandel condition, or volume averaging of the virtual work, has to be fulfilled:

$$\mathbf{P}_M \, : \, \delta\mathbf{F}_M =< \mathbf{P} \, : \, \delta\mathbf{F} > \tag{5.18}$$

To guarantee that the Hill-Mandel condition is satisfied, a proper kinematic vector field $\mathbf{U}(\omega)$ has to be chosen. Using equation 5.13 in the Hill-Mandel condition 5.18, it can be obtained:

$$\mathbf{P}_M \, : \, \mathbf{F}_M = \mathbf{P}_M \, : \, \mathbf{F}_M + < P \, : \, \left(\delta\mathbf{w} \bigotimes \nabla_0 \right) > \tag{5.19}$$

Using the Gauss theorem on equation 5.13, a constraint on the fluctuation field $\mathbf{w}$ can be introduced using equation 5.16:

$$\int_{\partial\omega} \mathbf{w} \bigotimes \mathbf{N} \, d\partial\omega = \int_\omega \mathbf{w} \bigotimes \nabla_0 \, d\omega = \mathbf{0} \tag{5.20}$$

The kinematic vector field $\mathbf{U}(\omega)$ is thus defined by equation 5.20 with the consequence that all $\delta\mathbf{w} \in \mathbf{U}(\omega)$ satisfying equation 5.15 also automatically satisfy the Hill-Mandel condition 5.19.

5.2.4 Enforcing boundary conditions at the microscopic level

Neumann boundary conditions at the microscopic level can be prescribed in terms of the macroscopic stress as:

$$\mathbf{T} = \mathbf{P}_M \,.\, \mathbf{N},\ \forall \mathbf{x} \in \partial\omega. \tag{5.21}$$

Regarding Dirichlet boundary conditions, these can be expressed in terms of the macroscopic strain:

$$\mathbf{w} = \mathbf{u} - (\mathbf{F}_M - \mathbf{I})\ .\mathbf{x} = 0,\ \forall \mathbf{x} \in \partial\omega \tag{5.22}$$

Both conditions satisfy 5.19. However, condition 5.21 is known to be too compliant, while condition 5.22 is too stiff. Imposing periodic boundary conditions has been found to be able to provide better estimation [120].

In order to enforce periodicity, the fluctuation field 5.13 can be constrained by:

$$\mathbf{w}(\mathbf{x}^+) = \mathbf{w}(\mathbf{x}^-),\ \forall \mathbf{x}^- \in \partial\omega^- \text{ and matching } \mathbf{x}^+ \in \partial\omega^+ \tag{5.23}$$

With $\partial\omega^-$ and $\partial\omega^+$ being, respectively, the negative and positive part of the boundary $\partial\omega$ and partitioning it:

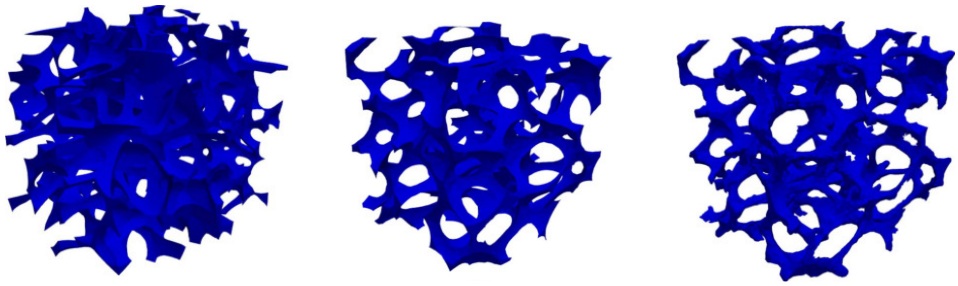
$$\partial\omega^- \cup \omega^+ = \partial\omega \tag{5.24}$$
$$\partial\omega^- \cap \omega^+ = \emptyset \tag{5.25}$$

5.2.5 Contribution

The multi-scale computational approach thus combines both advantages of the former two approaches: a scheme which is computationally moderate and which provides insights on the strains and stress undergone by the microstructure. Nonetheless, it needs as input an RVE of the microstructure. The present work in this thesis aims to provide such RVE from CT-scan images of foam samples.

In order to demonstrate the ability of the CT scan-based RVE generation to reproduce the structural properties of open foams by both the DN-CT-SCAN and the Ellipsoidal Model, finite element simulations have been conducted using the generated meshes obtained from the industrial aluminium open foam sample presented in section 3.13.1. More precisely, three meshes have been considered: two using the DN-CT-SCAN model, respectively obtained from the parent ellipsoids and associated polyhedra (see Figure s5.3a and 5.3b), and one using the Ellipsoidal Model (see Figure 5.3c). In all cases, the considered RVEs have sizes of approximatively $10 \times 10 \times 10\, mm^3$ and contain around 25 pores.

Simulations using these three RVEs were conducted using the finite element procedures proposed in [119, 122] and have been qualitatively compared against experimental measurements of an uniaxial compression reported in Reference [62].



(a) (b) (c)

FIGURE 5.3: Meshed RVEs used for the simulations and generated from: (a) DN-CT-SCAN model using ellipsoids, (b) DN-CT-SCAN model using polyhedra, (c) Ellipsoidal Model. Around $\backsim$ 100.000 nodes and $\backsim$ 70.000 tetrahedral elements were used.

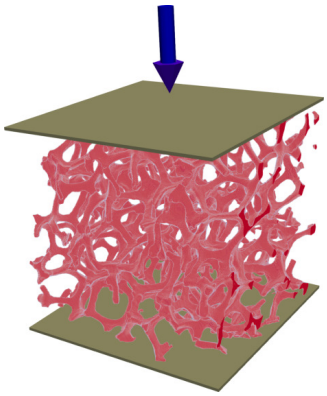


FIGURE 5.4: Experimental setup for the uniaxial compression test: top and bottom extremities molded into a small casting to ensure a stiff and stable fixture. Sides were left free.

5.3 Experimental data

The mechanical behaviour of the cubic sample of aluminium foam described in Section 3.13.1 is investigated by uniaxial compression tests using an ElectroPulsTM E10000 universal testing machine of Instron Ltd., Pfungstadt, Germany. A displacement control using a quasi-static strain rate of $5.10^{-3}\,s^{-1}$ was applied for each test. The top and bottom extremities of the studied sample were molded into a small casting to generate plane parallel plates as force transmission points for the compression tests (see Figure 5.4). The Wood's alloy guarantees a gentle molding and demolding, providing a stiff and stable fixture at the same time. These requirements cannot be adequately met by polymer resins. The boundaries orthogonal to the compression directions were left free.

TABLE 5.1: Averaged struts material properties identified for the isotropic hardening law used for the simulations.

Material property	Value	Description
E	3968.12 MPa	Young's modulus.
σ^0	46.35 MPa	Initial yield stress.
H_{iso}	214.61 MPa	Hardening modulus.
ν	0.33	Poisson's ratio.

5.4 Material properties

Simulations were carried out using for the struts a linear hardening hyperelastic-based J_2-elasto-plastic material law applied for large strains (see Appendix A in Nguyen et al. [119] for details), with the isotropic hardening law given in equation 5.26.

$$\sigma_y^0\left(\bar{\epsilon}^{pl}\right) = \sigma_0 + H_{iso}\bar{\epsilon}^{pl} \tag{5.26}$$

Where $\bar{\epsilon}^{pl}$ is the equivalent plastic strain.

Struts material properties required to parametrise the material law were identified in [62], using an inverse identification procedure based on compression tests of single pores. As the provided values tend to vary from one pore to another, averaged material properties have been considered. Table 5.1 reports the material properties used for the simulations.

5.5 Boundary conditions

In order to understand the numerical response of the RVEs, several boundary conditions were tested. This was required by the fact that the three RVEs extracted from the CT-scan data are rather small compared to the samples on which experimental measurements were conducted. Namely, for each RVE, three sets of boundary conditions were considered (see Figure 5.5): enforcement of free boundary conditions which are a simple uniaxial compression, mixed boundary conditions which are obtained by imposing a uniaxial load while constraining struts extremities to lie in common planes, and periodic boundary conditions using a 5th-order Lagrangian polynomial based interpolation [120].

More precisely, struts nodes in contact with the top plane were constrained to vertically follow its displacement, while they were free to move in the horizontal directions. Similarly, struts nodes in contact with the bottom plane were vertically constrained by it, while free to move in the horizontal directions. However, in order to avoid rigid body motion, two nodes in contact with the bottom plane were constrained as follows. The first one had also its x and y coordinates fixed, while the second one has its x coordinate fixed.

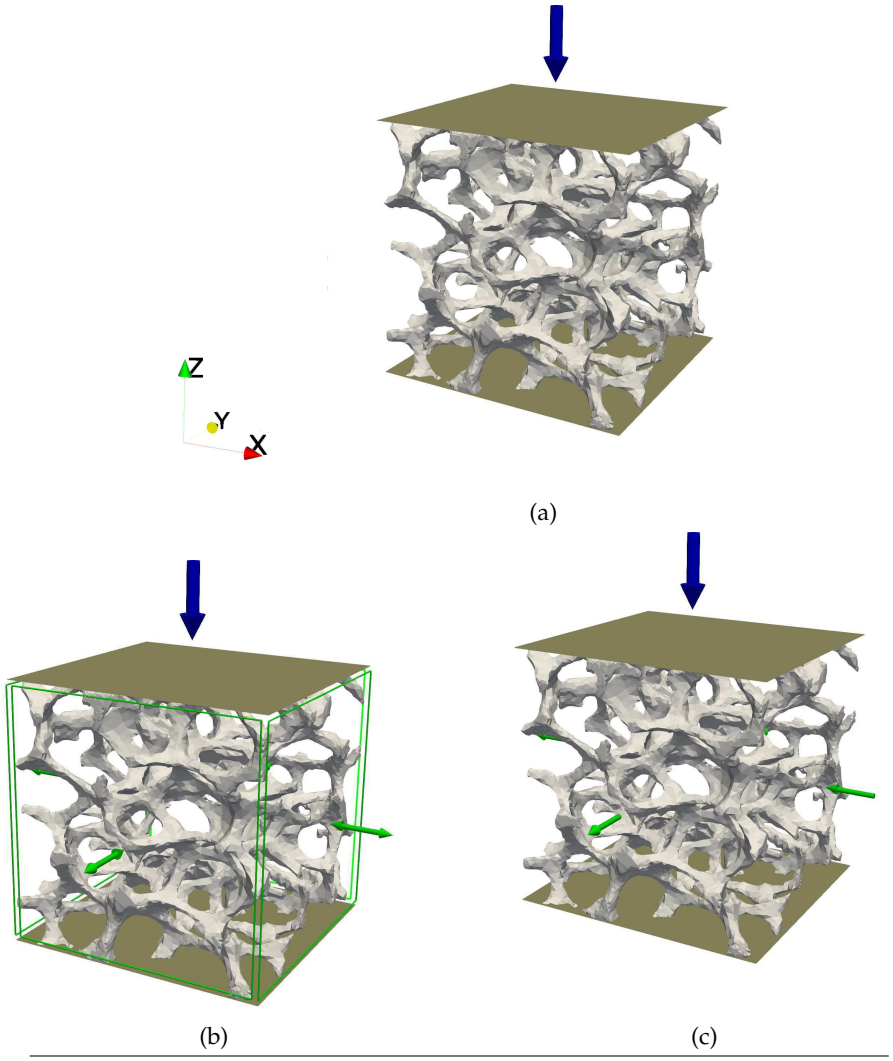


FIGURE 5.5: Tested boundary conditions for a simple uniaxial compression: (a) free boundary conditions, (b) mixed boundary conditions (uniaxial load while ensuring struts extremities lie in common planes), (c) periodic boundary conditions using Lagrangian polynomial interpolation [120].

The free boundary conditions had no other imposed conditions, and struts extremities not in contact with the planes were free to move. For the mixed boundary conditions, struts extremities located on a given lateral side were constrained to move within the same vertical plane.

Finally, for the periodic boundary conditions, displacements of struts extremities from one lateral side were constrained in terms of the displacement of the struts extremities located on the corresponding opposite side following an interpolation method as described in Reference [122] to constrain periodic boundary conditions for non-periodic meshes. For these different kinds of boundary conditions, uniaxial tension can be obtained by selecting the components of the macroscopic scale deformation gradient which are enforced during the constrained resolution of the RVE, see details in Section 3.5 of Reference [121].

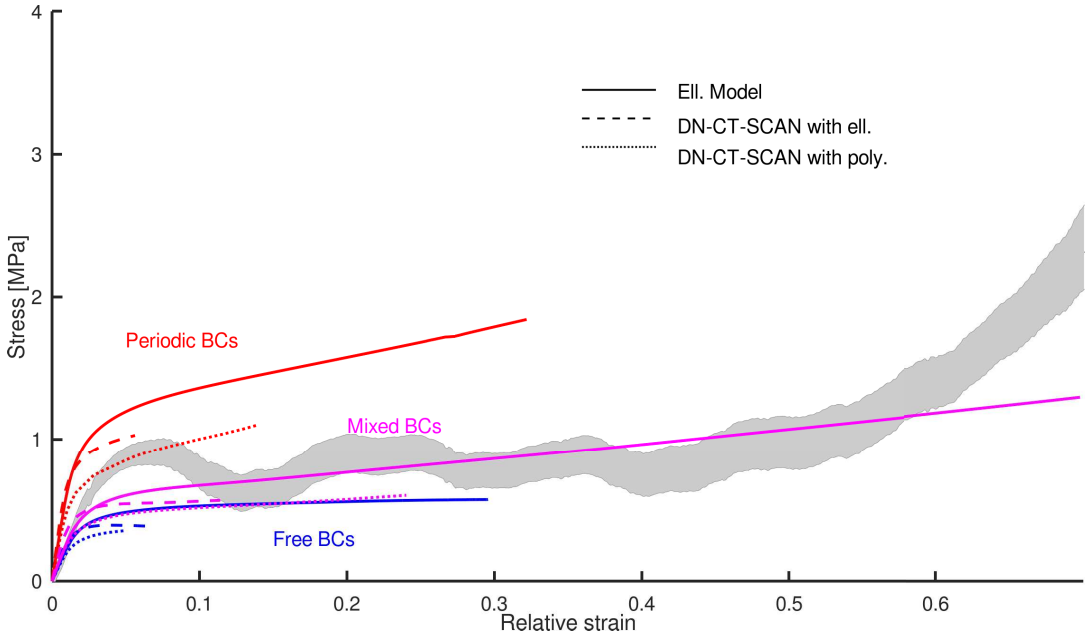


FIGURE 5.6: Simulations of a uniaxial compression test on an open cubic sample of aluminium foam (see Section 3.13.1) with periodic boundary conditions (red curves), free boundary conditions (blue curves) and mixed boundary conditions (magenta curves); using $\approx$ 70.000 tetrahedra in all the cases. Obtained (relative) strain-stress curves using the three generated RVEs presented in Figure 5.3, against experimental measurements (gray area).

5.6 Comparison of simulations against experimental data

5.6.1 Periodic boundary conditions

Simulations of uniaxial compression were conducted on the three considered RVEs showed in Figure 5.3 using periodic boundary conditions with a 5th-order Lagrangian polynomial based interpolation [120]. Figures 5.6, and 5.7 report on the red curves the (relative) strain-stress curves obtained from the simulations for each RVE along the compression direction. From this figure, some over-stiffness can be observed with respect to the experimental data in the obtained curves. As shown in [120], the obtained strain-stress curves using Lagrangian polynomial enforced periodic boundary conditions show a convergence behaviour with higher degrees of interpolation. In order to investigate if the origin of the observed over-stiffness is due to the use of a too small interpolation degree, a convergence study of the strain-stress curves with respect to the interpolation degree was conducted. From this study it has been observed that the choice of the interpolation degree cannot explain the observed discrepancies between the simulations and the experimental data (see Figure 5.8). Especially, the obtained initial slopes in the linear deformation regime are very similar for all the tested interpolation degrees and does not comply with the experimental measurements. Therefore, it has been concluded that the enforcement of periodic boundary conditions adds extra constraints on the deformation of the struts and cannot be realistic for the considered RVEs because of their reduced sizes and the non-periodicity of their geometry.

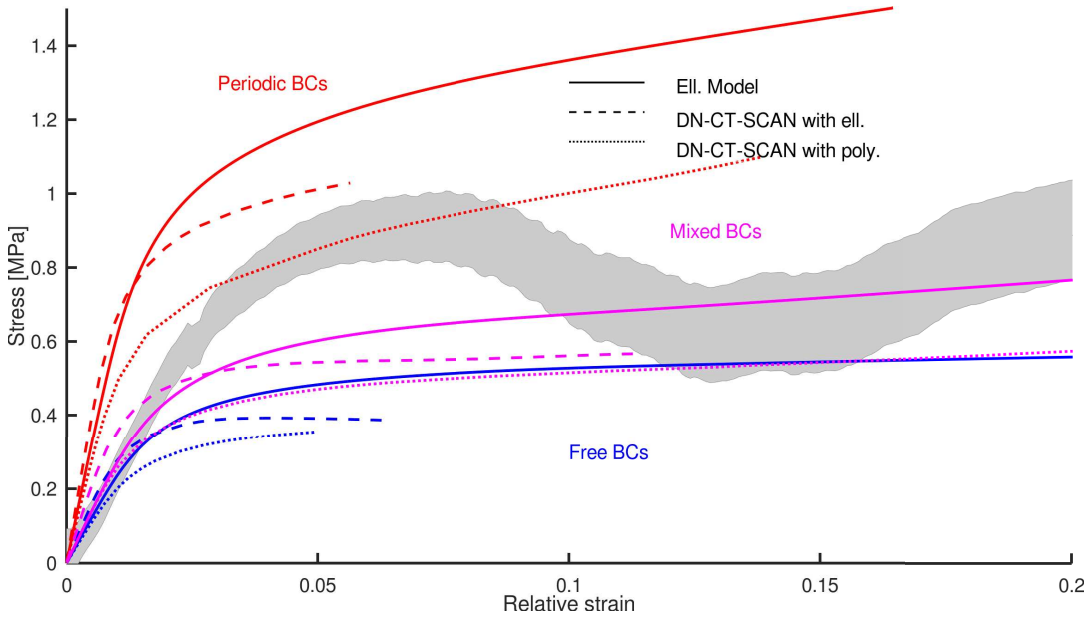


FIGURE 5.7: Zoom of Figure 5.6.

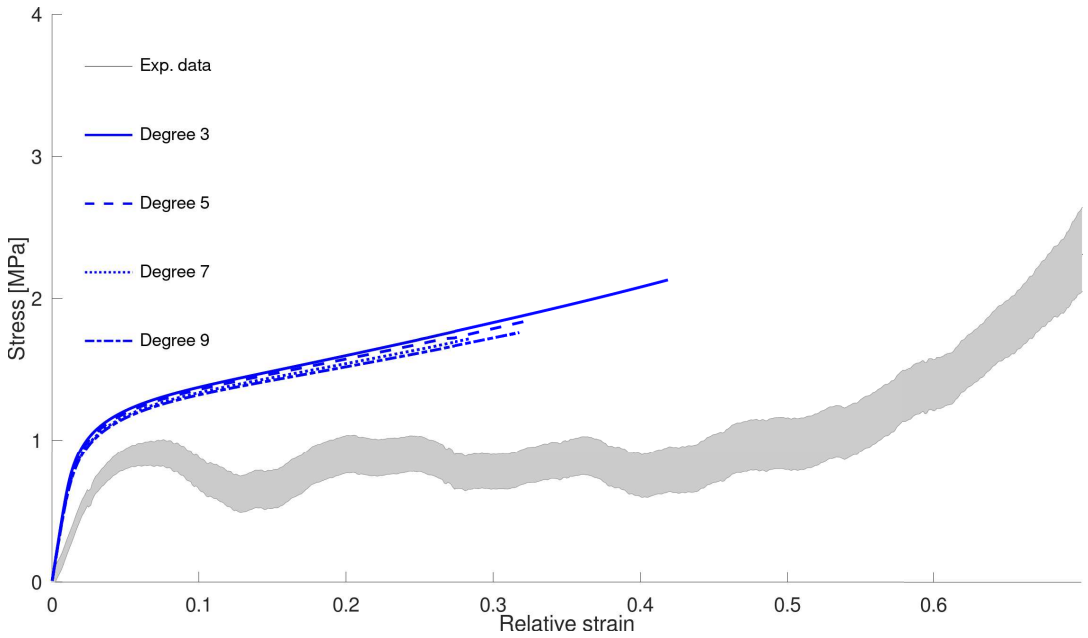


FIGURE 5.8: Simulations of a uniaxial compression test on an open cubic sample of aluminium foam (see Section 3.13.1) with periodic boundary conditions using increasing degrees of interpolation for Lagrangian polynomials at the boundaries [120].

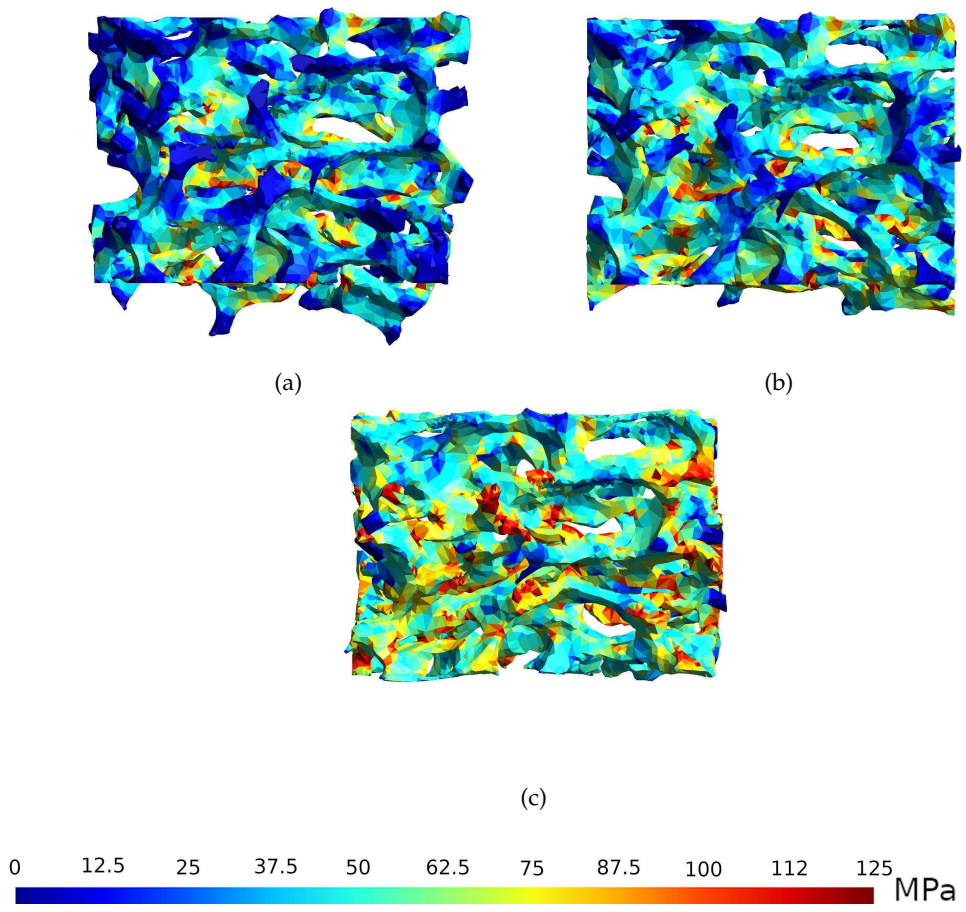


FIGURE 5.9: Simulations of uniaxial compression tests on an open cubic aluminium foam (see Section 3.13.1) using the RVE generated by the Ellipsoidal Model for different boundary conditions. (a) Free, (b) mixed and (c) periodic. The von Mises stresses are plotted on the 30 % deformed RVE.

5.6.2 Free boundary conditions

As the enforcement of periodic boundary conditions discussed in section 5.6.1 leads to an over-stiffness behaviour, one can release them and just consider the three RVEs with free lateral boundaries. Figures 5.6, and 5.7 report on the blue curves the obtained strain-stress curves for the three considered RVEs when imposing a simple uniaxial compression. It can be observed that, for this case, the result presents some over-softness with respect to experimental data.

5.6.3 Mixed boundary conditions

From the previous Sections 5.6.1 and 5.6.2, the periodic and free boundary conditions induce, respectively, over-stiffness and over-softness in the simulated behaviour of all three considered RVEs. Figure 5.7 reports on the magenta curves the strain-stress curves obtained with the considered RVEs. It can be seen that the mixed boundary conditions avoid imposing artificial constraints on the struts and that the computed strain-stress curves exhibit a comparable behaviour as observed experimentally.

TABLE 5.2: Relative differences between the computed strain–stress curves obtained at a relative strain of 20% using different meshes computed from the geometry provided from the Ellipsoidal Model. The differences were computed using a reference mesh consisting in 430.001 tetrahedra.

Boundary conditions	42.740 **tets.**	70.704 **tets.**	150.304 **tets.**
Free	11.43%	5.45%	3.3%
Mixed	10.4%	6.5%	2.9%
Periodic	13.0%	7.8%	4.1%

Indeed, the initial slope in the linear deformation regime is rather well reproduced for the polyhedra-based DN-CT-SCAN model and the Ellipsoidal Model (which present very similar geometrical features, contrary to the quite different ellipsoid-based DN-CT-SCAN model), though they display plastic deformation quite early compared to the experimental results. Moreover, as can be seen in Figure 5.6, the plateau regime is rather well reproduced by polyhedra-based DN-CT-SCAN model and the Ellipsoidal Model (magenta curves). At high relative strains, around 60 % and beyond, struts begins to pile-up against each other and a so-called densification regime is experimentally observed as a fast stress increase occurs for small deformation increments. Here, the densification regime was not reproduced by the FEM model as this one allowed the struts to interpenetrate, thus preventing any struts stacking.

It should be noted that the RVEs generated by the DN-CT-SCAN model did in general not allow to compute deformations beyond 30 %, since the presence of narrow and/or high curvature regions cause the presence of bad-shaped mesh elements, which, in turn, prevented the FEM simulations to continue due to these elements exhibiting negative Jacobians when deformed. A solution to this issue was proposed in Reference [43] by optimising meshes using the Persson-Strang analogy, and might be considered in the future for improving the meshing of the DN-CT-SCAN model.

5.6.4 Discussion: numerical convergence

Strain-stress simulations results presented in Figure 5.6 were produced using order-2 tetrahedral meshes containing approximately 70.000 tetrahedra. In order to ensure that convergence was indeed obtained, additional simulations with an increasing number of tetrahedra were conducted.

As the Ellipsoidal Model is able to provide a surface geometry, it was easy to extract several meshes with an increasing number of tetrahedra from it by using the Gmsh [54] software. Figure 5.10 shows the obtained strain-stress curves for four different meshes containing 42.740 up to 430.001 tetrahedra. It can be observed from Figure 5.10 and Table 5.2 that the differences between the strain-stress curves from the mesh using 70.704 tetrahedra to the finest mesh and for all three boundary conditions can be considered low (i.e. under 10% relative difference at a relative strain of 20%).

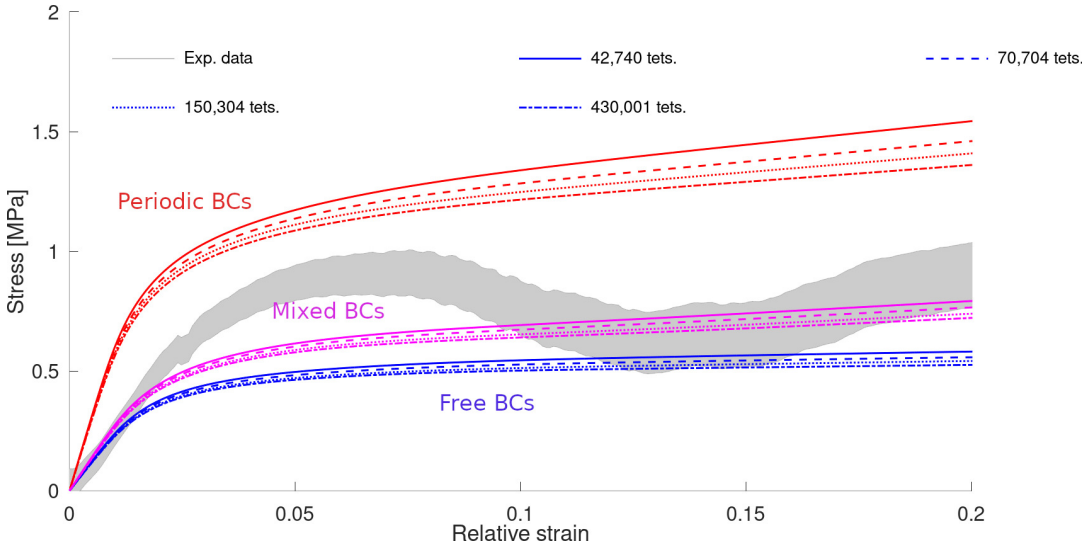


FIGURE 5.10: Simulations of a uniaxial compression test on an open cubic sample of aluminium foam (see Section 3.13.1), using the geometry provided by the Ellipsoidal Model, with the three considered boundary conditions, using order–2 meshes with different numbers of tetrahedra.

5.6.5 Discussion: the Gurson-Tvergaard-Needleman model

One may ask if the Gurson-Tvergaard-Needleman model [81] may lead to similar results as their presented here (i.e. J_2-elasto plastic material law 5.26). Indeed, despite the fact that the fracture mechanics is quite different in both models, the Gurson-Tvergaard-Needleman model with an increasing number of voids could result in a metallic foam.

In order to investigate this, simulations using the Gurson-Tvergaard-Needleman model with 93% porosity were conducted on a aluminium matrix using the same material properties as the J_2-elasto plastic model (see Table 5.1). However, the Gurson-Tvergaard-Needleman model was not able to capture the experimental strain-stress slope in the elastic region (see Figure 5.11). To the knowledge of the author of this thesis, the literature (see, e.g., Reference [44]) suggests that no suitable Gurson-Tvergaard-Needleman models for porosities above 50% are currently available.

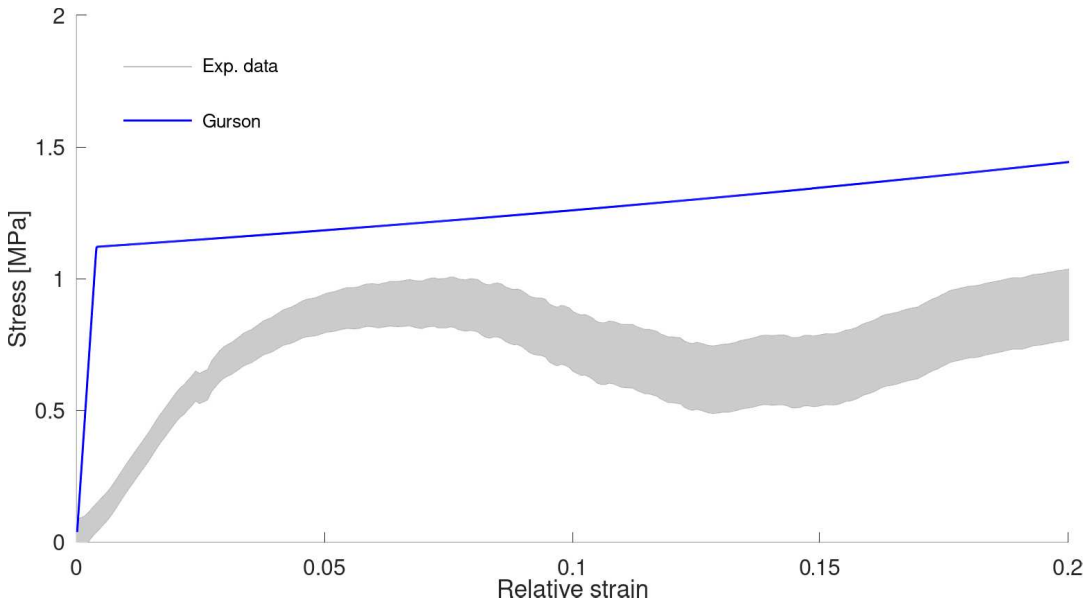


FIGURE 5.11: Simulation of an uniaxial compression test using the Gurson-Tvergaard-Needleman metal porosity model for aluminium at 93% porosity.

Chapter 6

Conclusion and perspectives

6.1 Recall of the motivation

In the introduction (chapter 1), it has been discussed how obtaining accurate numerical models of cellular materials is a challenging but necessary task in numerous engineering fields. Challenging because of the complexity of the microstructure of such materials, inducing highly complex mechanical behaviours; and necessary because of their usefulness in countless applications. Moreover, fabrication of cellular materials with specific mechanical behaviour is nowadays too dependent to costly trial-errors tests. Furthermore, ensuring constant quality during the fabrication process is also a difficult task. In this context, the ability developed this last decade to obtain precise CT-scan images of the microstructure of cellular materials, and in particular foams, as offered a precious insight for characterising the microstructures of such materials.

6.2 Contribution

This thesis aims to bridge the main two approaches for extracting suitable geometric descriptions of the microstructures of foams. These two approaches are the direct discretisation (see Section 1.2.1) and the use of idealised geometries (see Sections 1.2.2 and 1.2.3). Both these approaches have their own advantages and drawbacks. For instance, on the one hand the direct approaches bring precise geometric models of foams, but to the expense of computationally heavy models that may be difficult to handle. On the other hand, idealised approaches bring lightweight, but often imprecise, geometric models; which usually require the tuning of non–trivial parameters when handling complex (i.e. non–cell periodic) foams.

This thesis proposes an approach using the growth parent ellipsoids for identifying (possibly anisotropic) cells (see Section 2.3.9). This growth is computed using an efficient and highly parallelised algorithm provided by R. Deits et al. [39]. Over–segmentation (see Sections 2.3.9 and 2.3.10) is avoided by clustering and merging of the parent ellipsoids. This allows no to use the computationally expensive watershed and H-maxima transforms usually dedicated to this task.

Once the cells uniquely identified by a corresponding parent ellipsoids, auxiliary ellipsoids are grown in order to obtain a fit of the considered microstructure. This fit may be more or less precise depending angle increment parameter (see Section 2.3.11); allowing to obtain more or less detailed geometries of the microstructures of foams. It is worth noting that, contrary to other models, the fit is performed automatically and does not require the tuning any non–trivial parameter. The proposed model also allows to reproduce local defects such as partial or missing struts.

Then, the set of parent and auxiliary ellipsoids is discretised (see Section 4.2) and a closed surface is reconstructed using a Poisson surface reconstruction algorithm (see Section 4). The density of the discretisation can be controlled in order to get either a lightweight but less precise geometric representation, or a more precise but heavier geometric representation.

Alternatively, the set of ellipsoids (and associated polyhedra) may be used to "feed" other models as demonstrated with the DN-CT-SCAN model in Section 4.1.

Next, from the obtained geometry, it is possible to generate several meshes (from coarse to thin) according to the needs of the user. It is even possible to add user–defined geometries (e.g. using boolean operations) to the obtained geometry for, for instance, simulating the mechanical behaviour of a foam in a given environment (for example an insulating foam glued to a metallic surface).

Finally, the proposed processing steps have been implemented using the "filter" paradigm proposed by the library *Insight Toolkit* [69]. This allows more flexibility and gives the user the possibility to design and add its own "filters"/algorithms for fitting its own specific needs.

Last but not least, the proposed model offers the possibility to *stream* the data to be processed (see Section 3). This allows to handle CT–scan image data that do not fit into the RAM of the computer of the user. It is then possible to obtain the same reconstructed geometry as if there were enough RAM to process the data.

6.3 Perspectives and limitations

As for any work, here are some possible improvements for this thesis.

- The streaming part of the image analysis step can be further improved by considering blocks instead of slices in 3D CT-scan images. This should further reduce the peak memory usage for especially huge foam samples. However, it should be investigated if the efforts needed to adapt the implemented algorithms are worth the expected gain. Indeed, some algorithms are not trivially streamable by block. For instance, algorithm of Maurer et al. computing the distance transform, needs to scan for local maximum candidates along lines that extend along the whole CT-scan image in a given direction. Streaming by blocks would mean that these lines would be truncated, which imply a special treatment at their truncated extremities.
- The *Ellipsoidal Model* currently uses auxiliary ellipsoids for reconstructing the microstructure of a foam. Using the associated polyhedra to those auxiliary ellipsoids may improve the reconstruction of the microstructure, as polyhedra fit

more closely than ellipsoids the cell structures. Such a *Polyhedral Model* would use a Poisson surface reconstruction based on points and normals extracted from these polyhedra. Such an improvement was indeed noticed in terms of the Hausdorff distance when using associated polyhedra to parent ellipsoids for the *DN-CT-SCAN* model in Figure 4.6.

- When generating auxiliary ellipsoids from an associated parent ellipsoid, those are distributed evenly on the surface of the parent ellipsoid. Resolving small features in a microstructure, as illustrated in Figure 4.12, may require a large amount of auxiliary ellipsoids per parent ellipsoid. However, small features are usually localised and it is thus not useful to have a large number of auxiliary ellipsoid in regions of the microstructure where only a few is sufficient in order to obtain an acceptable fit. Generating adaptively more auxiliary ellipsoids in regions with small features and less in regions with big features can lead on some significant improvements in terms of the Hausdorff distance to the original microstructure and in terms of computational time (less auxiliary ellipsoids to be grown). A criterion for identifying where to increase the density of auxiliary ellipsoids and associated polyhedra may be the distance to these ellipsoids/polyhedra to the closest feature pixel.

- In the thesis, only parent and auxiliary ellipsoids (and associated polyhedra to parent ellipsoids) were used in both the *DN-CT-SCAN* model and the *Ellipsoid Model*. It may be interesting to investigate whether or not auxiliary ellipsoids should in their turn be used as parent for so-called *level-2* auxiliary ellipsoids (the original auxiliary ellipsoids being the *level-1* auxiliary ellipsoids). From there, one can imagine *level-3*, ..., *level-n* auxiliary ellipsoids. However, this would increase exponentially the number of ellipsoids to be grown, and, accordingly, increase exponentially the computational time for growing them. Using the above described strategy to adaptively distribute *level-n* auxiliary ellipsoids on the surface of their *level-(n-1)* parent ellipsoid, may help to mitigate this effect.

- In this thesis, auxiliary ellipsoids were seeded from there associated parent ellipsoids by using points on a given iso-surface of these parent ellipsoids. An ellipsoid iso-surface can be described through a parameter $ISO_{param} = \alpha$, with the iso-surface $S_{\mathcal{E}}$ of an ellipsoid $\mathcal{E}$ being:

$$S_{\mathcal{E}}(\alpha) = \left\{ \mathbf{x} \in \mathcal{R}^n \mid (\mathbf{x} - \mathbf{c})^t M (\mathbf{x} - \mathbf{c}) = \alpha \right\} \tag{6.1}$$

With, $\mathbf{c} \in \mathcal{R}^n$ is the centre of the ellipsoid $\mathcal{E}$, $M \in \mathcal{R}^{n \times n}$ is its shape matrix, and $\alpha > 0$ is the iso-surface parameter. When $\alpha = 1$, $S_{\mathcal{E}}(\alpha)$ is simply the surface of the ellipsoid $\mathcal{E}$.

In this thesis, α is a global parameter for all ellipsoids. Its aim is to allow the user to tune it in order to obtain a foam with a given porosity. However, α being global did not allow to precisely reached the experimental porosity of 93% for the open foam data used in the thesis while keeping a sensible reconstructed microstructure. Along with the above suggested improvements, the ability to tune this parameter α locally for each ellipsoid ,in order to reach a porosity for the reconstructed microstructure closer to the experimental one, may be worth investigating.

- The *Ellipsoidal Model* uses for the moment a standard Poisson surface reconstruction technique for generating a surface from a set of points and normals. However, this technique is known to only approximate points and unable to accurately reproduce sharp features. More advanced surface reconstruction techniques should be considered such as the *screened* Poisson surface reconstruction [78] which better approximate data points and sharp features, or reconstruction techniques explicitly detecting and labelling points located on (or near) sharp features such as described in References [146, 178, 179].

- With little adaptation[1], the proposed image analysis steps and the *Ellipsoidal Model* may be applied in other context than foam, e.g., for composite materials or in the context of crystal microstructures.

- It has been shown in this thesis that the generated ellipsoids and associated polyhedra can be used in other models than the *Ellipsoidal Model* (namely in the *DN-CT-SCAN* model). It may be worthwhile to use these ellipsoids or associated polyhedra into other models, such an improved version of the *DN-CT-SCAN* model described in Reference [43].

- Finally, it should be also possible, instead of streaming, to parallelise[2] the proposed image analysis steps among multiple cores, each with a limited amount of available memory.

[1] Such as using other ITK filters, depending on the context.
[2] The requirement for streaming a given dataset are indeed stricter than those for parallelising it.

Appendix A

Streaming

Streaming is the ability to process image by parts. This can be useful when memory is low or data to process are huge. Moreover, as free benefit, streaming allows parallel computations.

A.1 ITK's pipeline execution

The following is mostly a rewritten text taken from the ITK's documentation[1]. However, it is given here for the sake of completeness, but more importantly, this text is also given and for helping the understanding of the user. Indeed, the Ellipsoidal Model uses streaming intensively and streaming can not be properly understood without a good insight of the ITK's pipeline execution.

A.1.1 ITK's image meta-data

Inside the ITK toolbox, an image is associated to the following meta-data: the size of the image, the origin of the image, the spacing, the directions and the physical extent (see Figure A.1). Moreover, three regions are associated to each image: the *LargestPossibleRegion*, the *RequestedRegion* and the *BufferedRegion*. A region is defined by its index (starting coordinates into the image) and a size (see Figure A.2).

- The *LargestPossibleRegion* generally corresponds to the whole image, unless a filter sets it otherwise by overriding the *GenerateOutputInformation()* method.
- The *RequestedRegion* is a subregion of the *LargestPossibleRegion* allowing a filter to request only some data instead of all of them. This is for instance useful for the streaming process, when only a part of the *LargestPossibleRegion* needs to be update through the pipeline.
- The *BufferedRegion* is also a subregion of the *LargestPossibleRegion*. It sets the actual amount of data allocated by the current data object for storing (part of) the image. The *BufferedRegion* is useful if a given filter has several outputs to feed with different output *RequestedRegion* (that may overlap). This region contains the union of all output requested regions. It gives the filter the ability to provide the needed data to each output without invoking too often the upstream filters (less than the number of requested outputs).

[1] See the ITK software guide [69], pp. 195

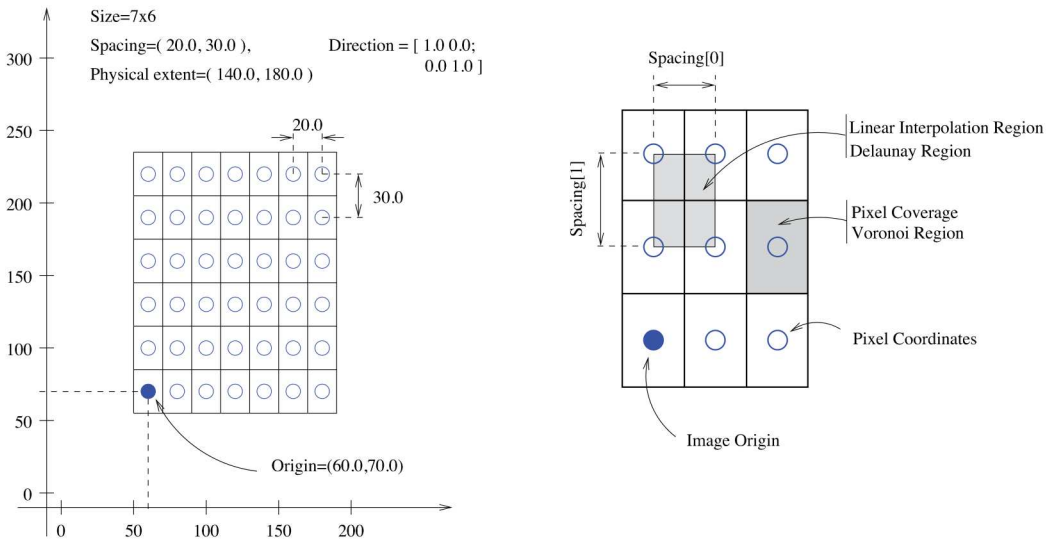


FIGURE A.1: Meta–data associated to an ITK image (scheme from [69], Figure 4.1, pp. 46).

A.1.2 ITK's pipeline execution

When a filter receive the *Update()* method invocation, the filter delegates the method to its input, invoking the *DataObject::Update()* method. In this way, the data request is propagated upstream in the pipeline (see Figure 2.5). The *DataObject::Update()* method calls itself three other methods:

1. *DataObject::UpdateOutputInformation()*.
2. *DataObject::PropagateRequetedRegion()*.
3. *DataObject::UpdateOutputData()*.

A.1.2.1 *DataObject::UpdateOutputInformation()*

This method sets a timer telling when the pipeline has been modified at the point of the corresponding calling object. This method set the image region needed at this point in the pipeline. As the image region depends on the calling object, the *UpdateOutputInformation()* method propagates upstream into the pipeline (by calling itself the *UpdateOutputInformation()* method of the upstream filter) and terminates at the source of the pipeline. During a call to the *UpdateOutputInformation()* method, a filter has the ability to set what he can provides as output to his calling object. This can be done through the overriding of the *GenerateOutputInformation()* method (which is invoked by *UpdateOutputInformation()*). The provided informations by the *GenerateOutputInformation()* method are the meta–data associated to the output image the current filter can provide. For instance, the *itk::ShrinkImageFilter* overrides this method in order to inform the downstream filter that it can only provides a maximum image size (largest possible region) which is a fraction of the original image size. If not overridden, the *GenerateOutputInformation()* method simply copy the meta–data generated by the *GenerateOutputInformation()* method of the upstream filter to the downstream filter.

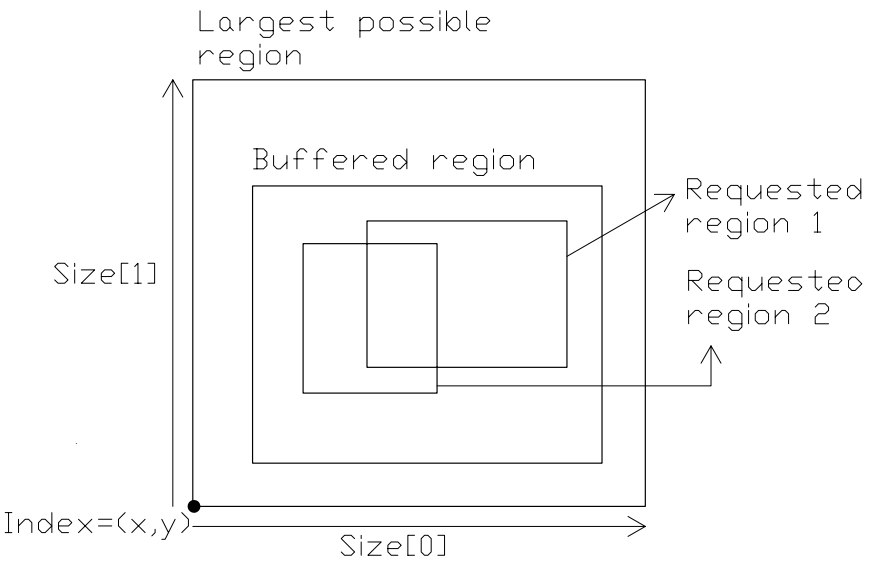


FIGURE A.2: Example of largest possible region, requested regions and buffered region.

A.1.2.2 *DataObject::PropagateRequestedRegion()*

This method propagates upstream the different data request of the filters. By default, the method requests the *LargestPossibleRegion*, but, if necessary, it can request any valid subregion of the *LargestPossibleRegion*. The *PropagateRequestedRegion()* method calls itself the following methods:

- *EnlargeOutputRequestedRegion(DataObject output)* allows the filter to tell downstream that it will provide more data than requested.
- *GenerateOutputRequestedRegion(DataObject output)* allows the filter to define separately different regions for each of its output (if more than one). By default, each output requested region are the same.
- *GenerateInputRequestedRegion()* allows the filter to inform the upstream objects that it will need a different input region than the current output requested region from the downstream filters. This is useful for filters needing a padding such as filters computing erosion or dilation.

A.1.2.3 *DataObject::UpdateOutputData()*

This is the last method called by the *Update()* method. The *UpdateOutputData()* method ensures that the current filter is up-to-date within the pipeline. If not, this methods execute the filter through a call to the *GenerateData()* method, triggering the update of all downstream filters. Filters need to be executed in the following cases:

- An instance variable of a filter has been modified.
- The input of a filter has been changed.
- The input data has been released.
- An invalid *RequestedRegion* has been set and the filter did not output data.

The *UpdateOutputData()* method of a given filter invokes recursively the *UpdateOutputData()* of the upstream filters., until a source object is met or the encountered upstream filter is determined to be up-to-date and valid. Then, the recursion is unrolled by calling the *GenerateData()* method of each filter in the recursion stack.

Appendix B

Notion of distance

B.1 Distance transforms

Definition B.1. Given a binary image I of domain D_I and a distance d, the *distance transform* $\mathcal{D}(I)$ of image I, in the sense of the associated distance d, is the image to image transformation $\mathcal{D} : I \to J, (\mathbf{x}, p) \mapsto (\mathbf{x}, [\mathcal{D}(I)](\mathbf{x}))$, with:

$$[\mathcal{D}(I)](\mathbf{x}) = min_{\mathbf{y} \in D_I} \{d(\mathbf{x}, \mathbf{y}) \mid I(\mathbf{y}) = 0\} \tag{B.1}$$

The distance transform replaces the value of each non–zero pixel (called *foreground* pixel) by the value of the distance from its closer zero–valued pixel (called *background* or *feature* pixel). Note that the distance transform of a binary image is no more a binary image.

Definition B.2. The *Euclidean distance transform* $\mathcal{ED}$ is a distance transform where the associated distance d, is the euclidean distance $d : E \times E \to \mathcal{R}^+, (u, v) \mapsto d(u, v) = d_2(u, v) = \|u - v\|_2^2 =< u - v | u - v >$.

B.2 Maurer et al. algorithm

The algorithm of Maurer et al. [111] perform the Euclidean distance tranform of a binary image. Here are the details of how it works.

B.2.1 Preliminary definitions.

Before describing the algorithm of Maurer et al., some definitions and properties are needed. The following definitions and properties are extracted from the article of Maurer et al. [111]. Figure are inspired from figures of the same article.

Definition B.3. In a binary image I, a *feature pixel* (FP) is a non–zero pixel.

Definition B.4. In a binary image I, the *closest feature pixel* (CFP) $\mathbf{y}$ of a given pixel $\mathbf{x}$ is the closest (in the sense of a given distance) non–zero pixel from $\mathbf{x}$ (which may be itself if $I(\mathbf{x}) \neq 0$). If two or more pixels are equally distant to $\mathbf{x}$ the CFP is chosen arbitrary among them.

Definition B.5. The *closest feature transform* (FT) of an image I of domain D_I is: $FT : I \to D_I \times D_I; (\mathbf{x}, p) \mapsto (\mathbf{x}, F(\mathbf{x}, p)) = (\mathbf{x}, \mathbf{y})$.

Where $\mathbf{y} = F(\mathbf{x}, p)$ is the position of the CFP of pixel p.

If the closest feature transform (FT) of an image is known, computing its distance transform (DT) is easy an can be done in linear time. Therefore, Maurer et al. algorithm focus on the computation of the FT in a linear time (with respect to the number of pixels).

Definition B.6. The *d–dimensional subimage* $I_{d,\mathbf{x}}$ of image I is the restriction of I to the subspace for which the last $n-d$ coordinates of are identical to whose of $\mathbf{x}$. Obviously, $I_{n,\mathbf{x}} = I$.

Definition B.7. F_d denotes the function F given in definition B.5 applied to the dth dimension level $d > 0$. More precisely:
$F_d : \mathbf{x} \mapsto F_d(\mathbf{x})$ is the CFP of $\mathbf{x}$ in $D_{I_{d,\mathbf{x}}}$.
Naturally, $F_n = F$.

For $d = 0$, $F_0 : \mathbf{x} \mapsto F_0(\mathbf{x})$ is defined as follow:

$$F_0(\mathbf{x}) = \begin{cases} \mathbf{x} \text{ if } I(\mathbf{x}) = 1 \\ \emptyset \text{ elsewhere} \end{cases}$$

Where $\emptyset$ denotes an undefined CFP.

Example B.1. $\forall \mathbf{x} \in D_I$:

- $F_1(\mathbf{x})$ is the CFP of $\mathbf{x}$ in $I_{1,\mathbf{x}}$, which is the image line containing $\mathbf{x}$ along the first image's direction.
- $F_2(\mathbf{x})$ is the CFP of $\mathbf{x}$ in $I_{2,\mathbf{x}}$, which is the image plane containing $\mathbf{x}$ along the first and second image's directions.
- $F_3(\mathbf{x})$ is the CFP of $\mathbf{x}$ in $I_{3,\mathbf{x}}$, which is the image hyperplane containing $\mathbf{x}$ along the first three image's directions.

Definition B.8. $X_d = \{\mathbf{x}_i\}$ is the set of pixels in D_I containing $\mathbf{x}_i$ and obtained by fixing all coordinates except the dth one which varies over the allowed image domain D_I.

Definition B.9. R_d is the continuous line running through a particular set X_d.

Definition B.10. S_d denotes the set of FP in the binary subimage $I_{d,\mathbf{x}}$.

Definition B.11. V_{S_d} denotes the Voronoï diagram which seeds are the points of S_d.

Definition B.12. $V_d^* = V_{S_d} \cap R_d$ is the intersection of the above Voronoï diagram with a line R_d.

Definition B.13. S_d^* is the set of seeds from which the Voronoï diagram V_d^* can be generated.

Definition B.14. $S'_d = \{F_{d-1}(\mathbf{x}_i)\}$ is the set of CFP in the next lower dimension for the set $X_d = \{\mathbf{x}_i\}$. Clearly, $S'_d \subseteq S_d$ (see Figure B.1).

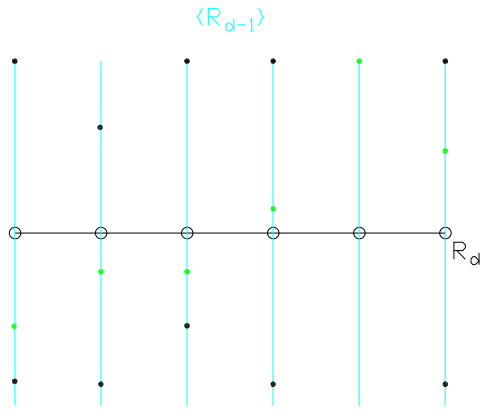


FIGURE B.1: Points of the sets S'_d (green dots) and S_d (all dots) along a particular line R_d. The open circles represents points from $X_d = \{\mathbf{x}_i\}$. The dots of S_d represent the feature points (FP). The green dots of S'_d represent the closest feature points (CFP) of the points in X_d for the next lower dimension. Example for a 2–dimensional image. Figure inspired from figure 1 of Maurer et al. [111].

B.2.2 Properties

The following propositions describe the very heart of Maurer et al. algorithm.

Proposition B.1. (Property 4 of Maurer et al. [111]). Let's $\mathbf{x}$ and $\mathbf{y}$ two n–tuples that differ only in the values of the dth coordinate (i.e., $x_i = y_i,\ i \neq d$). For concreteness, assume that $x_d < y_d$. For any $\mathbf{u}$ and $\mathbf{v}$ such that one of the following two conditions is satisfied:

1. $d(\mathbf{x}, \mathbf{u}) \leq d(\mathbf{x}, \mathbf{v})$ and $d(\mathbf{y}, \mathbf{v}) < d(\mathbf{y}, \mathbf{u})$
2. $d(\mathbf{x}, \mathbf{u}) < d(\mathbf{x}, \mathbf{v})$ and $d(\mathbf{y}, \mathbf{v}) \leq d(\mathbf{y}, \mathbf{u})$

Then: $u_d < v_d$.
Where $d : D_I \times D_I \to \mathcal{R}^+,\ (\mathbf{x}, \mathbf{y}) \mapsto d(\mathbf{x}, \mathbf{y})$ is a distance.

Proposition B.2. (Property 5 of Maurer et al. [111]). Let's $\mathbf{x}$ and $\mathbf{y}$ be two n–tuples that differ only in the values of the dth coordinate (i.e. $x_i = y_i,\ i \neq d$). Let's $\mathbf{u}$ and $\mathbf{v}$ be two n–tuples with identical values of the dth coordinates (i.e. $u_d = v_d$). If $d(\mathbf{x}, \mathbf{u}) \leq d(\mathbf{x}, \mathbf{v})$, then $d(\mathbf{y}, \mathbf{u}) \leq d(\mathbf{y}, \mathbf{v})$

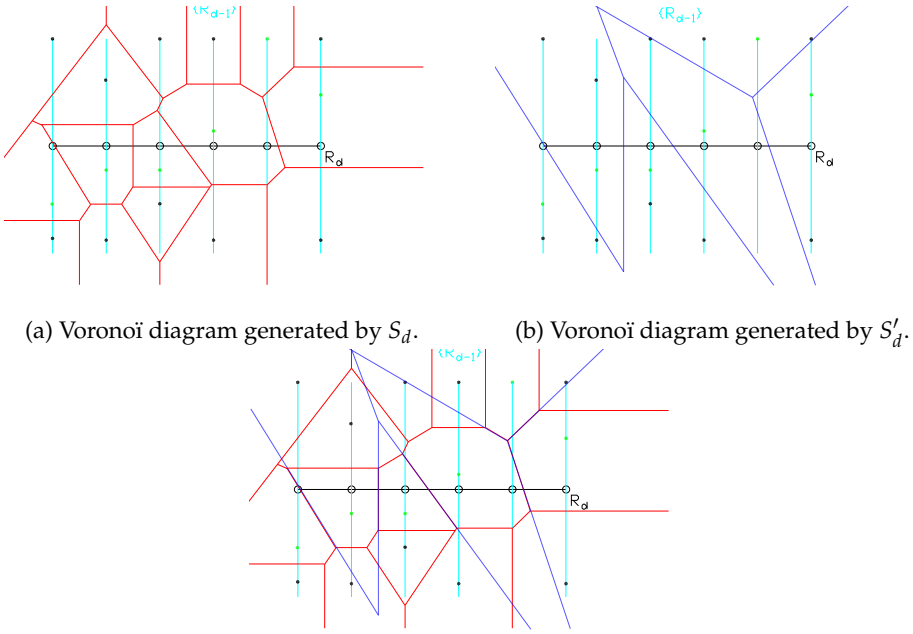
Proposition B.3. (Remark 1 of Maurer et al. [111]).

Given:

- $\mathbf{f} = F_{d-1}(\mathbf{x})$ where $\mathbf{x}$ is a pixel position on R_d. It comes that $\mathbf{f} \in S_d$ as $\mathbf{f}$ is a CFP and, thus, a FP.
- $\mathbf{g} \neq \mathbf{f}$ another FP such that $\mathbf{g} \in S_d$ and $g_d = f_d$. By the definition of F_{d-1}, $d(\mathbf{x}, \mathbf{f}) \leq d(\mathbf{x}, \mathbf{g})$.
- $\mathbf{y} \neq \mathbf{x}$ another point on R_d. By proposition B.2: $d(\mathbf{y}, \mathbf{f}) \leq d(\mathbf{y}, \mathbf{g})$. Therefore, all points belonging to R_d are at least as close to $\mathbf{f}$ than $\mathbf{g}$.

As a consequence, the Voronoï cell associated to $\mathbf{g}$ either does not intersect R_d, or the Voronoï cells of seeds $\mathbf{f}$ and $\mathbf{g}$ both intersect R_d along a common boundary. Hence, for any $\mathbf{g} \in S_d$, either $\mathbf{g} \in S'_d$ or $\mathbf{g}$ shares a same dth coordinate of a point in S'_d. mathematically speaking this translates as: $V^*_d = V_{S_d} \cap R_d = V_{S'_d} \cap R_d$.

As a result, it is possible to construct the partial Voronoï diagram V^*_d knowing only the CFP of S'_d obtained from the next lower dimension. This is the core of the idea of dimensional reduction used by the algorithm. Figure B.2 illustrates this fact with an example.



(a) Voronoï diagram generated by S_d.

(b) Voronoï diagram generated by S'_d.

(c) Superposition of the Voronoï diagrams generated by S_d and S'_d. Their intersections with R_d are identical.

FIGURE B.2: Comparison of Voronoï diagrams constructed from sets S'_d (green dots) and S_d (all dots) along a particular line R_d. The open circles represents points from $X_d = \{\mathbf{x}_i\}$. The dots of S_d represent the feature points (FP). The green dots of S'_d represent the closest feature points (CFP) of the points in X_d for the next lower dimension. It can be seen that the intersections of the two Voronoï diagrams with R^d are the same. The seeds computed in S'_d are thus sufficient for computing the CFP of points in R_d for the next upper dimension. Example for a 2–dimensional image. Figures inspired from Figure 1 of Maurer et al. [111].

Proposition B.4. (Remark 2 of Maurer et al. [111]). Let's $\mathbf{f}, \mathbf{g} \in S^*_d$ and $\mathbf{x}, \mathbf{y}$ pixel coordinates on a line R_d. By proposition B.1, if $x_d < y_d$, then $f_d < g_d$. Reciprocally, if $f_d < g_d$, then $x_d < y_d$.

Thus, for knowing to which seed in S^*_d a particular $\mathbf{x}$ on R_d is associated to, it is sufficient to sort the points in S^*_d by their d-th coordinate. Indeed, in that case, V^*_d is a set of disjoint segments. Sweeping through this set of segments in the d-th coordinate order allows to retrieve the seed in S^*_d associated to $\mathbf{x}$. Therefore, constructing explicitly V^*_d is not needed.

Proposition B.5. (Remark 3 of Maurer et al. [111]). Let's:

- $\mathbf{u}, \mathbf{v}, \mathbf{w} \in S'_d$ such that $u_d < v_d < w_d$.
- $\mathbf{x}_{\overline{\mathbf{uv}}}$ such that $d(\mathbf{u}, \mathbf{x}_{\overline{\mathbf{uv}}}) = d(\mathbf{v}, \mathbf{x}_{\overline{\mathbf{uv}}})$.
- $\mathbf{x}_{\overline{\mathbf{vw}}}$ such that $d(\mathbf{v}, \mathbf{x}_{\overline{\mathbf{vw}}}) = d(\mathbf{w}, \mathbf{x}_{\overline{\mathbf{vw}}})$.

By propositions B.1 and B.4 if $(\mathbf{x}_{\overline{\mathbf{uv}}})_d > (\mathbf{x}_{\overline{\mathbf{vw}}})_d$ then the Voronoï cell associated to $\mathbf{v}$ does not intersect R_d.

As $S^*_d \subseteq S'_d$, this proposition is useful for constructing S^*_d from S'_d by discarding points in S'_d whose associated Voronoî cells do not intersect R_d.

B.2.3 Computation of the FT

Maurer's et al. algorithm is sketched in algorithm 39. A more complete and precise version can be found in the article of Maurer et al. [111].

Algorithm 39 Maurer et al. algorithm.

Require: Binary image I of dimension n.

procedure MAURERFT(I)

▷ Initialisation

  for Traverse the pixel coordinates **x** **do**
    Compute the set $S'_1 \leftarrow \{F_0(\mathbf{x})\}$
  end for
  $S^*_1 \leftarrow S'_1$

▷ Recursion

  for $d = 1, \ldots, n$ **do**
    for Each line R_d along direction d **do**
      for Each pixel coordinate **x** along the current line **do**
        Knowing S^*_d containing the seeds of V^*_d,
        update $S'_{d+1} \leftarrow S'_{d+1} \cup \{F_d(\mathbf{x})\}$
        by querying to which Voronoi cell of V^*_d, **x** belongs.
      end for
    end for
    From S'_{d+1} construct S^*_{d+1} using proposition B.5.
  end for

end procedure

Appendix C

Mathematical morphology

The present appendix is devoted to the definition of some basic concepts of *mathematical morphology* and some popular algorithms (watershed transform, H-maxima transform, etc.) T

Definition C.1. Following P. Soille [155], "*mathematical morphology* can be defined as a theory for the analysis of spatial structures. [...] it aims at analysing the shape and forms of objects".

C.1 Grayscale reconstruction

Grayscale reconstruction is a major concept in the context of mathematical morphology, as it is at the origin of some important image to image transformations as, e.g. the h-maxima transform, h-convex transform, regional maxima and watershed transformation (see References [176] or [155] for more informations).

C.1.1 Binary reconstruction

Definition C.2. (L. Vincent [176].)
If I, J are two binary images sharing the same discrete domain D such that:

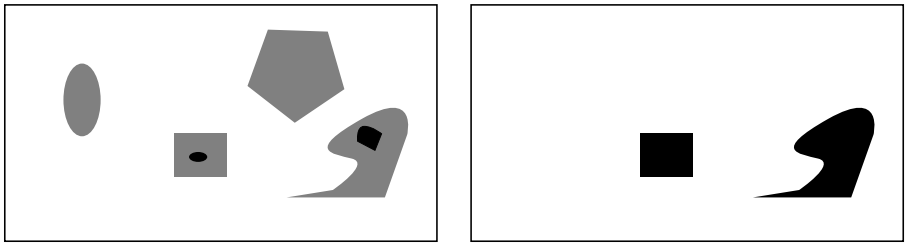
$$\forall \mathbf{x} \in D,\ J(\mathbf{x}) = 1 \Rightarrow I(\mathbf{x}) = 1$$

then, J is called the *marker*, while I is called the *mask*.

Definition C.3. (L. Vincent [176].)
Let's I, J be two binary images where I is the mask and J the marker. Given connected components $I_1, \ldots, I_k$ of I the *reconstruction* $\rho_I(J)$ is the union of the connected components of I which contain at least one pixel of J:

$$\rho_I(J) = \cup_{J \cap I_k \neq \emptyset} I_k \tag{C.1}$$

Figure C.1 illustrates a reconstruction of a binary image I from its marker J.



(a) Mask I and marker J before reconstruction.

(b) Reconstructed image.

FIGURE C.1: Binary image reconstruction of a mask I (gray objects) from a marker J (black objects).

C.1.2 Geodesic distance

Definition C.4. (P. Soille [155])
For a binary image I of domain D_I, let's $X \subset D_I$. The *geodesic distance* $d_X(\mathbf{x}, \mathbf{y})$ between two pixels $\mathbf{x}, \mathbf{y} \in X$ is the minimal length $L(P_{\mathbf{x},\mathbf{y}})$ of a path $P_{\mathbf{x},\mathbf{y}}$ joining $\mathbf{x}$ to $\mathbf{y}$ such that $P_{\mathbf{x},\mathbf{y}} \subset X$:

$$d_X(\mathbf{x}, \mathbf{y}) = min\left\{L(P_{\mathbf{x},\mathbf{y}}) \mid P_{\mathbf{x},\mathbf{y}} \subset X\right\}.$$

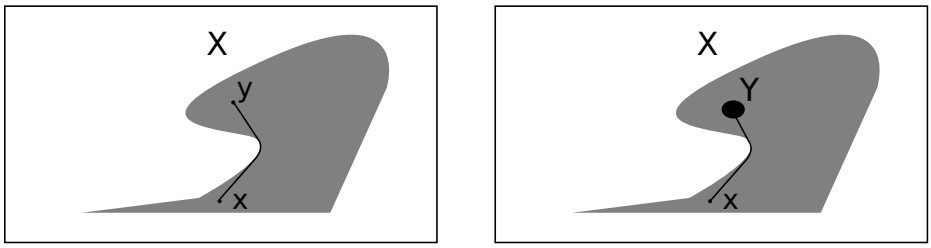
Figure C.2a illustrates the concept of geodesic distance between two pixels. Note that the geodesic distance depends on how the pixel of the image I are connected to their neighbours. Indeed the length $L(P_{\mathbf{x},\mathbf{y}})$ of a path $P_{\mathbf{x},\mathbf{y}}$ between two pixels $\mathbf{x}$ and $\mathbf{y}$ depends on the digitalisation network that allows to reach $\mathbf{y}$ from $\mathbf{x}$.

Definition C.5. (P. Soille [155])
For a binary image I of domain D_I, let's $X \subset D_I$ and $Y \subset X$. Then, the *geodesic distance* $d_X(\mathbf{x}, Y)$ between a pixel $\mathbf{x} \in X$ and Y is:

$$d_X(\mathbf{x}, Y) = min_{\mathbf{y} \in Y}\, d_X(\mathbf{x}, \mathbf{y}).$$

Where $d_X(\mathbf{x}, \mathbf{y})$ is the geodesic distance between $\mathbf{x}$ and $\mathbf{y}$.

Figure C.2b illustrates the concept of geodesic distance between a pixel and a connected set.



(a) Geodesic distance $d_X(\mathbf{x}, \mathbf{y})$ between two pixels $\mathbf{x}$ and $\mathbf{y}$ in a connected set X.

(b) Geodesic distance $d_X(\mathbf{x}, Y)$ between a pixel $\mathbf{x}$ and a connected set $\mathbf{Y}$ in a connected set X.

FIGURE C.2: Geodesic distance between a pixel and a connected set.

Definition C.6. (P. Soille [155])
Let's considerer an image I of domain D_I, a set $X = \cup_{i=1}^{k} K_i$, where the $K_i \subset A$ are disjoints connected components and $A \subset D_I$ is a larger connected component.

The *geodesic influence zone* $IZ_A(K_i)$ of a connected component K_i of X in A is the locus of points of A whose geodesic distance (see definition C.5) is smaller than any other component of X:

$$IZ_A(K_i) = \left\{ \mathbf{x} \in A \mid \forall j = 1, \ldots, k,\ j \neq i,\ d_A(\mathbf{x}, K_i) \leq d_A(\mathbf{x}, K_j) \right\}.$$

Definition C.7. (L. Vincent [176].)
Under the conditions of definition C.5, the *geodesic dilation* $\delta_X^{(n)}(Y)$ of size $n \geq 0$ of Y within X is the set of pixels in X whose distance from Y is smaller or equal to n:

$$\delta_X^{(n)}(Y) = \{ \mathbf{x} \in X \mid d_X(\mathbf{x}, Y) \leq n \}.$$

The following proposition allows to express a binary reconstruction in terms of geodesic distance.

Proposition C.1. (L. Vincent [176].)
For a binary image I of domain D_I and two sets X and Y such that $Y \subset X \subset D_I$, the reconstruction of X from Y is obtained from:

$$\rho_X(Y) = \cup_{n \geq 1} \delta_X^{(n)}(Y).$$

C.1.3 Grayscale reconstruction

Grayscale reconstruction extend the above concept of binary reconstruction to grayscale image. Basically, grayscale reconstruction is obtained from several binary reconstructions applied to the *threshold decomposition* of the considered image (see definition C.8).

Definition C.8. (L. Vincent [176].)
For a grayscale image I of domain D_I, its *threshold at level k*, is the image transformation $T_k : I \to J; I \mapsto T_k(I)$, with $k \in \{0, 1, \ldots, t_\mu\}$, defined as:

$$T_k(I) = \{ \mathbf{x} \in D_I \mid I(\mathbf{x}) \geq k \}. \tag{C.2}$$

The set of binary images $\{T_k(I)\}_{k=0,\ldots,t_m u}$ is called the *threshold decomposition* of I.

Definition C.9. (L. Vincent [176].)
Given I and J two grayscale images defined on the same domain D_I, taking their values in the discrete set $\{0, 1, \ldots, N-1\}$ and such that:

$$\forall \mathbf{x} \in D_I,\ J(\mathbf{x}) \leq I(\mathbf{x}).$$

The grayscale reconstruction $\rho_I(J)$ of I from J is given by:

$$\forall \mathbf{x} \in D_I,\ \rho_I(J)(\mathbf{x}) = max \left\{ k \in [0, N-1] \mid \mathbf{x} \in \rho_{T_k(I)}\left(T_k(J)\right) \right\}.$$

Figure C.3 sketches a grayscale reconstruction.

Definition C.10. A *queue–based reconstruction* of a grayscale image I from a grayscale image J of same domain D_I, is the grayscale reconstruction of I from J obtained from a grayscale reconstruction algorithm using a queue of pixels, such as described in [176] or [53].

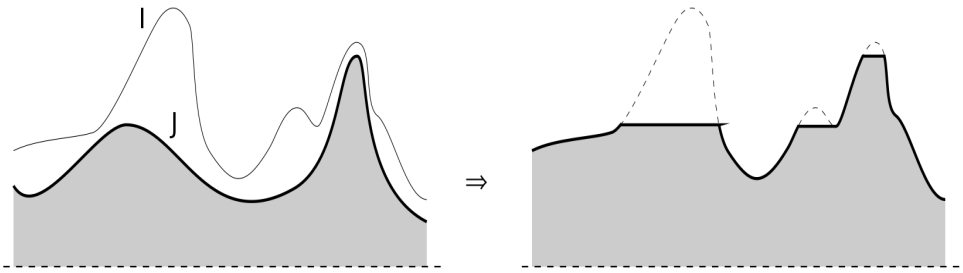


FIGURE C.3: Example of a grayscale reconstruction of a mask I from a marker J. (Image from L.Vincent [176], Figure 8).

C.2 Local maxima of an image

Obtaining local (or regional) maxima of an image is a useful morphological operation for identifying and marking image features such as cell centres or dark/bright features. For instance, these identified features may subsequently be used for marker-controlled segmentation processes, such as the marker-controlled watershed transformation (see Reference [155]).

Definition C.11. (L. Vincent [176].)
For a grayscale image I of domain D_I, $\mathbf{x} \in D_I$ is a *discrete local maximum* over a given neighbourhood $N_I(\mathbf{x})$ if:

$$\forall \mathbf{y} \in N_I(\mathbf{x}),\ I(\mathbf{y}) \leq I(\mathbf{x}). \tag{C.3}$$

Definition C.12. (L. Vincent [176].)
For a grayscale image I of domain D_I, a *regional maximum at altitude h* of I is a connected component C of $T_h(I)$ such that $C \cap T_{h+1}(I) = \emptyset$.

Definition C.13. For a discrete grayscale digitalised image I of domain D_I, a *plateau* is a connected component $C \subset D_I$ such that $\forall \mathbf{x} \in C,\ I(\mathbf{x}) = cst$.

Proposition C.2. (L. Vincent [176].)
Given a grayscale image I, the (binary) image $RMAX(I)$ of the regional maxima of I is given by:

$$RMAX(I) = I - \rho_I(I-1). \tag{C.4}$$

Where $\rho_I(I-1)$ is the grayscale reconstruction of I from $I-1$.

C.3 H-maxima/minima transformation

For a given image I, local/regional maxima mark relevant as well as irrelevant features. The H-maxima transformation acts as a filter for discarding irrelevant maxima. More precisely, the H–maxima transformation discards maxima whose depth is lower than the value h.

Definition C.14. Given a grayscale image I of domain D_I and global maximum $h_{max} = max_{\mathbf{x}\in D_I}\, I(\mathbf{x})$, its *inverse* $INV(I)$ is:

$$\forall \mathbf{x} \in D_I,\ INV(I)(\mathbf{x}) = h_{max} - I(\mathbf{x}) \tag{C.5}$$

Definition C.15. (P. Soille [155])
Given a grayscale image I, its h-maxima transformation $HMAX_h(I)$ is defined by:

$$HMAX_h(I) = \rho_I(I - h). \tag{C.6}$$

Where $\rho_I(I-h)$ is the grayscale reconstruction of I by $(I-h)$ (see definition C.9). Figure C.4 illustrates a h–maxima transform of an image.

Definition C.16. Given a grayscale image I, its h-minima transformation $HMIN_h(I)$ is defined by:

$$HMIN_h(J) = \rho_{J+h}(J),\ J = INV(I). \tag{C.7}$$

Where $\rho_{J+h}(J)$ is the grayscale reconstruction of $(J+h)$ by J (see definition C.9) and $INV(I)$ is the "inverse" of image I (see definition C.14).

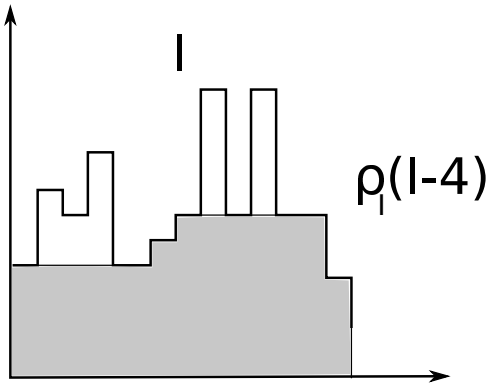


FIGURE C.4: H–maxima transformation of a 1–dimensional image using a depth $h = 4$. (Image inspired from P. Soille [155], Figure 6.16).

C.4 H-convex/concave transformation

The H-convex transformation of an image I extracts relevant regional maxima thanks to a H-maxima transform (see definition C.15).

Definition C.17. (P. Soille [155])
Given a grayscale image I, its H-convex transformation $HCONVEX_h(I)$ is defined by:

$$HCONVEX_h(I) = I - HMAX_h(I). \tag{C.8}$$

Where $HMAX_h(I)$ is the H-maxima transformation of image I (see definition C.15). Figure C.5 illustrates a H-convex transform of an image.

Definition C.18. (P. Soille [155])
Given a grayscale image I, its h–concave transformation $HCONCAVE_h(I)$ is defined by:

$$HCONCAVE_h(I) = HMIN_h(I) - I. \tag{C.9}$$

Where $HMIN_h(I)$ is the H-minima transformation of image I (see definition C.16).

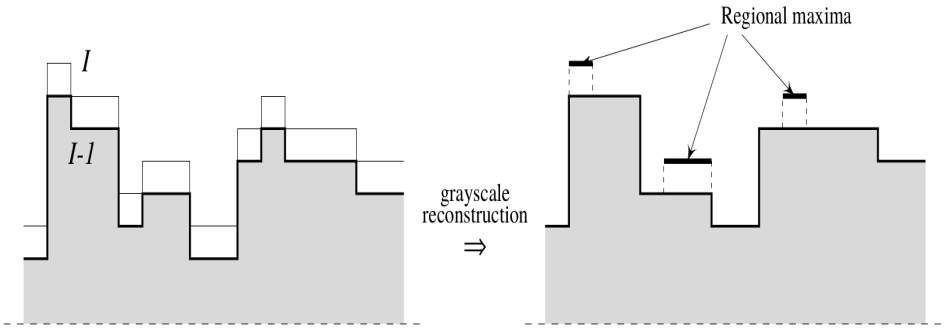


FIGURE C.5: H–convex transformation of a 1–dimensional image using a depth $h = 1$. (Image from L.Vincent [176], figure 11).

C.5 Watershed transform

Watershed transformation is a key morphological transformation for segmenting images (P.Soille [155]). In the context of cellular materials, porous materials and microscopy images, the watershed transformation is extensively used, often together with the H-maxima transform, for identifying features as cell and cell walls (e.g. in References [3, 149, 165, 33, 137, 38, 95, 169, 73, 117] to name a few).

C.5.1 Formal definition (P. Soille [155])

As suggested by the above basic idea, The watershed transformation can be defined formally in terms of flooding simulation (P. Soille [155]), where one can imagine that pinholes have been open at the regional minima. Then the topographic "landscape" represented by the grayscale image I is progressively immersed into water. The water will progressively flood the "landscape" through the pinholes, creating

catchment basins. Before defining the watershed transformation, some notation are needed:

- A grayscale image I of domain D_I is considered.
- h_{min} is the smallest image value of I: $h_{min} = min_{\mathbf{x} \in D_I}\ I(\mathbf{x})$.
- h_{max} is the largest image value of I: $h_{max} = max_{\mathbf{x} \in D_I}\ I(\mathbf{x})$.
- $CB(M)$ is the catchment basin associated to a minimum M.
- $CB_h(M)$ are the set of pixels of catchment basin $CB(M)$ which have an altitude less than or equal to h: $CB_h(M) = \{\mathbf{x} \in CB(M) \mid I(\mathbf{x}) \leq h\}$.
- $\forall \mathbf{x} \in D_I,\ \tau_t(I)(\mathbf{x}) = \begin{cases} I(\mathbf{x}) \text{ if } I(\mathbf{x}) \leq t \\ 0 \text{ elsewhere.} \end{cases}$
- $\tau_{t \leq h}(I) = \cup_{t \leq h} \tau_t(I)$.
- X_h is the subset of pixels of all catchment basins with gray value less than or equal to h: $X_h = \cup_i CB_h(M_i)$.

The catchment basins are then constructed recursively as:

$$\cdot X_{h_{min}} = RMIN_{h_{min}}(I) \tag{C.10}$$
$$\cdot \forall h \in [h_{min}, h_{max} - 1],\ X_{h+1} = RMIN_{h+1}(I) \cup IZ_{T_{h+1}(INV(I))}(X_h) \tag{C.11}$$

Where $RMIN$ is the regional minima transformations (see definition C.16), $INV(I)$ is the image "inverse" of I (see definition C.14) and IZ is the influence zone (see definition C.6), and $X_{h_{max}}$ is the set of catchment basins.

Definition C.19. (P. Soille [155])
The watershed transformation of an image I corresponds to the boundaries of the catchment basins $X_{h_{max}}$ of I.

Appendix D

Closest point of a polyhedra to an ellipsoid

This appendix describes how to solve the problem of finding the closest point belonging to a polyhedra to an ellipsoid.

Note: if the reader is not familiar with the notions of constrained convex optimisation, it is recommend to refer to the excellent book of Boyd and Vandenberghe [26] before continuing.

D.1 Problem to solve

The following convex constrained problem is considered:

$$\begin{cases} arg\ min_{\tilde{\mathbf{x}}\in\mathcal{R}^n,\mathbf{w}\in\mathcal{R}^m}\ \|\tilde{\mathbf{x}}\|^2 = \tilde{\mathbf{x}}^t\tilde{\mathbf{x}} \\ \quad\quad \sum_{k=1}^{m}\tilde{\mathbf{v}}_{j,k}w_k = \tilde{\mathbf{x}} \\ \text{s.t.}\ \ \sum_{k=1}^{m} w_k = 1 \\ \quad\quad \mathbf{w} \succeq \mathbf{0} \end{cases} \tag{D.1}$$

D.2 KKT constraint qualifications

In the problem D.1, it is possible to reduce the number of variables by injecting the first constraint in the objective function. Then problem D.1 can be expressed as:

$$\begin{cases} arg\ min_{\mathbf{w}\in\mathcal{R}^m}\ \mathbf{w}^t\tilde{V}^t\tilde{V}\mathbf{w} \\ \text{s.t.}\ \ \sum_{k=1}^{m} w_k = 1 \\ \quad\quad \mathbf{w} \succeq \mathbf{0} \end{cases} \tag{D.2}$$

Where, $\tilde{\mathbf{x}} = \tilde{V}\mathbf{w}$, and $\tilde{V} = \left(\tilde{\mathbf{v}}_{j,1},\ldots,\tilde{\mathbf{v}}_{j,m}\right) \in \mathcal{R}^{n\times m}$.

The restated problem D.2 is fo the form:

$$\begin{cases} arg\ min_{\mathbf{w}\in\mathcal{R}^m}\ \frac{1}{2}\mathbf{w}^t Q\mathbf{w} \\ \text{s.t.}\ \ G\mathbf{w} \preceq \mathbf{0} \\ \quad\quad A\mathbf{w} = b \end{cases} \tag{D.3}$$

Where, $Q = 2\tilde{V}^t\tilde{V} \in \mathcal{R}^{m\times m}$ is a symmetric definite positive matrix, $G = -I \in \mathcal{R}^{m\times m}$ (I is the identity matrix), $b = 1 \in \mathcal{R}$ and $A = (1,\ldots,1) \in \mathcal{R}^{1\times m}$.

Problems of the type of problem D.3 has been considered and solved by J. Mattingley and S. Boyd [110] in a more general form. By introducing a slack variable $\mathbf{s} \in \mathcal{R}^m$, J. Mattingley and S. Boyd transform the problem D.3 to:

$$\begin{cases} arg\ min_{\mathbf{w},\mathbf{s}\in\mathcal{R}^m}\ \frac{1}{2}\mathbf{w}^t Q\mathbf{w} \\ \quad G\mathbf{w}+\mathbf{s}=\mathbf{0} \\ \text{s.t.}\quad A\mathbf{w}=b \\ \quad \mathbf{s}\succeq\mathbf{0} \end{cases} \tag{D.4}$$

With the associated Lagrangian:

$$L:\mathcal{R}^m\times\mathcal{R}^m\times\mathcal{R}^m\times\mathcal{R}\to\mathcal{R};(\mathbf{w},\mathbf{s},\boldsymbol{\lambda},\mu)\mapsto L(\mathbf{w},\mathbf{s},\boldsymbol{\lambda},\mu)=\frac{1}{2}\mathbf{w}^tQ\mathbf{w}+\boldsymbol{\lambda}^tG\mathbf{w}+\mu(A\mathbf{w}-b) \tag{D.5}$$

Where $\boldsymbol{\lambda}$ are the dual variables associated with the inequality constraints, and μ the variable associated with the equality constraint.

And the associated KKT constraint qualifications are given by:

1. $\mathbf{s}\succeq\mathbf{0}$, $G\mathbf{w}+\mathbf{s}=\mathbf{0}$, $A\mathbf{w}=b$ (primal feasibility).
2. $\boldsymbol{\lambda}\succeq\mathbf{0}$ (dual feasibility).
3. $\boldsymbol{\lambda}^t\mathbf{s}=0$ (complementary slackness).
4. $\frac{\partial L}{\partial\mathbf{w}}=\mathbf{0}=Q\mathbf{w}+G^t\boldsymbol{\lambda}+A^t\mu$ (stationarity).

D.3 Algorithm solving the optimisation problem D.1

J. Mattingley and S. Boyd use a primal-dual iterative procedure for solving problem D.4 using the above KKT constraint qualifications. This iterative procedure requires initial values for primal and dual variables $\mathbf{w}$, $\mathbf{s}$ $\boldsymbol{\lambda}$, μ. These initial values are found by solving analytically the two problems D.6 and D.7. For details about the computations, see References [110] and [167].

$$\begin{cases} arg\ min_{\mathbf{s},\mathbf{w}\in\mathcal{R}^m}\ \frac{1}{2}\mathbf{w}Q\mathbf{w}+\frac{1}{2}\|\mathbf{s}\|^2 \\ \text{s.t.}\quad G\mathbf{w}+\mathbf{s}=\mathbf{0} \\ \quad\ A\mathbf{w}=1 \end{cases} \tag{D.6}$$

$$\begin{cases} arg\ max_{\boldsymbol{\nu},\boldsymbol{\lambda}\in\mathcal{R}^m,\mu\in\mathcal{R}}\ -\frac{1}{2}\boldsymbol{\nu}^tQ\boldsymbol{\nu}-b\mu-\frac{1}{2}\|\boldsymbol{\lambda}\|^2 \\ \text{s.t. } Q\boldsymbol{\nu}+G^t\boldsymbol{\lambda}+A^t\mu=\mathbf{0} \end{cases} \tag{D.7}$$

The procedure for solving problem D.4 is given by algorithm 41.

Algorithm 40 Initialisation algorithm for algorithm 41

Require: Convex polyhedral obstacle $\tilde{\zeta}_j$.

procedure CVXINIT($\tilde{\zeta}_j$)

Solve $\begin{pmatrix} Q & G^t & A^t \\ G & -I & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \\ \mu \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ b \end{pmatrix}$.

$\mathbf{w}^{(0)} \leftarrow \mathbf{w}$, $\mu^{(0)} \leftarrow \mu$, $\boldsymbol{\lambda} \leftarrow G\mathbf{w}$.

$\alpha_p \leftarrow inf\{\alpha \mid -\boldsymbol{\lambda} + \alpha\mathbf{1} \succeq \mathbf{0}\}$.

$\mathbf{s}^{(0)} \leftarrow \begin{cases} \boldsymbol{\lambda} \text{ if } \alpha_p < 0 \\ -\boldsymbol{\lambda} + (1+\alpha_p)\mathbf{1} \text{ elsewhere} \end{cases}$

$\alpha_d \leftarrow inf\{\alpha \mid \boldsymbol{\lambda} + \alpha\mathbf{1} \succeq \mathbf{0}\}$.

$\boldsymbol{\lambda}^{(0)} \leftarrow \begin{cases} \boldsymbol{\lambda} \text{ if } \alpha_d < 0 \\ \boldsymbol{\lambda} + (1+\alpha_d)\mathbf{1} \text{ otherwise} \end{cases}$

return $\mathbf{w}^{(0)}$, $\mathbf{s}^{(0)}$, $\boldsymbol{\lambda}^{(0)}$, $\mu^{(0)}$.

end procedure

Algorithm 41 Simplified version of the algorithm of J. Mattingley and S. Boyd ([110]). Find the closest point belonging to a convex polyhedral obstacle ζ_j from an ellipsoid $\mathcal{E}$.

Require: Ellipsoid $\mathcal{E}(E, \mathbf{c})$.
Require: Convex polyhedral obstacle ζ_j.

procedure CVX(E, $\mathbf{c}$, ζ_j)

$\tilde{\zeta}_j \leftarrow E^{-1}(\zeta_j - \mathbf{c})$ ▷ Step 0.

$\mathbf{w}^{(0)}, \mathbf{s}^{(0)}, \boldsymbol{\lambda}^{(0)}, \mu^{(0)} \leftarrow$ CVXINIT($\tilde{\zeta}_j$). ▷ Algorithm 40.

Compute affine scaling directions by solving: ▷ Step 1.

$$\begin{pmatrix} Q & 0 & G^t & A^t \\ 0 & \Lambda & S & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{w}^{aff} \\ \Delta\mathbf{s}^{aff} \\ \Delta\boldsymbol{\lambda}^{aff} \\ \Delta\mu^{aff} \end{pmatrix} = \begin{pmatrix} -\left(A^t\mu + G^t\boldsymbol{\lambda} + Q\mathbf{w}\right) \\ -S\boldsymbol{\lambda} \\ -\left(G\mathbf{w} + \mathbf{s}\right) \\ -\left(A\mathbf{w} - b\right) \end{pmatrix}$$

$$\text{Where } S = \begin{pmatrix} s_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & s_m \end{pmatrix} \text{ and } \Lambda = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_m \end{pmatrix}.$$

Compute centering–plus–correction directions by solving: ▷ Step 2.

$$\begin{pmatrix} Q & 0 & G^t & A^t \\ 0 & \Lambda & S & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{w}^{cc} \\ \Delta\mathbf{s}^{cc} \\ \Delta\boldsymbol{\lambda}^{cc} \\ \Delta\mu^{cc} \end{pmatrix} = \begin{pmatrix} 0 \\ \sigma\beta\mathbf{1} - \Delta S^{aff}\Delta\boldsymbol{\lambda}^{aff} \\ 0 \\ 0 \end{pmatrix}$$

Where

$$\Delta S^{aff} = \begin{pmatrix} \Delta s_1^{aff} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \Delta s_m^{aff} \end{pmatrix}$$

$\beta = \frac{\mathbf{s}^t\boldsymbol{\lambda}}{m}$ and $\sigma = \left(\frac{\left(\mathbf{s}+\alpha\Delta\mathbf{s}^{aff}\right)^t\left(\boldsymbol{\lambda}+\alpha\Delta\boldsymbol{\lambda}^{aff}\right)}{\mathbf{s}^t\boldsymbol{\lambda}}\right)^3$ with

$\alpha = sup\left\{\tilde{\alpha} \in [0,1] \mid \mathbf{s} + \tilde{\alpha}\Delta\mathbf{s}^{aff} \succeq \mathbf{0},\ \boldsymbol{\lambda} + \tilde{\alpha}\Delta\boldsymbol{\lambda}^{aff} \succeq \mathbf{0}\right\}$.

$\alpha \leftarrow min\left\{1, 0.99\, sup\left\{\tilde{\alpha} \geq 0 \mid \mathbf{s} + \tilde{\alpha}\Delta\mathbf{s} \succeq \mathbf{0},\ \boldsymbol{\lambda} + \tilde{\alpha}\Delta\boldsymbol{\lambda} \succeq \mathbf{0}\right\}\right\}$.

Update the primal and dual variables: ▷ Step 3.

$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(\Delta\mathbf{w}^{aff} + \Delta\mathbf{w}^{cc}\right)$.

$\mathbf{s} \leftarrow \mathbf{s} + \alpha\left(\Delta\mathbf{s}^{aff} + \Delta\mathbf{s}^{cc}\right)$.

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha\left(\Delta\boldsymbol{\lambda}^{aff} + \Delta\boldsymbol{\lambda}^{cc}\right)$.

$\mu \leftarrow \mu + \alpha\left(\Delta\mu^{aff} + \Delta\mu^{cc}\right)$.

if residual $\leq \epsilon$ **then**

$\tilde{\mathbf{x}}^* \leftarrow \sum_{k=1}^{m} \tilde{\mathbf{v}}_{j,k} w_k$. ▷ $\left\{\tilde{\mathbf{v}}_{j,k}\right\}_{k=1,\dots,m}$: vertices of $\tilde{\zeta}_j$.

return $\mathbf{x}^* = E\tilde{\mathbf{x}}^* + \mathbf{c}$.

else

Go to step 1.

end if

end procedure

Appendix E

Maximum volume ellipsoid inside a polyhedron

The algorithm of R. Deits and R. Tedrake also requires to find the maximum volume ellipsoid contained in a polyhedron. This problem has been tackle by numerous authors (see, e.g., [168] for a few references). The proposed solution here relies on the article of Y. Zhang and L. Gao ([189]). The following is merely a detailed explanation of the algorithm developed by Y. Zhang and L. Gao and taken from their own article [189].

Note: if the reader is not familiar with the notions of constrained convex optimisation, it is recommend to refer to the excellent book of Boyd and Vandenberghe [26] before continuing.

E.1 Mathematical expression of the problem.

Let's, on the one hand, $P \in \mathcal{R}^n$ be a polyhedron represented by:

$$P = P(A, \mathbf{b}) = \{\mathbf{x} \in \mathcal{R}^n \mid A\mathbf{x} \leq \mathbf{b}\} \tag{E.1}$$

Where $\mathbf{b} \in \mathcal{R}^m$ and $A \in \mathcal{R}^{m \times n}$, $m > n$ has full rank n and contains no zero rows. Moreover, it is assumed that $\exists \bar{\mathbf{x}} \in int(P)$ such that $A\bar{\mathbf{x}} < \mathbf{b}$. ($int(P)$ refers to the interior of set P).

On the other hand, let's $\mathcal{E} \subset \mathcal{R}^n$ be an ellipsoid of centre $\mathbf{c} \in \mathcal{R}^n$ with a non-singular scaling matrix $E = G^{-t}$. $\mathcal{E}$ can be represented as:

$$\mathcal{E}(\mathbf{c}, E) = \{\mathbf{x} \in \mathcal{R}^n \mid \mathbf{x} = \mathbf{c} + E\mathbf{y},\ \|\mathbf{y}\| \leq 1\} \tag{E.2}$$

Ellipsoid $\mathcal{E}(\mathbf{c}, E) \subset P$ if, and only if:

$$sup_{\|\mathbf{y}\|=1}\ \mathbf{a}_i^t(\mathbf{c} + E\mathbf{y}) \leq b_i,\ i = 1, \ldots, m \tag{E.3}$$

Where $\mathbf{a}_i$ is the ith row of A.

Equivalently, condition E.3 can be rewritten:

$$\mathbf{a}_i^t\mathbf{c} + \|E\mathbf{a}_i\| \leq b_i,\ i = 1, \ldots, m \tag{E.4}$$

Let's introduce the notation:

$$\mathbf{h}(E) = (\|E\mathbf{a}_1\|, \dots, \|E\mathbf{a}_m\|)^t \in \mathcal{R}^m \tag{E.5}$$

Then:

$$\mathcal{E}(\mathbf{c}, E) \subset P \Leftrightarrow \mathbf{b} - A\mathbf{c} - \mathbf{h}(E) \succeq 0 \tag{E.6}$$

The volume $Vol(\mathcal{E})$ of ellipsoid $\mathcal{E}$ is given by:

$$Vol(\mathcal{E}) = V_0 \prod_{i=1}^{n} r_i \propto \prod_{i=1}^{n} r_i \tag{E.7}$$

Where $V_0 = \frac{2\pi^{n/2}}{n\Gamma\left(\frac{1}{2}n\right)}$ is the volume of the unit sphere in $\mathcal{R}^n$[1] and the $\{r_i\}_{i=1,\dots,n}$ are the semi-axis lengths of the ellipsoid.

The semi-axis lengths $\{r_i\}_{i=1,\dots,n}$ are related to the singular values $\{\mu_i\}_{i=1,\dots,n}$ of the scaling matrix E as follow:

$$r_i = \mu_i,\ i = 1, \dots, n$$

Therefore:

$$\frac{1}{Vol(\mathcal{E})} \propto det\left(E^{-1}\right) = (det(E))^{-1} \tag{E.8}$$

Maximising the ellipsoid volume $Vol(\mathcal{E})$ is thus equivalent to minimising

$$ln\left((det(E))^{-1}\right) = -ln\ det(E) \tag{E.9}$$

as the logarithmic function ln is bijective over $\mathcal{R}_+ \backslash \{0\}$.

Finding the maximum volume ellipsoid $\mathcal{E}(\mathbf{c}, E) \subset P$ amounts to solve the following optimisation problem:

$$\begin{cases} min_{\mathbf{c},E} & -ln\ det(E) \\ \text{s.t.} & \mathbf{b} - A\mathbf{c} - \mathbf{h}(E) \succeq 0 \\ & E \succ 0 \end{cases} \tag{E.10}$$

The Lagrangian and dual function associated to the optimisation problem E.10 are respectively:

$$L(\mathbf{c}, E, \boldsymbol{\lambda}) = -ln\ det(E) - \boldsymbol{\lambda}^t (\mathbf{b} - A\mathbf{c} - \mathbf{h}(E))$$
$$g(\boldsymbol{\lambda}) = inf_{\mathbf{c},E}\ L(\mathbf{c}, E, \boldsymbol{\lambda})$$

[1] $\Gamma(m) = \int_0^\infty e^{-r^2} r^{2m-1}\, dr$ is the gamma function.

E.1.1 KKT constraint qualifications

The optimisation problem E.10 is solved using the Karush-Kuhn-Tucker (KKT) constraint qualifications (see Section 5.5.3 of the book [26]), where the following two differentiation formulae are used[2]:

$$\nabla_E \left(ln\ det(E) \right) = E^{-1} \tag{E.11}$$

$$\nabla_E h_i(E) = \frac{E\mathbf{a}_i\mathbf{a}_i^t + \mathbf{a}_i\mathbf{a}_i^t E}{2h_i(E)},\ i = 1, \ldots, m \tag{E.12}$$

With these differentiation formulae and introducing the notations:

$$\Lambda = \begin{pmatrix} \lambda_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_m \end{pmatrix} \in \mathcal{R}^{m \times m} \tag{E.13}$$

$$Y = Y(E, \boldsymbol{\lambda}) = \begin{pmatrix} \frac{1}{h_1(E)} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \frac{1}{h_m(E)} \end{pmatrix} \Lambda \in \mathcal{R}^{m \times m} \tag{E.14}$$

$$\mathbf{y} = \left(\frac{\lambda_1}{h_1(E)}, \ldots, \frac{\lambda_m}{h_m(E)} \right)^t \in \mathcal{R}^m \tag{E.15}$$

The KKT constraint qualifications are (Y. Zhang and L. Gao [189]):

$$A^t\boldsymbol{\lambda} = \mathbf{0} \tag{E.16}$$

$$E^{-1} - \left(E\left(A^tYA\right) + \left(A^tYA\right)E \right)/2 = 0 \tag{E.17}$$

$$\mathbf{z} - (\mathbf{b} - A\mathbf{c} - \mathbf{h}(E)) = \mathbf{0} \tag{E.18}$$

$$\Lambda\mathbf{z} = \mathbf{0} \tag{E.19}$$

$$\boldsymbol{\lambda}, \mathbf{z} \succeq 0 \tag{E.20}$$

Where $\mathbf{z} \in \mathcal{R}^m$ is a slack variable.

Indeed

- Condition E.16 comes from the KKT stationarity condition with respect to variable $\mathbf{c}$:
$$\nabla_{\mathbf{c}} L = \mathbf{0} = A^t\boldsymbol{\lambda}$$
- Condition E.17 comes from the KKT stationarity condition with respect to variable E and by making use of the differentiation formulae E.11 and E.12:
$$\begin{aligned} \nabla_E L \quad &= 0 = -E^{-1} + \textstyle\sum_{i=1}^n \lambda_i \left(\frac{E\mathbf{a}_i\mathbf{a}_i^t + \mathbf{a}_i\mathbf{a}_i^t E}{2h_i(E)} \right) \\ \Leftrightarrow \quad &0 = E^{-1} - \left(E\left(A^tYA\right) + \left(A^tYA\right)E \right)/2 \end{aligned}$$

[2] See Reference [168], for details about how these formulae are derived.

- For conditions E.18 and E.20, the constraint $\mathbf{b} - A\mathbf{c} - \mathbf{h}(E) \succeq 0$ is considered. By introducing the slack variable $\mathbf{z} \in \mathcal{R}^m$ such that $\mathbf{b} - A\mathbf{c} - \mathbf{h}(E) = \mathbf{z} \succeq 0$, this constraint can be rewritten as:

  $$\mathbf{z} - (\mathbf{b} - A\mathbf{c} - \mathbf{h}(E)) = \mathbf{0}$$

  Which is condition E.18.

  And, with the KKT dual feasibility condition $\boldsymbol{\lambda} \succeq 0$, one obtain:

  $$\boldsymbol{\lambda}, \mathbf{z} \succeq 0$$

  Which is condition E.20.

- Finally, the KKT complementary slackness condition $\boldsymbol{\lambda}^t (\mathbf{b} - A\mathbf{c} - \mathbf{h}(E)) = \mathbf{0}$ can be expressed as

  $$\Lambda \mathbf{z} = \mathbf{0}$$

  Which is condition E.19.

E.1.2 Simplification of the KKT constraint qualifications

The above KKT conditions can be further simplified by eliminating the matrix variable E by solving equation E.17. The solution[3] to equation E.17 is indeed[4]:

$$E(\mathbf{y}) = \left(A^t Y A\right)^{-1/2} \tag{E.21}$$

As, by definition, $\mathbf{h}(E) = (\|E\mathbf{a}_1\|, \ldots, \|E\mathbf{a}_m\|)^t$; with solution E.21 $\mathbf{h}(E)$ can be re-expressed as:

$$\tilde{\mathbf{h}}(\mathbf{y}) = \mathbf{h}\left(E(\mathbf{y})\right) \tag{E.22}$$

Moreover, by equation E.14 one can obtain:

$$\boldsymbol{\lambda} = \mathbf{g}(\mathbf{y}) = Y\tilde{\mathbf{h}}(\mathbf{y}) \tag{E.23}$$

Using the above results, the KKT constraint qualifications can then be expressed as:

$$A^t \boldsymbol{\lambda} = \mathbf{0} \tag{E.24}$$
$$\mathbf{z} - \mathbf{b} + A\mathbf{c} + \tilde{\mathbf{h}}(\mathbf{y}) = \mathbf{0} \tag{E.25}$$
$$\Lambda \mathbf{z} = \mathbf{0} \tag{E.26}$$
$$\mathbf{y}, \boldsymbol{\lambda}, \mathbf{z} \succeq \mathbf{0} \tag{E.27}$$
$$\boldsymbol{\lambda} - \mathbf{g}(\mathbf{y}) = \mathbf{0} \tag{E.28}$$

[3] The uniqueness of the solution is proven in Reference [189].
[4] This is easy the check by plugging the solution into the equation.

Or, in a more condensed form (Y. Zhang and L. Gao [189]):

$$\mathbf{F}_0(\mathbf{c},\mathbf{y},\boldsymbol{\lambda},\mathbf{z}) = \mathbf{0},\ \mathbf{y},\boldsymbol{\lambda},\mathbf{z} \succeq \mathbf{0} \tag{E.29}$$

$$\mathbf{F}_0 : \mathcal{R}^{n+3m} \to \mathcal{R}^{n+3m}; (\mathbf{c},\mathbf{y},\boldsymbol{\lambda},\mathbf{z}) \mapsto \mathbf{F}_0(\mathbf{c},\mathbf{y},\boldsymbol{\lambda},\mathbf{z}) = \begin{pmatrix} A^t\boldsymbol{\lambda} \\ A\mathbf{c} + \tilde{\mathbf{h}}(\mathbf{y}) + \mathbf{z} - \mathbf{b} \\ \boldsymbol{\lambda} - \mathbf{g}(\mathbf{y}) \\ \Lambda\mathbf{z} \end{pmatrix} \tag{E.30}$$

The system E.29 can be further simplified by eliminating the variable $\boldsymbol{\lambda}$ by using equation E.23 (Y. Zhang and L. Gao [189]):

$$\mathbf{F}_1(\mathbf{c},\mathbf{y},\mathbf{z}) = \mathbf{0},\ \mathbf{y},\mathbf{z} \succeq \mathbf{0} \tag{E.31}$$

$$\mathbf{F}_1 : \mathcal{R}^{n+2m} \to \mathcal{R}^{n+2m}; (\mathbf{c},\mathbf{y},\mathbf{z}) \mapsto \mathbf{F}_1(\mathbf{c},\mathbf{y},\mathbf{z}) = \begin{pmatrix} A^t\mathbf{g}(\mathbf{y}) \\ A\mathbf{c} + \tilde{\mathbf{h}}(\mathbf{y}) + \mathbf{z} - \mathbf{b} \\ Z\mathbf{g}(\mathbf{y}) \end{pmatrix} \tag{E.32}$$

And:

$$Z = \begin{pmatrix} z_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & z_m \end{pmatrix} \in \mathcal{R}^{m\times m} \tag{E.33}$$

E.1.3 Algorithm solving the optimisation problem E.10

The algorithm of Y. Zhang and L. Gao (see algorithm 42[5]) can be viewed as a damped Newton's method applied to a perturbed version of the system E.31. The perturbed version is the following:

$$\mathbf{F}_1(\mathbf{c},\mathbf{y},\mathbf{z}) = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w} \end{pmatrix},\ \mathbf{y},\mathbf{z} \succ \mathbf{0} \tag{E.34}$$

Where $\mathbf{w} = \mu\mathbf{w}^0$ for some $\mathbf{w}^0 \in \mathcal{R}^m_{++}$. Usually, one chooses $\mathbf{w}^0 = \mathbf{e}$, with $\mathbf{e}$ a vector of all ones.

The perturbed system E.34 admits a unique solution for every $\mu > 0$ and, as $\mu \to 0$, its solution converges to the solution of the original system E.31 (see Reference [189] for a proof).

[5] A Matlab implementation can be found at http://www.caam.rice.edu/~zhang/mve/index.html.

Algorithm 42 Algorithm of Y. Zhang and L. Gao [189]

Require: Bounded non–empty polyhedron $P = P(A, \mathbf{b})$.
Require: $\mathbf{c}^0 \in int(P)$.

procedure MAXVOLELLIPSOIDINPOLYHEDRON(A, $\mathbf{b}$, $\mathbf{c}^0$)
  $k \leftarrow 0$.
  $\mathbf{y}^0 \leftarrow (1, \ldots, 1)^t$.
  $E(\mathbf{y}^0) \leftarrow \left(A^t Y^0 A\right)^{-1/2}$.
  $\tilde{\mathbf{h}}(\mathbf{y}^0) \leftarrow \left(\|E(\mathbf{y}^0)\mathbf{a}_1\|, \ldots, \|E(\mathbf{y}^0)\mathbf{a}_m\|\right)^t$.
  $\mathbf{z}^0 \leftarrow \mathbf{b} - A\mathbf{c}^0 - \tilde{\mathbf{h}}(\mathbf{y}^0)$.
  ▷ **Step 1**
  Choose $\sigma_k \in]0, 1[$.
  $\mu_k \leftarrow \sigma_k \frac{\mathbf{g}(\mathbf{y}^k)^t \mathbf{z}^k}{m}$.
  ▷ **Step 2**
  Solve for $(d\mathbf{c}, d\mathbf{y}, d\mathbf{z})$ from:
  $$F_1'(\mathbf{c}^k, \mathbf{y}^k, \mathbf{z}^k) \begin{pmatrix} d\mathbf{c} \\ d\mathbf{y} \\ d\mathbf{z} \end{pmatrix} = \mu_k \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{e} \end{pmatrix} - \mathbf{F}_1(\mathbf{c}^k, \mathbf{y}^k, \mathbf{z}^k).$$
  ▷ **Step 3**
  Choose a step length $\alpha_k \in]0, 1]$ and update:
  $\left(\mathbf{c}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}\right) \leftarrow \left(\mathbf{c}^k, \mathbf{y}^k, \mathbf{z}^k\right) + \alpha_k \left(d\mathbf{c}, d\mathbf{y}, d\mathbf{z}\right)$ such that $\mathbf{c}^{k+1} \in P$ and $\mathbf{y}^{k+1}, \mathbf{z}^{k+1} \succeq \mathbf{0}$.
  ▷ **Step 4**
  if $\|\mathbf{F}_1\left(\mathbf{c}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}\right)\| \leq \epsilon$ **then**
    $\mathbf{c}^* \leftarrow \mathbf{c}^{k+1}$.
    $E^* \leftarrow E(\mathbf{y}^{k+1}) = \left(A^t Y^{k+1} A\right)^{-1/2}$.
    return $\mathbf{c}^*$, E^*.
  else
    $k \leftarrow k + 1$ and goto Step 1.
  end if
end procedure

In algorithm 42, the following quantities are used:

$$F_1'(\mathbf{c},\mathbf{y},\mathbf{z}) = \begin{pmatrix} 0 & A^t g'(\mathbf{y}) & 0 \\ A & \tilde{h}'(\mathbf{y}) & I \\ 0 & Zg'(\mathbf{y}) & G(\mathbf{y}) \end{pmatrix} \tag{E.35}$$

$$G(\mathbf{y}) = \begin{pmatrix} g_1(\mathbf{y}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & g_m(\mathbf{y}) \end{pmatrix} \tag{E.36}$$

and $g'(\mathbf{y})$ and $\tilde{h}'(\mathbf{y})$ are the Jacobian matrices of $\mathbf{g}(\mathbf{y})$ and $\tilde{\mathbf{h}}(\mathbf{y})$, respectively.

For the sake of completeness, these Jacobian matrices are given by (Y. Zhang and L. Gao [189]):

$$g'(\mathbf{y}) = H(\mathbf{y}) + Yh'(\mathbf{y}) \tag{E.37}$$

And:

$$h'(\mathbf{y}) = -\frac{1}{2}H(\mathbf{y})^{-1}\left(Q(\mathbf{y}) \circ Q(\mathbf{y})\right) \tag{E.38}$$

Where:

$$Q(\mathbf{y}) = A\left(A^tYA\right)^{-1}A^t \tag{E.39}$$

With "$\circ$" referring to the Hadamard product between matrices.

And:

$$H(\mathbf{y}) = \begin{pmatrix} \tilde{h}_1(\mathbf{y}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{h}_m(\mathbf{y}) \end{pmatrix} \tag{E.40}$$

Appendix F

The *GDBSCAN* algorithm

F.1 Preliminary definitions

Let's D be a spatial database of objects (here ellipsoids) with spatial attributes (here the semi-axes of the ellipsoids and their associated lengths) and non-spatial attributes (here the value of the maximum associated to each ellipsoid). It is assumed here that:

1. The objects of D lie in an affine euclidean space E: $\forall o \in D, o \subset E$.
2. D has finite cardinality ($\#D < \infty$).
3. D is bounded (for any point O in the space E containing the objects of D, there exist a radius $R > 0$ such that, for a given distance d, the closed ball[1] centred in O of radius R contains all objects of D: $\forall o \in D$, $o \subset B[O;R]$).

Definition F.1. (From http://mathworld.wolfram.com/PowerSet.html) Given a set D, its *powerset*, noted 2^D or $\mathcal{P}(D)$, is the set of all subsets of D. For D of cardinality $\#D$, the cardinality of its powerset 2^D is $\#\left(2^D\right) = 2^{\#D}$.

Definition F.2. (J. Sander et al. [147])
Let's $NPred$ be a binary predicate on D which is reflexive and symmetric, i.e., $\forall p, q \in D : \ NPred(p,p)$ and, if $NPred(p,q)$ then $NPred(q,p)$. Then the *NPred-neighbourhood* of an object $o \in D$ is defined as:

$$N_{NPred}(o) = \left\{o' \in D \mid NPred(o,o')\right\} \tag{F.1}$$

Definition F.3. (J. Sander et al. [147])
Let's $wCard$ be a function from the powerset of the database D (see definition F.1) into the nonnegative Real numbers, $wCard : 2^D \to \mathcal{R}^+$ and $MinCard$ be a positive real number. Then, the predicate $MinWeight$ for a set S of objects is defined to be true if and only if $wCard \geq MinCard$.

Definition F.4. (J. Sander et al. [147])
An object p is *directly density–reachable* from an object q with respect to $NPred$ and $MinWeight$ if:

1. $p \in N_{NPred}(q)$.
2. $MinWeight\left(N_{NPred}(q)\right) = true$ (core object condition).

Figure F.1 illustrates a situation where an object p is directly density reachable from an object q, but where the converse is not true.

[1] For a space E, a closed ball $B[O;R]$ of radius $R > 0$ centred in $O \in E$ is defined as: $B[O;R] = \{y \in E \mid d(o,y) \leq R\}$.

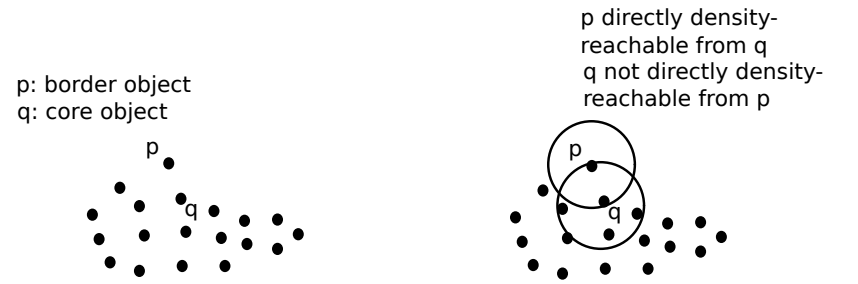


FIGURE F.1: Core objects and border objects. [147].

Definition F.5. (J. Sander et al. [147])
An object p is *density-reachable* from an object q with respect to $NPred$ and $MinWeight$ if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that $\forall i = 1, \ldots, n-1$, p_{i+1} is directly density reachable from p_i with respect to $NPred$ and $MinWeight$.

The left part of Figure F.2 illustrates this concept.

Definition F.6. (J. Sander et al. [147])
An object p is *density-connected* to an object q with respect to $NPred$ and $MinWeight$ if there is an object o such that both, p and q are density-reachable from o with respect to $NPred$ and $MinWeight$.

The right part of Figure F.2 illustrates this concept.

FIGURE F.2: Density–reachability and density–connectivity [147].

Definition F.7. (J. Sander et al. [147])
A *density-connected set* C with respect to $NPred$ and $MinWeight$ in D is nonempty subset of D satisfying the following conditions:

1. Maximality: $\forall p, q \in D$, if $p \in C$ and q is density-reachable from p with respect to $NPred$ and $MinWeight$, then $q \in C$.
2. Connectivity: $\forall p, q \in C$: p is density-connected to q with respect to $NPred$ and $MinWeight$.

Definition F.8. (J. Sander et al. [147])
A *clustering* CL of D with respect to $NPred$ and $MinWeight$ is a set of density-connected sets with respect to $NPred$ and $MinWeight$ in D, $CL = \{C_1, \ldots, C_k\}$, such that for all C the following hold: if C is a density-connected set with respect to $NPred$ and $MinWeight$ in D, then $C \in CL$.

Definition F.9. (J. Sander et al. [147])
Let $CL = \{C_1, \ldots, C_k\}$ be a clustering of the database D with respect to $NPred$ and $MinWeight$. The the *noise* of D is defined here as the set of objects in the database D not belonging to any density–connected set C_i, i.e., $noise_{CL} = D \setminus (C_1 \cup \ldots \cup C_k)$.

F.2 Algorithm

The GDBSCAN algorithm starts with a set of objects marked as unclassified. It takes then the first unclassified object p of the set and marks it with a cluster number (e.g. starting from 1). Then, it tries to add to this first cluster candidate and mark all density connected objects $p_1, \ldots, p_n$ with respect to $NPred$ and $MinWeight$ by scanning the neighbourhood of the first object and the subsequent neighbourhoods of the possibly added objects in the cluster (constructing a *queue* of objects). It adds objects to the current cluster candidate while the queue is not empty.

If the $MinWeight$ predicate for the first object q is false, then q is classified as noise and the next unclassified object is tested against $NPred$ and $MinWeight$ for constructing the next cluster candidate. The algorithm repeats the above steps while there are unclassified objects in the set.

Algorithm 43 Generalised Density-Based SCAN (GDBSCAN) algorithm for clustering spatial objects (J. Sander et al. [147]). Algorithm particularised for the case of ellipsoids.

Require: D set of spatial objects with R*-tree structure (insert objects using algorithm 51).
Require: $MinCard$ cluster volume threshold.
Require: τ overlapping volume ratio criterion as given in condition 2.64.
Require: M_{max} maximum number points for estimating the overlapping volume.
Require: tol convergence tolerance for algorithm 28.

```
1: procedure GDBSCAN(D, MinCard, tau, M_max, tol)
2:     ClusterId = nextId(NOISE)          ▷ Objects in set D are UNCLASSIFIED.
3:     for i = 1, ..., D.size() do
4:         Object = D.get(i)
5:         if Object.ClusterId = UNCLASSIFIED then
6:             if EXPANDCLUSTER(D, Object, ClusterId, MinCard, τ, M_max, tol) then
   ▷ See algorithm 44
7:                 ClusterId = nextId(ClusterId)
8:             end if
9:         end if
10:    end for
11: end procedure
```

Algorithm 44 Expand a cluster for the GDBSCAN algorithm 43 (J. Sander et al. [147]). Algorithm particularised for the case of ellipsoids.

Require: D set of spatial objects.
Require: $Object$ considered object in D.
Require: $ClusterId$ current cluster identifier (number in $\mathcal{Z}$).
Require: $MinCard$ cluster volume threshold.
Require: τ overlapping volume ratio criterion as given in condition 2.64.
Require: M_{max} maximum number of integration points for estimating the overlapping volume.
Require: tol convergence tolerance for algorithm 28.

procedure EXPANDCLUSTER(D, $Object$, $ClusterId$, $MinCard$, τ, M_{max}, tol)
  if WCARD($\{Object\}$)≤ 0 **then** ▷ Object not in selection. Algorithm 46.
    $D.changeClusterId(Object, UNCLASSIFIED)$
    return false
  end if
  $seeds = D$.NPREDNEIGHBOURHOOD($Object$, τ, M_{max}, tol) ▷ Algorithm 45
  if WCARD($seeds$)$< MinCard$ **then** ▷ No core object. Algorithm 46.
    $D.changeClusterId(Object, NOISE)$
    return false
  end if
  ▷ Still here ? Object is a core object.
  $D.changeClusterId(seeds, ClusterId)$
  $seeds.delete(Object)$
  while $seeds \neq empty$ **do**
    $currentObject = seeds.first()$
    $result = D$.NPREDNEIGHBOURHOOD($currentObject$, τ, M_{max}, tol) ▷ Algorithm 45
    if WCARD($result$)$\geq MinCard$ **then** ▷ Algorithm 46.
      for $i = 1, \ldots, result.size()$ **do**
        $P = result.get(i)$
        if WCARD($\{P\}$)> 0 and $P.clusterId \in \{UNCLASSIFIED, NOISE\}$ **then** ▷ Algorithm 46.
          if $P.clusterId = UNCLASSIFIED$ **then**
            $seeds.append(P)$
          end if
          $D.changeClusterId(P, ClusterId)$
        end if ▷ WCARD(...)> 0 and $UNCLASSIFIED$ or $NOISE$
      end for
    end if ▷ WCARD(...) $\geq MinCard$
    $seeds.delete(currentObject)$
  end while ▷ $seeds \neq empty$
  return true
end procedure

F.2.1 Remarks

- The GDBSCAN algorithm 43 assumes that the lowest non-negative object identifier (Id) is the "unclassified" Id, followed by the "noise" Id. The function $nextId$ simply increase the input Id by one: $nextId : \mathcal{Z} \to \mathcal{Z}, x \mapsto nextId(x) = x + 1$.
- the ExpandCluster algorithm 44 only consider object that are "selected", i.e. with a non-negative identifier, this allows to perform clustering on a subset of the database D.
- The $D.get(i)$ function returns the ith object from set D.

The NPred-neighbourhood of an object p in dataset D queries all intersecting objects q with p, using the above predicate $NPred$. It takes advantage of the R*-tree structure of the dataset D, testing p with the predicate $NPred$ only for objects q belonging to the same parent node as object p. For instance, in the simplified example given in Figure 2.35, if p is the object 1, only objects 2 and 3 are tested against p with the predicate $NPred$. The R*-tree data structure allows to reduce the complexity of one query from $O(n)$ to $O(log(n))$, where $n = \#D$ is the number of objects in the dataset. The NPred-neighbourhood procedure is given in algorithm 45.

Algorithm 45 Given an object (ellipsoid) belonging to a R*-tree data structure, return its neighbourhood satisfying the predicate $NPred$ (see algorithm 22).

Require: *Object* considered object in a R*-tree data structure.
Require: τ overlapping volume ratio criterion as given in condition 2.64.
Require: M_{max} maximum number of integration points for estimating the overlapping volume.
Require: *tol* convergence tolerance for algorithm 28.

```
1:  procedure NPredNeighbourhood(Object, τ, M_max, tol)
2:      neighbours = {Object}
3:      candidateNeighbours = QueryNeighbourhood(Object)      ▷ Algorithm 55.
4:      for candidate ∈ candidateNeighbours do
5:          if NPred(Object, candidate, τ, M_max, tol) then      ▷ Algorithm 22
6:              neighbours = neighbours ∪ {candidate}
7:          end if
8:      end for
9:      return neighbours
10: end procedure
```

The *wCard* function used here is simply the volume of the objects it receive as arguments. The *MinCard* volume threshold is simply set by the user, and can be seen as a minimum volume threshold. Algorithm 46 describes the *wCard* function.

Algorithm 46 *wCard* function used in this thesis. Simply computes the volume of ellipsoids it gets as arguments

Require: $\{\mathcal{E}_i\}_{i=1,\dots,m}$ finite set of ellipsoids.

procedure WCARD($\{\mathcal{E}_i\}_{i=1,\dots,m}$)
    $volume = 0$
    for $i = 1, \dots, m$ **do**
        Extract the semi-axes lengths r_1, r_2 and r_3 of the current ellipsoid $\mathcal{E}_i$.
        $volume = volume + \frac{4\pi}{3} r_1 r_2 r_3$
    end for
    return $volume$
end procedure

Appendix G

Algorithms for a *R*-tree* data structure

This appendix contains algorithms for handling a *R*-tree* data structure as they are given in the article of N. Beckmann et al. [17].

Algorithm 47 Subtree choice for the insertion of a new entry in a *R*-tree* (N. Beckmann et al. [17]).

Require: N the root of a *R*-tree*.

```
1:  procedure ChooseSubtree(N)
2:      if N is a leaf then
3:          return N
4:      else
5:          if Childpointers in N point to leaves then
6:              Choose the entry in N whose box needs least overlap enlargement to
                include the new data box.
7:              Resolve ties by choosing the entry whose box needs least volume en-
                largement, then the entry with the box of smallest volume.
8:          end if
9:          if Childpointers in N do not point to leaves then
10:             Choose the entry in N whose box needs least volume enlargement to
                include the new data box. Resolve ties by choosing the entry with
                the box of smallest volume.
11:         end if
12:     end if
13:     Set N to be the child node pointed to by the childpointer of the chosen entry
            and repeat from beginning.
14: end procedure
```

Algorithm 48 Node split if maximum number of child M is reached (N. Beckmann et al. [17]).

```
1: procedure Split
2:     Invoke ChooseSplitAxis to determine the axis, perpendicular to which the
           split is performed.                                  ▷ Algorithm 49
3:     Invoke ChooseSplitIndex to determine the best distribution into two
           groups along that axis.                              ▷ Algorithm 50
4:     Distribute the entries into two groups.
5: end procedure
```

Algorithm 49 Choose the split axis for the *Split* algorithm (N. Beckmann et al. [17]).

```
1: procedure CHOOSESPLITAXIS
2:     for Each axis do
3:         Sort the entries by the lower then by the upper volume-value, overlap-
               value and margin–value of their boxes and determine all distributions.
4:         Compute S, the sum of all margin–values of the different distributions.
5:     end for
6:     Choose the axis with the minimum S as split axis.
7: end procedure
```

Algorithm 50 Choose the split index along a given split axis (N. Beckmann et al. [17]).

```
1: procedure CHOOSESPLITINDEX
2:     Along the chosen split axis, choose the distribution with the minimum
           overlap-value.
3:     Resolve ties by choosing the distribution with minimum volume-value.
4: end procedure
```

Algorithm 51 Insert a new entry in a *R*-tree* (N. Beckmann et al. [17]).

```
1: procedure INSERTDATA
2:     Invoke INSERT(level) starting with the leaf level as a parameter, to insert a
           new data box.                                          ▷ Algorithm 52
3: end procedure
```

Algorithm 52 Insert a new entry in a *R*–tree* (N. Beckmann et al. [17]).

Require: level: leaf level.

```
1: procedure INSERT(level)
2:     Invoke CHOOSESUBTREE(N), with the level as a parameter, to find an appro-
           priate node N, in which to place the new entry E.                ▷
           Algorithm 47
3:     if N has less than M entries then
4:         Accommodate E in N.
5:     end if
6:     if N has M entries then
7:         Invoke OVERFLOWTREATMENT(N, level) with the level of N as
               parameter.
   ▷ Algorithm 53
8:     end if
9:     if OverflowTreatment was called and split occured then
10:        Propagate OverflowTreatment upwards if necessary.
11:    end if
12:    if OverflowTreatment caused a split of the root then
13:        Create a new root.
14:    end if
15:    Adjust all covering boxes in the insertion path such that they are minimum
           bonding boxes enclosing the children boxes.
16: end procedure
```

Algorithm 53 Split a node which has to contains more that M children (N. Beckmann et al. [17]).

Require: N: the node to split.
Require: level: the node level.
1: **procedure** OVERFLOWTREATMENT(N, level)
2: **if** level is not the root level and this is the first call of *OverflowTreatment* for the given level **then**
3: Invoke REINSERT(N). ▷ Algorithm 54
4: **else**
5: Invoke SPLIT. ▷ Algorithm 48
6: **end if**
7: **end procedure**

Algorithm 54 Reorganise boxes for node N (N. Beckmann et al. [17]).

Require: N: a node.
1: **procedure** REINSERT(N)
2: For all $M + 1$ entries of node N, compute the distance between the centres of their boxes and the centre of the bounding box of N.
3: Sort the entries in decreasing order of their distances computed here above.
4: Remove the first $p = 30\%$ of the entries from N and adjust the bounding box of N.
5: In the above sort, starting with the maximum distance (= far reinsert) or minimum distance (= close reinsert), invoke INSERT(level) to reinsert the entries at the level of N. ▷ Algorithm 52
6: **end procedure**

Algorithm 55 Perform a neighbourhood query for a given R*-tree.

Require: *Object* an object in the R*–tree.
1: **procedure** QUERYNEIGHBOURHOOD(*Object*)
2: Compute the bounding box B of *Object*.
3: Starting from the root node of the tree, select the sub–node from the root whose associated bounding box contains B.
4: If the current sub–node is not a leaf node, select the next sub–node from the current one whose associated bounding box contains B.
5: Go back to line 4 while the last selected sub–node is not a leaf node.
6: **return** Objects in current leaf node.
7: **end procedure**

Appendix H

Intersection volume between two colliding ellipsoids

This section is devoted to the numerical computation of the intersection volume $Vol\left(\mathcal{E}_1 \cap \mathcal{E}_2\right)$ of two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$. This intersection volume is used from the evaluation of the overlapping criterion 2.64.

H.1 Monte Carlo integration and the VEGAS algorithm

With an integration domain separable along each dimension taken as the intersection of axis-aligned box of the two considered ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$, the VEGAS algorithm of G.P. Lepage [129] can be used for estimating the intersection volume $Vol\left(\mathcal{E}_1 \cap \mathcal{E}_2\right)$ with the integrand given in equation H.1.

$$f^{inter}_{\mathcal{E}_1,\mathcal{E}_2} : \mathcal{R}^3 \to \{0,1\}; \mathbf{x} \mapsto f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in \mathcal{E}_1 \cap \mathcal{E}_2 \\ 0 \text{ elsewhere} \end{cases} \tag{H.1}$$

H.1.1 Description of the VEGAS algorithm

The following section describes how G.P. Lepage's algorithm works and is mainly extracted from G.P. Lepage's article [129]. It is given here for the sake of completeness. The complete procedure is given in algorithm 56.

H.1.1.1 Classic Monte Carlo integration

Given an integrable function $f : \mathcal{R}^n \to \mathcal{R}; \mathbf{x} \mapsto f(\mathbf{x})$ over a domain $\Omega \subset \mathcal{R}^n$, the integral $I = \int_\Omega f(\mathbf{x})\, d\mathbf{x}$ is to be calculated. If a primitive of f is not known and if f and/or the integration domain Ω are not well-suited for classic integration methods (as the Gauss-Legendre quadrature rule), Monte Carlo methods may be effective alternatives for computing the integral I.

The classic Monte Carlo integration method is the following. Given a set $\{\mathbf{x}\}$ of M point chosen randomly in the integration domain Ω following a probability density function $p : \mathcal{R}^n \to \mathcal{R}^+; \mathbf{x} \mapsto p(\mathbf{x})$, the integral I can be approximated by equation H.2.

$$S^{(1)} = \frac{Vol(\Omega)}{M} \sum_{\mathbf{x}} \frac{f(\mathbf{x})}{p(\mathbf{x})} \text{ (G.P. Lepage [129], eq. 1.)} \tag{H.2}$$

Where $Vol(\Omega) = \int_\Omega 1\, d\mathbf{x}$ and $\int_\Omega p(\mathbf{x})\, d\mathbf{x} = 1$. In that case, we have:

$$lim_{M\to\infty} S^{(1)} = I \tag{H.3}$$

For different sets of M points, $S^{(1)}$ in equation H.2 will fluctuate with a variance σ^2 estimated by equation H.4 for M big.

$$\sigma^2 \approx \frac{S^{(2)} - \left(S^{(1)}\right)^2}{M-1} \text{ (G.P. Lepage [129], eq. 2)} \tag{H.4}$$

Where $S^{(2)} = \frac{(Vol(\Omega))^2}{M} \sum_{\mathbf{x}} \left(\frac{f(\mathbf{x})}{p(\mathbf{x})}\right)^2$.

Integration method of G.P. Lepage. In general, Monte Carlo integration methods require a high number of evaluation points M in order to obtain an accurate result, i.e. a small variance σ^2. However, evaluating numerous time a function may be computationally expensive. G.P. Lepage's methods consist in avoiding a to high number of evaluation points M, while minimising the variance σ^2.

This aim is achieved iteratively adapting the probability density function $p(\mathbf{x})$. The idea, known as *importance sampling* consist in adapting the probability density function $p(\mathbf{x})$ such that the chosen random points $\{\mathbf{x}\}$ concentrate where the function $|f(\mathbf{x})|$ is larger. As probability density function $p(\mathbf{x})$ G.P. Lepage suggests to use the piecewise constant function given in equation H.5.

$$\begin{aligned} &p(\mathbf{x}) = p^{(1)}(x^{(1)}) \dots p^{(n)}(x^{(n)}), \\ &p^{(\mu)}(x^{(\mu)}) = \tfrac{1}{N\Delta x_i^{(\mu)}}, x_i^{(\mu)} - \Delta x_i^{(\mu)} \leq x^{(\mu)} \leq x_i^{(\mu)}, \\ &i = 1, \dots, N,\ \mu = 1, \dots, n \end{aligned} \tag{H.5}$$

Where N is the number of piece along each direction $\mu = 1, \dots, n$ and the $\Delta x_i^{(\mu)}$ are intervals along each direction $\mu = 1, \dots, n$ such that $\sum_{i=1}^{N} \Delta x_i^{(\mu)} = 1,\ \mu = 1, \dots n$.

The procedure consists now into adapting the different intervals $\Delta x_i^{(\mu)}$ such that the variance σ^2 is minimised. To this aim, G.P. Lepage suggests to subdivide each interval $\delta x_i^{(\mu)}$ into $(m_i^{(\mu)} + 1)$ subintervals given by equation H.6.

$$m_i^{(\mu)} = K \frac{\overline{f}_i^{(\mu)} \Delta x_i^{(\mu)}}{\sum_{j=1}^{N} \overline{f}_j^{(\mu)} \Delta x_j^{(\mu)}},\ i = 1, \dots, N,\ \mu = 1, \dots, n \tag{H.6}$$

Where K is a constant and $\overline{f}_i^{(\mu)} = \sum_{x^{(1)}} \dots \sum_{x^{(\mu-1)}} \sum_{x^{(\mu)} \in x_i^{(\mu)} - \Delta x_i^{(\mu)}} \sum_{x^{(\mu+1)}} \dots \sum_{x^{(n)}} \frac{|f(\mathbf{x})|}{p^{(1)}(x^{(1)}) \dots p^{(\mu-1)}(x^{(\mu-1)}) p^{(\mu+1)}(x^{(\mu+1)}) \dots p^{(n)}(x^{(n)})}$

Note that in practice, and in order to ensure algorithmic stability, a damped version of the expression of $m_i^{(\mu)}$, involving a kind of "learning parameter" α, is used (see algorithm 56 line 34).

Once the subintervals along each direction μ are computed, these subintervals are amalgamated into N new intervals along each direction μ. Neighbouring intervals where evaluations of $|f(\mathbf{x})|$ are low are merged, while intervals with high evaluations of $|f(\mathbf{x})|$ are kept. The change of these intervals directly impact the probability density function $p(\mathbf{x})$ which is updated. This process is then repeated with a new set of M randomly distributed points $\{\mathbf{x}\}$ according to the new probability density function $p(\mathbf{x})$ until convergence is met. Figure H.1 shows a sketch of this process.

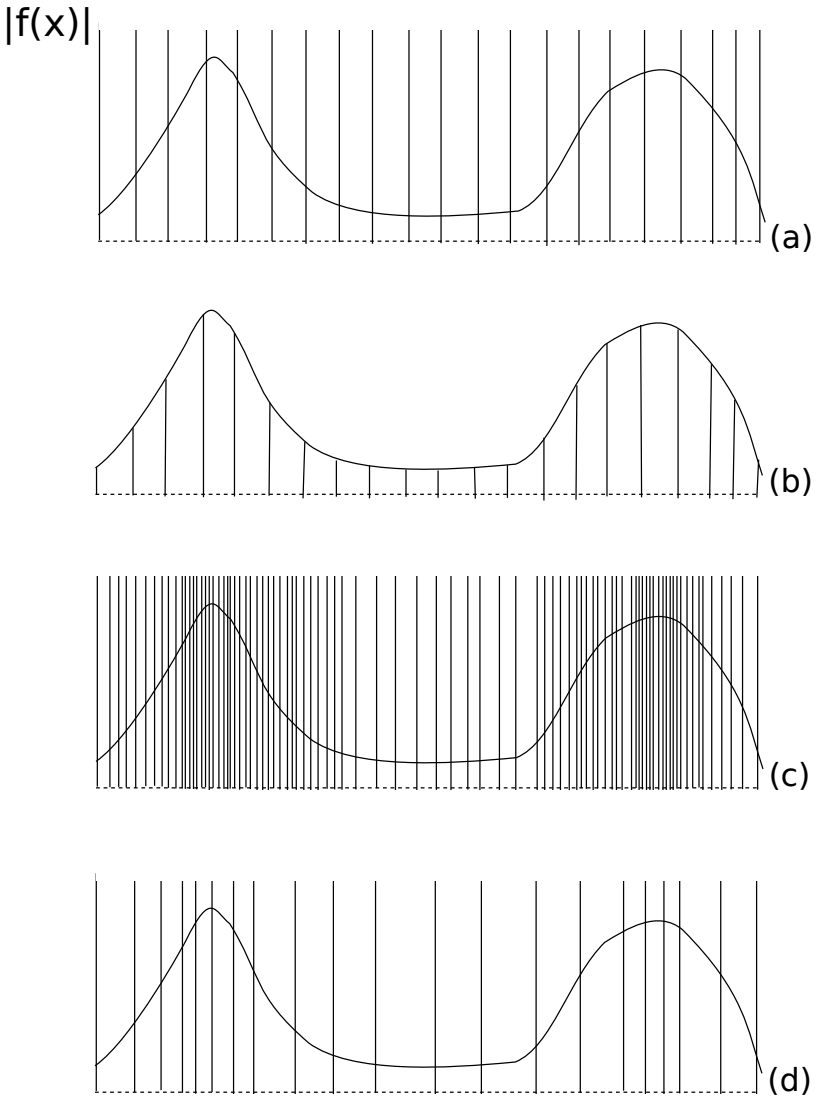


FIGURE H.1: One–dimensional sketch of the process dividing interval into subintervals and merging the obtained subintervals according to the evaluations of a function $|f(\mathbf{x})|$. a) Initial intervals. b) Sampling of the function $|f(\mathbf{x})|$. c) Division of the intervals into subintervals. d) Amalgamation of the subintervals.

H.1.1.2 Error estimation

At each iteration involving a new set of M random points $\{\mathbf{x}\}$, the quantities given in equations H.7 are computed.

$$\begin{aligned} \bar{I} &= \sigma_{\bar{I}}^2 \sum_j \frac{I_j}{\sigma_j^2} \\ \sigma_{\bar{I}} &= \left(\sum_j \frac{1}{\sigma_j^2} \right)^{-1/2} \end{aligned} \quad \text{(G.P. Lepage [129], eq. 5.)} \tag{H.7}$$

Where I_j and σ_j are the integral and standard deviation estimated at iteration j.

The error at a given iteration can then be estimated as given by the sum H.8 which follows a χ^2 distribution[1].

$$\chi^2 \approx \sum_j \frac{\left(I_i - \bar{I}\right)^2}{\sigma_j^2} \quad \text{(G.P. Lepage [129], eq. 7.)} \tag{H.8}$$

If the sum H.8 greatly exceeds the number of carried iterations, the integral estimation should not be trusted.

Algorithm 56 G.P. Lepage's VEGAS algorithm [129]

Require: Integrand function $f^{inter}_{\mathcal{E}_1,\mathcal{E}_2} : \mathcal{R}^n \to \mathcal{R}$.
Require: Number of integration points M.
Require: Integration domain $\Omega = [d_{min,1}, d_{max,1}] \times \ldots \times [d_{min,n}, d_{max,n}]$.
Require: Number of subdivisions along each axis N.

procedure VEGAS($f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, M, Ω, N)
  $\alpha \leftarrow 1.5$ ▷ GSL default value for convergence parameter.
  $K \leftarrow 1000$ ▷ Typical value according to G.P. Lepage [129].
  $trial \leftarrow 1$
  for each subdivision $i = 1, \ldots, N$ **do**
    $\Delta x_i^{(\mu)} \leftarrow \frac{d_{max,i} - d_{min,i}}{N}$, $\mu = 1, \ldots, n$
    $p^{(\mu)}(x^{(\mu)}) \leftarrow \frac{1}{\Delta x_i^{(\mu)}}$, $\mu = 1, \ldots, n$
  end for
  $p(\mathbf{x}) \leftarrow p^{(1)}(x^{(1)}) \ldots p^{(n)}(x^{(n)})$
  $B_{j_1,\ldots,j_n} \leftarrow \left[d_{min,1} + j_1 \Delta x_{j_1}^{(1)}, d_{min,1} + (j_1+1)\Delta x_{j_1}^{(1)}\right] \times \ldots \times \left[d_{min,n} + j_n \Delta x_{j_n}^{(n)}, d_{min,n} + (j_n+1)\Delta x_{j_n}^{(n)}\right]$, $j_1, \ldots, j_n = 0, \ldots, N-1$
  $\mathbf{x} \leftarrow \{$set of two uniform random points in each box $B_{j_1,\ldots,j_n}\}$
  $Vol(\Omega) \leftarrow \prod_{\mu=1}^{n} \left(d_{max,\mu} - d_{min,\mu}\right)$
  $\frac{\tilde{\chi}^2}{trial} \leftarrow$ VEGASLOOP($p(\mathbf{x})$, $f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, $\mathbf{x}$, Ω, $Vol(\Omega)$, M, N, $trial$)
  while $\left(\frac{\tilde{\chi}^2}{trial} - 1\right) > 0.1$ **do**
    $\left\{\frac{\tilde{\chi}^2}{trial}, \bar{I}\right\} \leftarrow$ VEGASLOOP($p(\mathbf{x})$, $f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, $\mathbf{x}$, Ω, $Vol(\Omega)$, M, N, $trial$)
    $trial \leftarrow trial + 1$
  end while
  return $\bar{I}$
end procedure

[1] A random variable X follows a χ^2 distribution if it is the sum of n normal independent random variables $\{Y_i\}_{i=1,\ldots,n}$ of mean 0 and variance 1.

procedure VEGASLOOP($p(\mathbf{x})$, $f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}$, $\mathbf{x}$, Ω, $Vol(\Omega)$, M, N, $trial$)

$s^{(1)}_{trial} \leftarrow \frac{Vol\Omega}{M} \sum_{\mathbf{x}} \frac{f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}(\mathbf{x})}{p((x))}$

$S^{(2)}_{trial} \leftarrow \frac{Vol^2(\Omega)}{M} \sum_{\mathbf{x}} \left(\frac{f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}(\mathbf{x})}{p(\mathbf{x})} \right)^2$

$\tilde{\sigma}^2_{trial} \leftarrow \frac{S^{(2)}_{trial} - \left(S^{(1)}_{trial}\right)^2}{M-1}$

$\sigma_{\overline{I}} \leftarrow \left(\sum_{j=1}^{trial} \frac{1}{\tilde{\sigma}^2_j} \right)^{-1/2}$

$\overline{I} \leftarrow \sigma^2_{\overline{I}} \sum_{j=1}^{trial} \frac{S^{(1)}_j}{\tilde{\sigma}^2_j}$

$\frac{\tilde{\chi}^2}{trial} \leftarrow \sum_{j=1}^{trial} \frac{\left(S^{(1)}_j - \overline{I}\right)^2}{\tilde{\sigma}^2_j}$

for each axis $\mu = 1, \ldots, n$ **do**

  for each subinterval $i = 1, \ldots, N$ **do**

$$\overline{f}^{(\mu)}_i \leftarrow \sum_{x^{(1)}} \cdots \sum_{x^{(\mu-1)}} \sum_{x^{(\mu)} \in x^{(\mu)}_i - \Delta x^{(\mu)}_i} \sum_{x^{(\mu+1)}} \cdots \sum_{x^{(n)}} \frac{f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}(\mathbf{x})}{p^{(1)}(x^{(1)}) \ldots p^{(\mu-1)}(x^{(\mu-1)}) p^{(\mu+1)}(x^{(\mu+1)}) \ldots p^{(n)}(x^{(n)})}$$

  end for

end for

for each axis $\mu = 1, \ldots, n$ **do**

  for each subinterval $i = 1, \ldots, N$ **do**

$$m^{(\mu)}_i \leftarrow K \left\{ \frac{\overline{f}^{(\mu)}_i \Delta x^{(\mu)}_i}{\sum_{j=1}^{N} \overline{f}^{(\mu)}_j \Delta x^{(\mu)}_j} \; \frac{1}{log\left(\frac{\overline{f}^{(\mu)}_i \Delta x^{(\mu)}_i}{\sum_{j=1}^{N} \overline{f}^{(\mu)}_j \Delta x^{(\mu)}_j} \right)} \right\}^{\alpha}$$

  end for

end for

for each axis $\mu = 1, \ldots, n$ **do**

  Subdivide each interval increment $\Delta x^{(\mu)}_i$ in $(m^{(\mu)}_i + 1)$ subintervals.

  Amalgamate all the new subintervals along axis μ into N bigger intervals according to the weight of $|f^{inter}_{\mathcal{E}_1,\mathcal{E}_2}(\mathbf{x})|$.

end for

return $\left\{ \frac{\tilde{\chi}^2}{trial}, \overline{I} \right\}$

end procedure

Bibliography

[1] Hamza Sulayman Abdullahi, Yicheng Liang, and Shuming Gao. "Predicting the elastic properties of closed-cell aluminum foams: a mesoscopic geometric modeling approach". In: *SN Applied Sciences* 1.4 (2019), pp. 1–13. ISSN: 25233971. DOI: 10.1007/s42452-019-0382-y. URL: https://doi.org/10.1007/s42452-019-0382-y.

[2] James Ahrens, Berk Geveci, and Charles Law. "ParaView: An End-User Tool for Large Data Visualization". In: *Visualization Handbook*. ISBN 978-0123875822. Elsevier, 2005.

[3] Riyadh Al-Raoush and Khalid A. Alshibli. "Distribution of local void ratio in porous media systems from 3D X-ray microtomography images". In: *Physica A: Statistical Mechanics and its Applications* 361.2 (2006), pp. 441–456. ISSN: 03784371. DOI: 10.1016/j.physa.2005.05.043. URL: http://linkinghub.elsevier.com/retrieve/pii/S0378437105004905.

[4] Andreas Alpers et al. "Generalized balanced power diagrams for 3D representations of polycrystals". In: *Philosophical Magazine* 95.9 (2015), pp. 1016–1028. ISSN: 14786443. DOI: 10.1080/14786435.2015.1015469. arXiv: /arxiv.org/abs/1411.4535 [http:]. URL: http://dx.doi.org/10.1080/14786435.2015.1015469.

[5] E Anderson et al. *Lapack User's Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8.

[6] J Andersons et al. "Anisotropy of the stiffness and strength of rigid low-density closed-cell polyisocyanurate foams". In: *Materials and Design* 92 (2016), pp. 836–845. ISSN: 18734197. DOI: 10.1016/j.matdes.2015.12.122. URL: http://linkinghub.elsevier.com/retrieve/pii/S0264127515309813.

[7] E. W. Andrews et al. "Size effects in ductile cellular solids. Part II: Experimental results". In: *International Journal of Mechanical Sciences* 43.3 (2001), pp. 701–713. ISSN: 00207403. DOI: 10.1016/S0020-7403(00)00043-6.

[8] Frederick Arand and Jürgen Hesser. "Quantitative morphological analysis and digital modeling of polydisperse anisotropic carbon foam". In: *Carbon* 136 (2018), pp. 11–20. ISSN: 00086223. DOI: 10.1016/j.carbon.2018.04.049. URL: https://linkinghub.elsevier.com/retrieve/pii/S000862231830410X.

[9] Brett M Averick et al. "The MINPACK-2 Test Problem Collection". In: *Performance Computing* June (1992). DOI: 10.2172/79972. URL: https://www.osti.gov/biblio/79972.

[10] Jean Babaud et al. "Uniqueness of the Gaussian Kernel for Scale-Space Filtering". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.1 (1986), pp. 26–33. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767749. URL: http://ieeexplore.ieee.org/document/4767749/.

[11] T Bailey and K Jain. "A Note on Distance-Weighted k-Nearest Neighbor Rules". In: *IEEE Transactions on Systems, Man, and Cybernetics* 8.4 (1978), pp. 311–313. ISSN: 0018-9472. DOI: 10.1109/TSMC.1978.4309958. URL: http://ieeexplore.ieee.org/document/4309958/.

[12] John Banhart. "Manufacture, characterisation and application of cellular metals and metal foams". In: *Progress in Materials Science* 46.6 (2001), pp. 559–632. ISSN: 00796425. DOI: 10.1016/S0079-6425(00)00002-5. URL: https://linkinghub.elsevier.com/retrieve/pii/S0079642500000025.

[13] A. T. Barnes et al. "Dynamic crushing of aluminum foams: Part i - Experiments". In: *International Journal of Solids and Structures* 51.9 (2014), pp. 1631–1645. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2013.11.019. URL: http://dx.doi.org/10.1016/j.ijsolstr.2013.11.019.

[14] H. Bart-Smith et al. "Compressive deformation and yielding mechanisms in cellular Al alloys determined using X-ray tomography and surface strain mapping". In: *Acta Materialia* 46.10 (1998), pp. 3583–3592. ISSN: 13596454. DOI: 10.1016/S1359-6454(98)00025-1.

[15] Richard Beare. "Histogram-based thresholding - some missing methods". In: *The Insight Journal* (2011), pp. 1–5. DOI: 10.54294/efycla. URL: http://hdl.handle.net/10380/.

[16] Richard Beare and Gaëtan Lehman. "Finding regional extrema - methods and performance". In: *Insight Journal* (2005), pp. 1–7. DOI: 10.54294/hva3gz. URL: http://hdl.handle.net/1926/153.

[17] Norbert Beckmann et al. "The R*-tree: an efficient and robust access method for points and rectangles". In: *ACM SIGMOD Record* 19.2 (1990), pp. 322–331. ISSN: 01635808. DOI: 10.1145/93605.98741. URL: http://portal.acm.org/citation.cfm?doid=93605.98741.

[18] S Beucher. "The Watershed Transformation Applied to Image Segmentation". In: *Proceedings of the 10th Pfefferkorn Conference on Signal and Image Processing in Microscopy and Microanalysis* (1992), pp. 299–314. ISSN: 0892953X. DOI: 10.1.1.24.5229. URL: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1466&context=microscopy.

[19] Alexander Bezrukov, Monika Bargieł, and Dietrich Stoyan. "Statistical Analysis of Simulated Random Packings of Spheres". en. In: *Part. Part. Syst. Charact.* 19.2 (May 2002), pp. 111–118. ISSN: 1521-4117. DOI: 10.1002/1521-4117(200205)19:2<111::AID-PPSC111>3.0.CO;2-M.

[20] Alexander Bezrukov, Dietrich Stoyan, and Monika Bargieł. "Spatial Statistics for Simulated Packings of Spheres". In: *Image Analysis & Stereology* 20.3 (2001), p. 203. ISSN: 1854-5165. DOI: 10.5566/ias.v20.p203-206. URL: https://www.ias-iss.org/ojs/IAS/article/view/680.

[21] Andreas Bieniek et al. "A parallel watershed algorithm". In: *Proceedings of the 10th Scandinavian Conference on Image Analysis (SCIA'97)*. Proceedings of the 10th Scandinavian Conference on Image Analysis (SCIA'97), 1997, pp. 237–244.

[22] Jean Daniel Boissonnat and Monique Teillaud. *Effective Computational Geometry for Curves and Surfaces*. Ed. by Jean-Daniel Boissonnat and Monique Teillaud. Springer Berlin Heidelberg, 2006, pp. 1–343. ISBN: 978-3-540-33258-9. DOI: 10.1007/978-3-540-33259-6. URL: http://link.springer.com/10.1007/978-3-540-33259-6.

[23] Gunilla Borgefors. "Another comment on "a note on 'distance transformations in digital images'"". In: *CVGIP: Image Understanding* 54.2 (1991), pp. 301–306. ISSN: 10499660. DOI: 10.1016/1049-9660(91)90070-6. URL: http://linkinghub.elsevier.com/retrieve/pii/S0734189X86800470.

[24] Gunilla Borgefors. "Distance transformations in arbitrary dimensions". In: *Computer Vision, Graphics, and Image Processing* 27.3 (1984), pp. 321–345. ISSN: 0734189X. DOI: 10.1016/0734-189X(84)90035-5. URL: http://linkinghub.elsevier.com/retrieve/pii/0734189X84900355.

[25] D. P. Bourne et al. "Laguerre tessellations and polycrystalline microstructures: a fast algorithm for generating grains of given volumes". In: *Philosophical Magazine* 100.21 (2020), pp. 2677–2707. ISSN: 1478-6435. DOI: 10.1080/14786435.2020.1790053. arXiv: 1912.07188. URL: https://www.tandfonline.com/doi/full/10.1080/14786435.2020.1790053.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge. Cambridge, 2004, pp. 1–730. ISBN: 9780521833783. URL: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

[27] Mauro Bracconi et al. "A systematic procedure for the virtual reconstruction of open-cell foams". In: *Chemical Engineering Journal* 315 (2017), pp. 608–620. ISSN: 13858947. DOI: 10.1016/j.cej.2017.01.069. URL: http://dx.doi.org/10.1016/j.cej.2017.01.069.

[28] Derek Bradley and Gerhard Roth. "Adaptive Thresholding using the Integral Image". In: *Journal of Graphics Tools* 12.2 (2007), pp. 13–21. ISSN: 1086-7651. DOI: 10.1080/2151237X.2007.10129236. URL: https://www.tandfonline.com/doi/full/10.1080/2151237X.2007.10129236.

[29] David K. Buck and Aarron A. Collin. *POV-Ray - The Persistence of Vision Raytracer*. 2004. URL: http://www.povray.org.

[30] Mauricio Francisco Caliri Júnior et al. "Study of an anisotropic polymeric cellular material under compression loading". In: *Materials Research* 15.3 (2012), pp. 359–364. ISSN: 1516-1439. DOI: 10.1590/S1516-14392012005000034.

[31] C. Chen and N.A. Fleck. "Size effects in the constrained deformation of metallic foams". In: *Journal of the Mechanics and Physics of Solids* 50.5 (2002), pp. 955–977. ISSN: 00225096. DOI: 10.1016/S0022-5096(01)00128-4. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022509601001284.

[32] Youming Chen, Raj Das, and Mark Battley. "Effects of cell size and cell wall thickness variations on the stiffness of closed-cell foams". In: *International Journal of Solids and Structures* 52 (2015), pp. 150–164. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2014.09.022. URL: http://linkinghub.elsevier.com/retrieve/pii/S0020768314003692.

[33] Jierong Cheng and Jagath C Rajapakse. "Segmentation of clustered nuclei with shape markers and marking function". In: *IEEE Transactions on Biomedical Engineering* 56.3 (2009), pp. 741–748. ISSN: 00189294. DOI: 10.1109/TBME.2008.2008635. URL: http://www.ncbi.nlm.nih.gov/pubmed/19272880.

[34] P. Cignoni, C. Rocchini, and R. Scopigno. "Metro: Measuring Error on Simplified Surfaces". In: *Computer Graphics Forum* 17.2 (1998), pp. 167–174. ISSN: 0167-7055. DOI: 10.1111/1467-8659.00236. URL: https://onlinelibrary.wiley.com/doi/10.1111/1467-8659.00236.

[35] Olivier Cuisenaire. "Locally adaptable mathematical morphology using distance transformations". In: *Pattern Recognition* 39.3 (2006), pp. 405–416. ISSN: 00313203. DOI: 10.1016/j.patcog.2005.07.009. URL: http://linkinghub.elsevier.com/retrieve/pii/S0031320305003006.

[36] Vladimir Curic et al. "Adaptive mathematical morphology - A survey of the field". In: *Pattern Recognition Letters* 47 (2014), pp. 18–28. ISSN: 01678655. DOI: 10.1016/j.patrec.2014.02.022. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167865514000725.

[37] Per Erik Danielsson. "Euclidean distance mapping". In: *Computer Graphics and Image Processing* 14.3 (1980), pp. 227–248. ISSN: 0146664X. DOI: 10.1016/0146-664X(80)90054-4.

[38] P De Jaeger et al. "An experimentally validated and parameterized periodic unit-cell reconstruction of open-cell foams". In: *Journal of Applied Physics* 109.10 (2011), p. 103519. ISSN: 00218979. DOI: 10.1063/1.3587159.

[39] Robin Deits and Russ Tedrake. "Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming". In: *Springer Tracts in Advanced Robotics*. Ed. by H Levent Akin et al. Vol. 107. Springer Tracts in Advanced Robotics. Springer International Publishing, 2015, pp. 109–124. ISBN: 9783319165943. DOI: 10.1007/978-3-319-16595-0.

[40] R. Destefanis et al. "Selecting Enhanced Space Debris Shields for Manned Spacecraft". en. In: *International Journal of Impact Engineering* 33.1-12 (Dec. 2006), pp. 219–230. ISSN: 0734743X. DOI: 10.1016/j.ijimpeng.2006.09.065.

[41] Aaron Drake et al. "Horn and horn core trabecular bone of bighorn sheep rams absorbs impact energy and reduces brain cavity accelerations during high impact ramming of the skull". In: *Acta Biomaterialia* 44 (2016), pp. 41–50. ISSN: 18787568. DOI: 10.1016/j.actbio.2016.08.019. URL: http://dx.doi.org/10.1016/j.actbio.2016.08.019.

[42] Matthias Eck et al. "Multiresolution analysis of arbitrary meshes". In: *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* 1 (1995), pp. 173–180. DOI: 10.1145/218380.218440.

[43] Karim Ehab Moustafa Kamel, Bernard Sonon, and Thierry Jacques Massart. "An integrated approach for the conformal discretization of complex inclusion-based microstructures". In: *Computational Mechanics* 64.4 (2019), pp. 1049–1071. ISSN: 1432-0924. DOI: 10.1007/s00466-019-01693-4.

[44] M.I. El Ghezal and I. Doghri. "Porous plasticity: Predictive second moment homogenization models coupled with Gurson's single cavity stress-strain solution". In: *International Journal of Plasticity* 108.October 2017 (2018), pp. 201–221. ISSN: 07496419. DOI: 10.1016/j.ijplas.2018.05.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0749641917305946.

[45] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231. URL: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[46] Ricardo Fabbri et al. "2D Euclidean distance transform algorithms". In: *ACM Computing Surveys* 40.1 (2008), pp. 1–44. ISSN: 03600300. DOI: 10.1145/1322432.1322434.

[47] Frédéric Feyel. "A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua". In: *Computer Methods in Applied Mechanics and Engineering* 192.28-30 (2003), pp. 3233–3244. ISSN: 00457825. DOI: 10.1016/S0045-7825(03)00348-7.

[48] T. Fiedler et al. "Computed Tomography Based Finite Element Analysis of the Thermal Properties of Cellular Aluminium". en. In: *Materialwissenschaft und Werkstofftechnik* 40.3 (Mar. 2009), pp. 139–143. ISSN: 09335137, 15214052. DOI: 10.1002/mawe.200900419.

[49] S. Forest et al. "Continuum modeling of strain localization phenomena in metallic foams". In: *Journal of Materials Science* 40.22 (2005), pp. 5903–5910. ISSN: 00222461. DOI: 10.1007/s10853-005-5041-6.

[50] J Fritz. "Extremum Problems with Inequalities as Side Conditions". In: *Studies and Essays, Courant Anniversary Volume (K. O. Friedrichs, O. E. Neugebauer, and J. J. Stoker, eds.)* Ed. by Wiley (Interscience). J. J. Stoker. Boston, 1984, p. 187204.

[51] S. Gaitanaros and S. Kyriakides. "Dynamic crushing of aluminum foams: Part II – Analysis". In: *International Journal of Solids and Structures* 51.9 (2014), pp. 1646–1661. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2013.11.020. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020768313004630.

[52] M Galassi et al. *GNU Scientific Library Reference Manual*. 3rd ed. Vol. 954161734. March. Network Theory Ltd, 2009, p. 592. ISBN: 0954161734. DOI: ISBN0954612078. URL: http://www.gnu.org/software/gsl/.

[53] Thierry Géraud, Hugues Talbot, and Marc Van Droogenbroeck. "Algorithms for Mathematical Morphology". In: *Mathematical Morphology: From Theory to Applications*. Ed. by Laurent Najman and Hugues Talbot. Hoboken, NJ, USA: ISTE {&} Wiley, 2013. Chap. 12, pp. 323–353. ISBN: 9781848212152. DOI: 10.1002/9781118600788.ch12. URL: http://hdl.handle.net/2268/41515.

[54] Christophe Geuzaine and Jean-François Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331. ISSN: 00295981. DOI: 10.1002/nme.2579.

[55] S Ghosh and S Moorthy. "Three dimensional Voronoi cell finite element model for microstructures with ellipsoidal heterogeneties". In: *Computational Mechanics* 34.6 (2004), pp. 510–531. ISSN: 01787675. DOI: 10.1007/s00466-004-0598-5. URL: http://link.springer.com/10.1007/s00466-004-0598-5.

[56] Lorna J. Gibson and Michael F. Ashby. *Cellular Solids: Structure and Properties*. second. Cambridge: Cambridge University Press, 1997. ISBN: 978-1-139-87832-6. DOI: 10.1017/CBO9781139878326.

[57] Michael Godehardt and Katja Schladitz. *Geometric Characterisation of Light Weight Composites Using Computer Tomographic Images*. ECNDT 2006 - We.1.6.3, 2006, pp. 1–8. URL: http://www.ndt.net/article/ecndt2006/doc/We.1.6.3.pdf.

[58] L. Gong and S. Kyriakides. "Compressive Response of Open Cell Foams Part II: Initiation and Evolution of Crushing". en. In: *International Journal of Solids and Structures* 42.5-6 (Mar. 2005), pp. 1381–1399. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2004.07.024.

[59] Antonin Guttman. "R-trees". In: *ACM SIGMOD Record* 14.2 (1984), p. 47. ISSN: 01635808. DOI: 10.1145/971697.602266. arXiv: ISBN0-89791-128-8. URL: http://portal.acm.org/citation.cfm?doid=971697.602266.

[60] A.G. Hanssen et al. "Validation of constitutive models applicable to aluminium foams". In: *International Journal of Mechanical Sciences* 44.2 (2002), pp. 359–406. ISSN: 00207403. DOI: 10.1016/S0020-7403(01)00091-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020740301000911.

[61] Volker Hardenacke and Jrg Hohe. "Assessment of space division strategies for generation of adequate computational models for solid foams". In: *International Journal of Mechanical Sciences* 52.12 (2010), pp. 1772–1782. ISSN: 00207403. DOI: 10.1016/j.ijmecsci.2010.09.011.

[62] S. Heinze et al. "Experimental and Numerical Investigation of Single Pores for Identification of Effective Metal Foams Properties: Experimental and Numerical Investigation of Single Pores for Identification of Effective Metal Foams Properties". en. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 98.5 (May 2018), pp. 682–695. ISSN: 00442267. DOI: 10.1002/zamm.201700045.

[63] Hugues Hoppe et al. "Mesh optimization". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*. Vol. d. New York, New York, USA: ACM Press, 1993, pp. 19–26. ISBN: 0897916018. DOI: 10.1145/166117.166119. URL: http://portal.acm.org/citation.cfm?doid=166117.166119.

[64] Hugues Hoppe et al. "Surface reconstruction from unorganized points". In: *ACM SIGGRAPH Computer Graphics* 26.2 (1992), pp. 71–78. ISSN: 0097-8930. DOI: 10.1145/142920.134011. URL: https://dl.acm.org/doi/10.1145/142920.134011.

[65] A. T. Huber and L. J. Gibson. "Anisotropy of foams". In: *Journal of Materials Science* 23.8 (1988), pp. 3031–3040. ISSN: 0022-2461. DOI: 10.1007/BF00547486. URL: http://link.springer.com/10.1007/BF00547486.

[66] J Inglada and E Christophe. "The orfeo toolbox remote sensing image processing software". In: *Ieee Igarss*. 2009, pp. 1–4.

[67] Wen-Yea Jang, Andrew M. Kraynik, and Stelios Kyriakides. "On the Microstructure of Open-Cell Foams and Its Effect on Elastic Properties". In: *International Journal of Solids and Structures* 45.7 (Apr. 2008), pp. 1845–1875. ISSN: 0020-7683. DOI: 10.1016/j.ijsolstr.2007.10.008.

[68] Wen Yea Jang and Stelios Kyriakides. "On the crushing of aluminum open-cell foams: Part I. Experiments". In: *International Journal of Solids and Structures* 46.3-4 (2009), pp. 617–634. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2008.09.008. URL: http://dx.doi.org/10.1016/j.ijsolstr.2008.09.008.

[69] Ibanez. Johnson McCormick. Fourth. Kitware, Inc., 2015. URL: https://itk.org/ItkSoftwareGuide.pdf.

[70] A. Jung and S. Diebels. "Microstructural Characterisation and Experimental Determination of a Multiaxial Yield Surface for Open-Cell Aluminium Foams". en. In: *Materials & Design* 131 (Oct. 2017), pp. 252–264. ISSN: 02641275. DOI: 10.1016/j.matdes.2017.06.017.

[71] A. Jung et al. "Open-cell aluminium foams with graded coatings as passively controllable energy absorbers". In: *Materials and Design* 87 (2015), pp. 36–41. ISSN: 18734197. DOI: 10.1016/j.matdes.2015.07.165. URL: http://dx.doi.org/10.1016/j.matdes.2015.07.165.

[72] Avinash C Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. Ed. by W R Crone et al. IEEE Press. New-York, 1987. ISBN: 0-87942-198-3. URL: http://www.slaney.org/pct/pct-toc.html.

[73] Jorgen Kampf et al. "Segmentation, statistical analysis, and modelling of the wall system in ceramic foams". In: *Materials Characterization* 99 (2015), pp. 38–46. ISSN: 10445803. DOI: 10.1016/j.matchar.2014.11.008. URL: http://linkinghub.elsevier.com/retrieve/pii/S1044580314003386.

[74] Sergey Kanaun and Oleksandr Tkachenko. "Representative volume element and effective elastic properties of open cell foam materials with random microstructures". In: *Journal of Mechanics of Materials and Structures* 2.8 (2007), pp. 1607–1628. ISSN: 1559-3959. DOI: 10.2140/jomms.2007.2.1607. URL: http://msp.org/jomms/2007/2-8/p17.xhtml.

[75] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Ed. by Leonard Kaufman and Peter J. Rousseeuw. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., 1990. ISBN: 9780470316801. DOI: 10.1002/9780470316801. URL: http://doi.wiley.com/10.1002/9780470316801.

[76] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Symposium on Geometry Processing*. Ed. by Alla Sheffer and Konrad Polthier. The Eurographics Association, 2006. ISBN: 3-905673-24-X. DOI: 10.2312/SGP/SGP06/061-070.

[77] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, 61–70. ISBN: 3905673363. URL: https://hhoppe.com/poissonrecon.pdf.

[78] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics* 32.3 (2013), pp. 1–13. ISSN: 0730-0301. DOI: 10.1145/2487228.2487237. URL: https://dl.acm.org/doi/10.1145/2487228.2487237.

[79] N. G. Kilingar et al. "Computational generation of open-foam representative volume elements with morphological control using distance fields". In: *European Journal of Mechanics, A/Solids* 78 (2019). ISSN: 09977538. DOI: 10.1016/j.euromechsol.2019.103847.

[80] Les Kitchen and Azriel Rosenfeld. "Gray-level corner detection". In: *Pattern Recognition Letters* 1.2 (1982), pp. 95–102. ISSN: 0167-8655. DOI: 10.1016/0167-8655(82)90020-4. URL: https://www.sciencedirect.com/science/article/pii/0167865582900204.

[81] J. Koplik and A. Needleman. "Void growth and coalescence in porous plastic solids". In: *International Journal of Solids and Structures* 24.8 (1988), pp. 835–853. ISSN: 00207683. DOI: 10.1016/0020-7683(88)90051-0. URL: https://linkinghub.elsevier.com/retrieve/pii/0020768388900510.

[82] André Körbes et al. "A Proposal for a Parallel Watershed Transform Algorithm for Real-Time Segmentation". In: *Proceedings of Workshop de Visao Computacional WVC*. Proceedings of Workshop de Visao Computacional WVC, 2009.

[83] V. G. Kouznetsova, M. G.D. Geers, and W. A.M. Brekelmans. "Multi-scale second-order computational homogenization of multi-phase materials: A nested finite element solution strategy". In: *Computer Methods in Applied Mechanics and Engineering* 193.48-51 (2004), pp. 5525–5550. ISSN: 00457825. DOI: 10.1016/j.cma.2003.12.073.

[84] V.G. Kouznetsova. "Computational homogenization for the multi-scale analysis of multi-phase materials". PhD thesis. 2002, pp. 1–134. DOI: 10.6100/IR560009. URL: https://pure.tue.nl/ws/files/3129502/200213807.pdf.

[85] Andrew M. Kraynik. "The Structure of Random Foam". en. In: *Adv. Eng. Mater.* 8.9 (Sept. 2006), pp. 900–906. ISSN: 1527-2648. DOI: 10.1002/adem.200600167.

[86] Andrew M. Kraynik, Douglas Reinelt, and Frank van Swol. "Structure of Random Foam". en. In: *Physical Review Letters* 93.20 (Nov. 2004). ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.93.208301.

[87] Andrew M. Kraynik, Douglas A. Reinelt, and Frank van Swol. "Structure of Random Monodisperse Foam". en. In: *Physical Review E* 67.3 (Mar. 2003). ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.67.031403.

[88] Claudia Lautensack. "Fitting three-dimensional Laguerre tessellations to foam structures". In: *Journal of Applied Statistics* 35.9 (2008), pp. 985–995. ISSN: 02664763. DOI: 10.1080/02664760802188112.

[89] Claudia Lautensack and Tetyana Sych. "3D Image Analysis of Open Foams Using Random Tessellations". In: *Image Analysis & Stereology* 25.2 (2006), pp. 87–93. ISSN: 1854-5165. DOI: 10.5566/ias.v25.p87-93. URL: https://www.ias-iss.org/ojs/IAS/article/view/795.

[90] Claudia Lautensack and Sergei Zuyev. "Random Laguerre tessellations". In: *Advances in Applied Probability* 40.3 (2008), pp. 630–650. ISSN: 0001-8678. DOI: 10.1239/aap/1222868179. URL: https://www.cambridge.org/core/product/identifier/S000186780000272X/type/journal_article.

[91] G. Legrain et al. "An X-FEM and level set computational approach for image-based modelling: Application to homogenization". In: *International Journal for Numerical Methods in Engineering* 86.7 (2011), pp. 915–934. ISSN: 00295981. DOI: 10.1002/nme.3085. URL: https://onlinelibrary.wiley.com/doi/10.1002/nme.3085.

[92] G Lehmann. "Binary morphological closing and opening image filters". In: *Insight Journal* (2005), pp. 1–6. DOI: 10.54294/bcwtvq. URL: https://www.insight-journal.org/browse/publication/58.

[93] Zhiqiang Li et al. "On crushing response of the three-dimensional closed-cell foam based on Voronoi model". In: *Mechanics of Materials* 68 (2014), pp. 85–94. ISSN: 01676636. DOI: 10.1016/j.mechmat.2013.08.009. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167663613001634.

[94] Yun-chia Liang and Josue Cuevas. "An Automatic Multilevel Image Thresholding Using Relative Entropy and Meta-Heuristic Algorithms". In: *Entropy* 15.6 (2013), pp. 2181–2209. ISSN: 1099-4300. DOI: 10.3390/e15062181. URL: http://www.mdpi.com/1099-4300/15/6/2181/.

[95] A Liebscher et al. "Stochastic multiscale modeling of metal foams". In: *Procedia IUTAM* 6 (2013), pp. 87–96. ISSN: 22109838. DOI: 10.1016/j.piutam.2013.01.010.

[96] André Liebscher. "Laguerre approximation of random foams". In: *Philosophical Magazine* 95.25 (2015), pp. 2777–2792. ISSN: 14786443. DOI: 10.1080/14786435.2015.1078511.

[97] Tony Lindeberg. "Scale-space for discrete signals". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.3 (1990).

[98] "CLARANS (Clustering Large Applications Based Upon Randomized Search)". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 330–330. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_2178.

[99] Alba Martinez Lopez-Reina. *Conversion des images 2D obtenues par tomographie en vue de la modélisation par éléments finis*. 2011.

[100] Dung D. Luong, Dinesh Pinisetty, and Nikhil Gupta. "Compressive properties of closed-cell polyvinyl chloride foams at low and high strain rates: Experimental investigation and critical review of state of the art". In: *Composites Part B: Engineering* 44.1 (2013), pp. 403–416. ISSN: 13598368. DOI: 10.1016/j.compositesb.2012.04.060. URL: http://dx.doi.org/10.1016/j.compositesb.2012.04.060.

[101] Ivan Macia. "Generalized computation of gaussian derivatives using itk". In: *The Insight Journal (December 2007)* (2007). DOI: 10.54294/mrg5is.

[102] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". In: 1 (1967). Ed. by L. M. Le Cam and J. Neyman, pp. 281–297.

[103] Kevin Mader et al. "Quantitative 3D characterization of cellular materials: Segmentation and morphology of foam". In: *Colloids and Surfaces A: Physicochemical and Engineering Aspects* 415 (2012), pp. 230–238. ISSN: 09277757. DOI: 10.1016/j.colsurfa.2012.09.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0927775712006188.

[104] K. R. Mangipudi and P. R. Onck. "Multiscale modelling of damage and failure in two-dimensional metallic foams". In: *Journal of the Mechanics and Physics of Solids* 59.7 (2011), pp. 1437–1461. ISSN: 00225096. DOI: 10.1016/j.jmps.2011.02.008. URL: http://dx.doi.org/10.1016/j.jmps.2011.02.008.

[105] Huina Mao, Romain Rumpler, and Peter Göransson. "An inverse method for characterisation of the static elastic Hooke's tensors of solid frame of anisotropic open-cell materials". In: *International Journal of Engineering Science* 147 (2020), p. 103198. ISSN: 00207225. DOI: 10.1016/j.ijengsci.2019.103198. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020722519322724.

[106] M. Marvi-Mashhadi, C.S. Lopes, and J. LLorca. "High fidelity simulation of the mechanical behavior of closed-cell polyurethane foams". In: *Journal of the Mechanics and Physics of Solids* 135 (2020), p. 103814. ISSN: 00225096. DOI: 10.1016/j.jmps.2019.103814. arXiv: 1912.01083. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022509619306477.

[107] M. Marvi-Mashhadi, C.S. Lopes, and J. LLorca. "Modelling of the mechanical behavior of polyurethane foams by means of micromechanical characterization and computational homogenization". In: *International Journal of Solids and Structures* 146 (2018), pp. 154–166. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2018.03.026. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020768318301343.

[108] T. J. Massart, R. H. J. Peerlings, and M. G. D. Geers. "An enhanced multi-scale approach for masonry wall computations with localization of damage". In: *International Journal for Numerical Methods in Engineering* 69.5 (2007), pp. 1022–1059. ISSN: 00295981. DOI: 10.1002/nme.1799. URL: https://onlinelibrary.wiley.com/doi/10.1002/nme.1799.

[109] Kazumi Matsui, Kenjiro Terada, and Kohei Yuge. "Two-scale finite element analysis of heterogeneous solids with periodic microstructures". In: *Computers and Structures* 82.7-8 (2004), pp. 593–606. ISSN: 00457949. DOI: 10.1016/j.compstruc.2004.01.004.

[110] Jacob Mattingley and Stephen Boyd. "CVXGEN: a code generator for embedded convex optimization". In: *Optimization and Engineering* 13.1 (2012), pp. 1–27. ISSN: 1389-4420. DOI: 10.1007/s11081-011-9176-9. URL: http://link.springer.com/10.1007/s11081-011-9176-9.

[111] C.R. Maurer, Rensheng Qi, and V. Raghavan. "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.2 (2003), pp. 265–270. DOI: 10.1109/TPAMI.2003.1177156.

[112] Cahal McVeigh et al. "Multiresolution analysis for material design". In: *Computer Methods in Applied Mechanics and Engineering* 195.37-40 (2006), pp. 5053–5076. ISSN: 00457825. DOI: 10.1016/j.cma.2005.07.027.

[113] Matthew D. Montminy, Allen R. Tannenbaum, and Christopher W. Macosko. "The 3D structure of real polymer foams". In: *Journal of Colloid and Interface Science* 280.1 (2004), pp. 202–211. ISSN: 00219797. DOI: 10.1016/j.jcis.2004.07.032. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021979704006824.

[114] S Moorthy and S Ghosh. "A model for analysis of arbitrary composite and porous microstructures with voronoi cell finite elements". In: *International Journal for Numerical Methods in Engineering* 39.14 (1996), pp. 2363–2398. ISSN: 0029-5981. DOI: 10.1002/(SICI)1097-0207(19960730)39:14<2363::AID-NME958>3.0.CO;2-D.

[115] Kishore Mosaliganti, Arnaud Gelas, and Sean Megason. "An Adaptive Thresholding Image Filter". In: *Insight Journal* (2009), pp. 1–5. DOI: 10.54294/4vjovf. URL: http://hdl.handle.net/10380/3133.

[116] Kilingar Nanda. "Generation and data-driven upscaling of open foam representational volume elements". PhD thesis. University of Liège, 2020, pp. 1–192.

[117] K Natesaiyer et al. "X-ray CT imaging and finite element computations of the elastic properties of a rigid organic foam compared to experimental measurements: insights into foam variability". In: *Journal of Materials Science* 50.11 (2015), pp. 4012–4024. ISSN: 0022-2461. DOI: 10.1007/s10853-015-8958-4. URL: http://link.springer.com/10.1007/s10853-015-8958-4.

[118] Nicholas Nethercote and Julian Seward. "Valgrind". In: *ACM SIGPLAN Notices* 42.6 (2007), p. 89. ISSN: 03621340. DOI: 10.1145/1273442.1250746. URL: http://portal.acm.org/citation.cfm?doid=1273442.1250746.

[119] V.-D. Nguyen and L Noels. "Computational homogenization of cellular materials". In: *International Journal of Solids and Structures* 51.11-12 (2014), pp. 2183–2203. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2014.02.029. URL: http://linkinghub.elsevier.com/retrieve/pii/S0020768314000778.

[120] V.-D. Nguyen et al. "Imposing periodic boundary condition on arbitrary meshes by polynomial interpolation". In: *Computational Materials Science* 55 (2012), pp. 390–406. ISSN: 09270256. DOI: 10.1016/j.commatsci.2011.10.017. URL: http://linkinghub.elsevier.com/retrieve/pii/S0927025611005866.

[121] Van-Dung Nguyen, Ling Wu, and Ludovic Noels. "A micro-mechanical model of reinforced polymer failure with length scale effects and predictive capabilities. Validation on carbon fiber reinforced high-crosslinked RTM6 epoxy resin". en. In: *Mechanics of Materials* 133 (June 2019), pp. 193–213. ISSN: 0167-6636. DOI: 10.1016/j.mechmat.2019.02.017.

[122] Van-Dung Nguyen, Ling Wu, and Ludovic Noels. "Unified Treatment of Microscopic Boundary Conditions and Efficient Algorithms for Estimating Tangent Operators of the Homogenized Behavior in the Computational Homogenization Method". en. In: *Computational Mechanics* 59.3 (Mar. 2017), pp. 483–505. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-016-1358-z.

[123] Vinh Phu Nguyen, Martijn Stroeven, and Lambertus Johannes Sluys. "Multiscale Continuou and Discontinuous Modeling of Heterogeneous Materials: a Review on Recent developments". In: *Journal of Multiscale Modelling* 03.04 (2011), pp. 229–270. ISSN: 1756-9737. DOI: 10.1142/S1756973711000509. URL: https://www.worldscientific.com/doi/abs/10.1142/S1756973711000509 .

[124] NIST. *X-Ray Mass Attenuation Coefficients*. URL: http://physics.nist.gov/PhysRefData/XrayMassCoef/chap2.html.

[125] J Ohser and K Schladitz. *Image processing and analysis*. Oxford: Clarendon Press, 2006. URL: https://pdfs.semanticscholar.org/d381/14f34c886657ca978b946d6da80f013d5787.pdf.

[126] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310076. URL: http://ieeexplore.ieee.org/document/4310076.

[127] S. D. Papka and S. Kyriakides. "Experiments and full-scale numerical simulations of in-plane crushing of a honeycomb". In: *Acta Materialia* 46.8 (1998), pp. 2765–2776. ISSN: 13596454. DOI: 10.1016/S1359-6454(97)00453-9.

[128] Per-olof Persson and Gilbert Strang. "A Simple Mesh Generator in MATLAB". In: *SIAM Review* 46.2 (2004), pp. 329–345. ISSN: 0036-1445. DOI: 10.1137/S0036144503429121. URL: http://epubs.siam.org/doi/10.1137/S0036144503429121.

[129] G Peter Lepage. "A new algorithm for adaptive multidimensional integration". In: *Journal of Computational Physics* 27.2 (1978), pp. 192–203. ISSN: 00219991. DOI: 10.1016/0021-9991(78)90004-9. URL: http://linkinghub.elsevier.com/retrieve/pii/0021999178900049.

[130] Tuan Q Pham. "Non-maximum Suppression Using Fewer than Two Comparisons per Pixel". In: *Advanced Concepts for Intelligent Vision Systems*. Vol. 6474. 1987. 2010, pp. 438–451. DOI: 10.1007/978-3-642-17688-3_41.

[131] Schneider Philip J. and David H Eberly. *Geometric Tools for Computer Graphics*. Elsevier Inc., 2003. ISBN: 978-1-55860-594-7. URL: http://www.sciencedirect.com/science/book/9781558605947https://www.geometrictools.com.

[132] Joseph Antoine Ferdinand Plateau. *Statique expérimentale et théorique des liquides soumis aux seules forces moléculaires*. Gauthier-Villars, 1873.

[133] Stephen B Pope. "Algorithms for ellipsoids". In: *Cornell University Report No. FDA-08-01* February (2008). URL: https://tcg.mae.cornell.edu/pubs/Pope_FDA_08.pdf.

[134] P D Raju and G Neelima. "Image Segmentation by using Histogram Thresholding". In: *Ijcset* 2.1 (2012), pp. 776–779. URL: https://ijcset.net/docs/Volumes/volume2issue1/ijcset2012020103.pdf.

[135] C Redenbach, I Shklyar, and H Andrä. "Laguerre tessellations for elastic stiffness simulations of closed foams with strongly varying cell sizes". In: *International Journal of Engineering Science* 50.1 (2012), pp. 70–78. ISSN: 00207225. DOI: 10.1016/j.ijengsci.2011.09.002. URL: http://linkinghub.elsevier.com/retrieve/pii/S0020722511001807.

[136] Claudia Redenbach. "Microstructure models for cellular materials". In: *Computational Materials Science* 44.4 (2009), pp. 1397–1407. ISSN: 09270256. DOI: 10.1016/j.commatsci.2008.09.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0927025608004266.

[137] Claudia Redenbach. "Microstructure models for cellular materials". In: *Computational Materials Science* 44.4 (2009), pp. 1397–1407. ISSN: 09270256. DOI: 10.1016/j.commatsci.2008.09.018. URL: http://dx.doi.org/10.1016/j.commatsci.2008.09.018.

[138] Claudia Redenbach. "Modelling foam structures using random tessellations". In: *European Congress of Stereology and Image Analysis*. Milan: ESCULAPIO Pub. Co., 2009. ISBN: 978-88-7488-310-3. URL: http://publica.fraunhofer.de/dokumente/N-111433.html.

[139] S Ridler, T.W. Calvard. "Picture Thresholding Using an Iterative Selection Method". In: *IEEE Transactions on Systems, Man and Cybernetics* 8.8 (1978), pp. 630–632. ISSN: 00189472. DOI: 10.1109/TSMC.1978.4310039.

[140] A P Roberts and E J Garboczi. "Elastic moduli of model random three-dimensional closed-cell cellular solids". In: *Acta materialia* 49 (2001), pp. 189–197. DOI: 10.1016/S1359-6454(00)00314-1. URL: http://www.sciencedirect.com/science/article/pii/S1359645400003141.

[141] A.P. Roberts and E.J. Garboczi. "Elastic properties of model random three-dimensional open-cell solids". In: *Journal of the Mechanics and Physics of Solids* 50.1 (2002), pp. 33–55. ISSN: 00225096. DOI: 10.1016/S0022-5096(01)00056-4. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022509601000564.

[142] Kevin Robinson and Paul F. Whelan. "Efficient morphological reconstruction: a downhill filter". In: *Pattern Recognition Letters* 25.15 (2004), pp. 1759–1767. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2004.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0167865504001692.

[143] A. Rosenfeld and J.L. Pfaltz. "Distance functions on digital pictures". In: *Pattern Recognition* 1.1 (1968), pp. 33–61. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(68)90013-7. URL: https://www.sciencedirect.com/science/article/pii/0031320368900137.

[144] Azriel Rosenfeld and A. Kak. *Digital Picture Processing*. 2nd. London: Academic Press, 1976. ISBN: 978-0-12-597302-1. DOI: 10.1016/C2009-0-21955-6.

[145] Toyofumi Saito and Jun-Ichiro Toriwaki. "New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications". In: *Pattern Recognition* 27.11 (1994), pp. 1551–1565. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(94)90133-3. URL: https://www.sciencedirect.com/science/article/pii/0031320394901333.

[146] Nader Salman, Mariette Yvinec, and Quentin Merigot. "Feature preserving mesh generation from 3D point clouds". In: *Eurographics Symposium on Geometry Processing* 29.5 (2010), pp. 1623–1632. ISSN: 17278384. URL: https://hal.inria.fr/inria-00497632.

[147] J Sander et al. "Density-based clustering in spatial databases: The algorithm gdbscan and its applications". In: *Data Mining and Knowledge Discovery* 194 (1998), pp. 169–194. DOI: 10.1023/A:1009745219419. URL: http://link.springer.com/article/10.1023/A:1009745219419.

[148] Fabian M. Schaller et al. "Set Voronoi diagrams of 3D assemblies of aspherical particles". In: *Philosophical Magazine* 93.31-33 (2013), pp. 3993–4017. ISSN: 1478-6435. DOI: 10.1080/14786435.2013.834389. URL: http://www.tandfonline.com/doi/abs/10.1080/14786435.2013.834389.

[149] K Schladitz et al. "Microstructural characterisation of open foams using 3d images". In: 148.148 (2008). URL: https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/2047/file/bericht148.pdf.

[150] Ondřej Šedivý et al. "3D reconstruction of grains in polycrystalline materials using a tessellation model with curved grain boundaries". In: *Philosophical Magazine* 96.18 (2016), pp. 1926–1949. ISSN: 1478-6435. DOI: 10.1080/14786435.2016.1183829. URL: http://www.tandfonline.com/doi/full/10.1080/14786435.2016.1183829.

[151] Ondřej Šedivý et al. "Description of the 3D morphology of grain boundaries in aluminum alloys using tessellation models generated by ellipsoids". In: *Image Analysis and Stereology* 36.1 (2017), pp. 5–13. ISSN: 18545165. DOI: 10.5566/ias.1656.

[152] Mehmet Sezgin and Bülent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation". In: *Journal of Electronic Imaging* 13.1 (2004), pp. 146–165. ISSN: 1017-9909. DOI: 10.1117/1.1631316. URL: http://electronicimaging.spiedigitallibrary.org/article.aspx?doi=10.1117/1.1631316.

[153] Hang Si. "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator". en. In: *ACM Transactions on Mathematical Software* 41.2 (Feb. 2015), pp. 1–36. ISSN: 00983500. DOI: 10.1145/2629697.

[154] Shweta Singh and Naresh Bhatnagar. "A survey of fabrication and application of metallic foams (1925-2017)". In: *Journal of Porous Materials* 25.2 (2018), pp. 537–554. ISSN: 1380-2224. DOI: 10.1007/s10934-017-0467-1. URL: http://link.springer.com/10.1007/s10934-017-0467-1.

[155] Pierre Soille. *Morphological Image Analysis*. Vol. 24. 11. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 4512–4527. ISBN: 978-3-662-03941-0. DOI: 10.1007/978-3-662-03939-7. arXiv: 1011.1669. URL: http://link.springer.com/10.1007/978-3-662-03939-7.

[156] B Sonon, B François, and T J Massart. "An advanced approach for the generation of complex cellular material representative volume elements using distance fields and level sets". In: *Computational Mechanics* 56.2 (2015), pp. 221–242. ISSN: 0178-7675. DOI: 10.1007/s00466-015-1168-8. URL: http://link.springer.com/10.1007/s00466-015-1168-8.

[157] A. Spettl et al. "Fitting Laguerre tessellation approximations to tomographic image data". In: *Philosophical Magazine* 96.2 (2016), pp. 166–189. ISSN: 14786443. DOI: 10.1080/14786435.2015.1125540. arXiv: 1508.01341. URL: http://dx.doi.org/10.1080/14786435.2015.1125540.

[158] Robert Staubs et al. "Parallel N-Dimensional Exact Signed Euclidean Distance Transform". In: *Insight Journal* 123 (2006), pp. 1–5. DOI: 10.54294/ogv879. URL: http://www.insight-journal.org/browse/publication/123.

[159] Han Sun, Jingyu Yang, and Mingwu Ren. "A fast watershed algorithm based on chain code and its application in image segmentation". In: *Pattern Recognition Letters* 26.9 (2005), pp. 1266–1274. ISSN: 01678655. DOI: 10.1016/j.patrec.2004.11.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167865504003617.

[160] Kirubel Teferra and Lori Graham-Brady. "Tessellation growth models for polycrystalline microstructures". In: *Computational Materials Science* 102 (2015), pp. 57–67. ISSN: 09270256. DOI: 10.1016/j.commatsci.2015.02.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0927025615000671.

[161] I. Temizer and T. I. Zohdi. "A numerical method for homogenization in nonlinear elasticity". In: *Computational Mechanics* 40.2 (2007), pp. 281–298. ISSN: 01787675. DOI: 10.1007/s00466-006-0097-y.

[162] George Teodoro et al. "A fast parallel implementation of queue-based morphological reconstruction using gpus". In: *Data- and Compute-Intensive Clinical and Translational Imaging Applications DCICTIA-MICCAI* (2012). URL: https://hgpu.org/?p=7089.

[163] William Thomson. "LXIII. On the Division of Space with Minimum Partitional Area". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 24.151 (1887), pp. 503–514.

[164] Volnei Tita et al. "Experimental analyses of the poly(vinyl chloride) foams' mechanical anisotropic behavior". In: *Polymer Engineering {&} Science* 52.12 (2012), pp. 2654–2663. ISSN: 00323888. DOI: 10.1002/pen.23222. URL: http://doi.wiley.com/10.1002/pen.23222.

[165] G Tondi et al. "X-ray microtomography studies of tannin-derived organic and carbon foams." In: *Microscopy and microanalysis : the official journal of Microscopy Society of America, Microbeam Analysis Society, Microscopical Society of Canada* 15.5 (2009), pp. 384–394. ISSN: 1435-8115. DOI: 10.1017/S1431927609990444. URL: http://www.ncbi.nlm.nih.gov/pubmed/19709464.

[166] M. W. D. Van Der Burg et al. "On the Linear Elastic Properties of Regular and Random Open-Cell Foam Models". en. In: *Journal of Cellular Plastics* 33.1 (Jan. 1997), pp. 31–54. ISSN: 0021-955X, 1530-7999. DOI: 10.1177/0021955X9703300103.

[167] L Vandenberghe. *The CVXOPT linear and quadratic cone program solvers*. 2010. URL: https://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf.

[168] Lieven Vandenberghe, Stephen Boyd, and Shao-Po Wu. *Determinant Maximization with Linear Matrix Inequality Constraints*. Stanford, 1998. DOI: 10.1137/S0895479896303430. URL: http://epubs.siam.org/doi/10.1137/S0895479896303430.

[169] Irene Vecchio, Claudia Redenbach, and Katja Schladitz. "Angles in Laguerre tessellation models for solid foams". In: *Computational Materials Science* 83 (2014), pp. 171–184. ISSN: 09270256. DOI: 10.1016/j.commatsci.2013.11.017. URL: http://dx.doi.org/10.1016/j.commatsci.2013.11.017.

[170] Irene Vecchio et al. "Improved Models of Solid Foams Based on Soap Froth". In: *Computational Materials Science* 120 (July 2016), pp. 60–69. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2016.03.029.

[171] Irene Vecchio et al. *MAVI - Modular Algorithms for Volume Images*. 2012. URL: https://www.itwm.fraunhofer.de/en/departments/bv/products-and-services/mavi.html.

[172] Matej Vesenjak et al. "Behaviour of cellular structures with fluid fillers under impact loading". In: *The International Journal of Multiphysics* 1.1 (2007), pp. 101–122. ISSN: 1750-9548. DOI: 10.1260/175095407780130508. URL: http://www.journal.multiphysics.org/index.php/IJM/article/view/1-1-101.

[173] C. Veyhl et al. "Finite Element Analysis of the Mechanical Properties of Cellular Aluminium Based on Micro-Computed Tomography". en. In: *Materials Science and Engineering: A* 528.13-14 (May 2011), pp. 4550–4555. ISSN: 09215093. DOI: 10.1016/j.msea.2011.02.031.

[174] H.I. Villafán-Vidales et al. "Heat Transfer Simulation in a Thermochemical Solar Reactor Based on a Volumetric Porous Receiver". en. In: *Applied Thermal Engineering* 31.16 (Nov. 2011), pp. 3377–3386. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2011.06.022.

[175] Luc. Vincent. "Exact Euclidean distance function by chain propagations". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 520–525. DOI: 10.1109/CVPR.1991.139746.

[176] Luc Vincent. "Morphological grayscale reconstruction in image analysis: applications and efficient algorithms". In: *IEEE Transactions on Image Processing* 2.2 (1993), pp. 176–201. ISSN: 10577149. DOI: 10.1109/83.217222. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=217222.

[177] F. Wan et al. "Experimental and computational micro-mechanical investigations of compressive properties of polypropylene/multi-walled carbon nanotubes nanocomposite foams". In: *Mechanics of Materials* 91.P1 (2015), pp. 95–118. ISSN: 01676636. DOI: 10.1016/j.mechmat.2015.07.004. URL: http://dx.doi.org/10.1016/j.mechmat.2015.07.004.

[178] J. Wang et al. "Feature-preserving surface reconstruction from unoriented, noisy point data". In: *Computer Graphics Forum* 32.1 (2013), pp. 164–176. ISSN: 14678659. DOI: 10.1111/cgf.12006.

[179] Meili Wang et al. "3D Incomplete Point Cloud Surfaces Reconstruction With Segmentation and Feature-Enhancement". In: *IEEE Access* 7 (2019), pp. 15272–15281. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2891959. URL: https://ieeexplore.ieee.org/document/8610208/.

[180] Wenping Wang, Jiaye Wang, and Myung-Soo Kim. "An algebraic condition for the separation of two ellipsoids". In: *Computer Aided Geometric Design* 18.6 (2001), pp. 531–539. ISSN: 01678396. DOI: 10.1016/S0167-8396(01)00049-8. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167839601000498.

[181] D. Weaire and R. Phelan. "A Counter-Example to Kelvin's Conjecture on Minimal Surfaces". en. In: *Philosophical Magazine Letters* 69.2 (Feb. 1994), pp. 107–110. ISSN: 0950-0839, 1362-3036. DOI: 10.1080/09500839408241577.

[182] T Wejrzanowski et al. "Structure of foams modeled by Laguerre-Voronoi tessellations". In: *Computational Materials Science* 67 (2013), pp. 216–221. ISSN: 09270256. DOI: 10.1016/j.commatsci.2012.08.046. URL: http://dx.doi.org/10.1016/j.commatsci.2012.08.046.

[183] Badadjida Wintiba et al. "An Automated Procedure for the Generation and Conformal Discretization of 3D Woven Composites RVEs". en. In: *Composite Structures* 180 (Nov. 2017), pp. 955–971. ISSN: 02638223. DOI: 10.1016/j.compstruct.2017.08.010.

[184] Yugong Wu et al. "Modeling and characterization of two-phase composites by Voronoi diagram in the Laguerre geometry based on random close packing of spheres". In: *Computational Materials Science* 47.4 (2010), pp. 951–961. ISSN: 09270256. DOI: 10.1016/j.commatsci.2009.11.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S092702560900439X.

[185] E Alper Yildirim. "On the Minimum Volume Covering Ellipsoid of Ellipsoids". In: *SIAM Journal on Optimization* 17.3 (2006), pp. 621–641. ISSN: 1052-6234. DOI: 10.1137/050622560. URL: http://epubs.siam.org/doi/abs/10.1137/050622560.

[186] Shunjie Yin and Nassif Rayess. "Characterization of Polymer-metal Foam Hybrids for Use in Vibration Dampening and Isolation". In: *Procedia Materials Science* 4 (2014), pp. 311–316. ISSN: 22118128. DOI: 10.1016/j.mspro.2014.07.564. URL: http://dx.doi.org/10.1016/j.mspro.2014.07.564.

[187] S Youssef, E Maire, and R Gaertner. "Finite element modelling of the actual structure of cellular materials determined by X-ray tomography". In: *Acta Materialia* 53.3 (2005), pp. 719–730. ISSN: 13596454. DOI: 10.1016/j.actamat.2004.10.024. URL: http://linkinghub.elsevier.com/retrieve/pii/S1359645404006330.

[188] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH". In: *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*. Vol. 1. New York, New York, USA: ACM Press, 1996, pp. 103–114. ISBN: 0897917944. DOI: 10.1145/233269.233324. URL: http://portal.acm.org/citation.cfm?doid=233269.233324.

[189] Yin Zhang and Liyan Gao. "On Numerical Solution of the Maximum Volume Ellipsoid Problem". In: *SIAM Journal on Optimization* 14.1 (2003), pp. 53–76. ISSN: 1052-6234. DOI: 10.1137/S1052623401397230. URL: http://epubs.siam.org/doi/abs/10.1137/S1052623401397230.

[190] H. X. Zhu and N. J. Mills. "The in-plane non-linear compression of regular honeycombs". In: *International Journal of Solids and Structures* 37.13 (2000), pp. 1931–1949. ISSN: 00207683. DOI: 10.1016/S0020-7683(98)00324-2.

[191] Wenqi Zhu et al. "Effective elastic properties of periodic irregular open-cell foams". In: *International Journal of Solids and Structures* 143 (2018), pp. 155–166. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2018.03.003. URL: https://doi.org/10.1016/j.ijsolstr.2018.03.003.

[192] Wenqi Zhu et al. "Micromechanical modeling of effective elastic properties of open-cell foam". In: *International Journal of Solids and Structures* 115-116 (2017), pp. 61–72. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2017.02.031. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020768317300859.

[193] Xiaolei Zhu et al. "A novel modeling approach of aluminum foam based on MATLAB image processing". In: *Computational Materials Science* 82 (2014), pp. 451–456. ISSN: 09270256. DOI: 10.1016/j.commatsci.2013.10.020. URL: http://linkinghub.elsevier.com/retrieve/pii/S092702561300637X.

[194] M A Zuluaga, L Ibañez, and F Peyrin. "Large Image Streaming using ITKv4". In: *The Insight Journal* January-Ju (2011), pp. 1–9. DOI: 10.54294/2djpwb. URL: http://hdl.handle.net/10380/3263.